Giansalvatore Mecca, Sergio Greco

SEBD 2011

Proceedings of the Nineteenth Italian Symposium on Advanced Database Systems

Maratea, Italy

June 26-29, 2011

Organized by

Università della Basilicata, Dipartimento di Matematica e Informatica

Sponsored by:

Dipartimento di Matematica e Informatica – Università della Basilicata

Facoltà di Scienze MM FF NN – Università della Basilicata

Università della Basilicata

DEIS – Università della Calabria

Program Committee

- Giansalvatore Mecca (Program Committee Chair) - *Università della Basilicata*

- Annalisa Appice - *Università di Bari*
- Maurizio Atzori - *Università di Cagliari*
- Elena Baralis - *Politecnico di Torino*
- Ilaria Bartolini - *Università di Bologna*
- Domenico Beneventano - *Università di Modena e Reggio Emilia*
- Francesco Bonchi - *Yahoo Research*
- Daniele Braga - *Politecnico di Milano*
- Andrea Calì - *University of London, Birkbeck College*
- Dario Colazzo - *Université Paris Sud - INRIA*
- Carlo Curino - *Massachusetts Institute of Technology (MIT)*
- Pasquale De Meo - *Università di Reggio Calabria*
- Paolino Di Felice - *Università dell'Aquila*
- Claudia Diamantini - *Università Politecnica delle Marche*
- Enrico Franconi - *Libera Università di Bolzano*
- Filippo Furfaro - *Università della Calabria*
- Giorgio Ghelli - *Università di Pisa*
- Giovanna Guerrini - *Università di Genova*
- Giuseppe Manco - *ICAR CNR*
- Michele Melchiori - *Università di Brescia*
- Massimo Melucci - *Università di Padova*
- Paolo Papotti - *Università Roma Tre*
- Antonella Poggi - *Università di Roma "La Sapienza"*
- Elisa Quintarelli - *Politecnico di Milano*
- Alessandra Raffaetà - *Università di Venezia*
- Yannis Velegrakis - *Università di Trento*
- Pierangelo Veltri - *Università Magna Graecia di Catanzaro*

Organizing Committee

- Sergio Greco (General Chair) - *Università della Calabria*

- Angela Bonifati, *ICAR CNR e Università della Basilicata*
- Carlo Sartiani, *Università della Basilicata*
- Donatello Santoro, *Università della Basilicata*

External Reviewers

- Devis Bianchini
- Giulia Bruno
- Luca Cagliero
- Marco Carnuccio
- Federico Cavalieri
- Michelangelo Ceci
- Augusto Celentano
- Gianni Costa
- Alfredo Cuzzocrea
- Claudia D'Amato
- Nicola Fanizzi
- Fabio Fassetti
- Emilio Ferrara
- Alessandro Fiori
- Giacomo Fiumara
- Sergio Flesca
- Alberto Grand
- Luigi Grimaudo
- Massimo Guarascio
- Francesco Gullo
- Pietro Guzzi
- Matteo Interlandi
- Elio Masciari
- Antonino Nocera
- Salvatore Orlando
- Riccardo Ortale
- Francesco Pagliarecci
- Themis Palpanas
- Francesco Parisi
- Laura Po
- Luigi Pontieri
- Domenico Potena
- Alessandro Provetti
- Andrea Pugliese
- Giovanni Quattrone
- Emanuele Rabosio
- Abdul Rahman Dannaoui
- Angelo Rauseo
- Ettore Ritacco
- Domenico Rosaci
- Silvia Rota
- Carlo Sartiani
- Claudio Silvestri
- Serena Sorrentino
- Emanuele Storti
- Giuseppe Tradigo
- Maurizio Vincini

Table of Contents

Invited Talks and Tutorials

Session 1: Probabilistic Databases and Data Uncertainty

Session 2: Clustering

Session 3: Data Mining I

Session 4: Data Integration & Exchange I

Session 5: Pervasive Computing & Context-Awareness

Session 6: Queries & Views

Session 7: Data Integration & Exchange II

Sessione 8: Interdisciplinary Approaches I

Session 9: Data Mining II

Session 10: XML, Semistructured & Hierchical Data

Session 11: Deductive Databases, Ontologies, and Description Logics

Session 12: Interdisciplinary Approaches II

Session 13: Web and Network Applications

Poster Session

Invited Talks and Tutorials

Tutorial

Semantic Constraints for Data Quality Assessment and Cleaning

Leo Bertossi

School of Computer Science, Carleton University

Abstract: Data quality is an increasingly important issue and concern in business intelligence. Data quality is most of the time a relative property since it largely depends on additional semantic information and metadata. In this tutorial we will review how semantic conditions expressed by means of integrity constraints, quality constraints, and contextual information can be used to characterize, assess and obtain quality data.

Bio Sketch: Leopoldo Bertossi has been Full Professor at the School of Computer Science, Carleton University (Ottawa, Canada) since 2001. He is Faculty Fellow of the IBM Center for Advanced Studies (IBM Toronto Lab). He obtained a PhD in Mathematics from the Pontifical Catholic University of Chile (PUC) in 1988. He has been the theme leader for "Adaptive Data Quality and Data Cleaning" of the "NSERC Strategic Network for Data Management for Business Intelligence" (BIN), a project that involves more than fifteen academic researchers across Canada plus several industrial partners. Prof. Bertossi's research interests include database theory, data integration, peer data management, semantic web, intelligent information systems, data quality for business intelligence, knowledge representation, logic programming, and computational logic.

Tutorial

View-Based XML Rewriting

Ioana Manolescu

INRIA/LRI Leo team, INRIA

Abstract: The performance of XQuery evaluation can be greatly improved by using materialized views. To do so, one must be able, given a set of view definitions and a query, to identify the possible ways in which the query can be rewritten based on the views. The topic of the tutorial is to present the main algorithmic approaches and complexity results presented in the literature, concerning the equivalent rewriting of XML queries specified in various dialects of XQuery, using XML materialized views.

Bio Sketch: Ioana Manolescu is a senior researcher at INRIA Saclay and the head of the Leo group, joint with University of Paris Sud-XI, in France. Her main research interests concern efficient management of data, in particular in distributed architectures, and applied to Web data.

Invited Talk

Evolution and merging of real-life ontologies

Erhard Rahm

Database Group, University of Leipzig

Abstract: Ontologies are in wide-spread use in diverse domains. In life sciences, many large ontologies are used to annotate biomedical entities and perform analysis tasks such as functional profiling or term enrichment. On the web, simple ontologies such as web directories or product catalogs are heavily used for improved content categorization and search. These ontologies are subject to significant reorganizations and other evolutionary changes. There is thus an increasing need to better deal with ontology evolution, in particular to support the automatic detection of evolution mappings and to automate the migration of ontology instances and ontology-based mappings. Another common task is to combine or merge multiple related ontologies. In the talk we present new Match-based approaches to determine a Diff and a Merge between ontologies. The proposed COntoDiff scheme is rule-based and determines compact evolution mapping consisting of expressive change operations. We also point out open challenges for future work.

Bio Sketch: Erhard Rahm is a full professor for computer science at the University of Leipzig, Germany. He chairs the database group and a new innovation lab on Web Data Integration (WDI Lab). His Ph.D. and habilitation degrees are from the University of Kaiserslautern. He held visiting research positions at IBM Research and at Microsoft Research. His current work areas include data integration, metadata management and bioinformatics. Professor Rahm published numerous peer-reviewed research papers and authored or co-edited several books, including the 2011 Springer book on "Schema Matching and Mapping". At VLDB 2011, he will receive the VLDB 10 Year Best Paper Award for a paper on schema matching.

Invited Talk

Querying and Reasoning about Massive Social Networks

VS Subramahnian

Department of Computer Science, University of Maryland

Abstract: Companies and organizations are becoming increasingly aware of the value of social networks to their businesses. In this talk, I will discuss results on three kinds of problems related to social networks. First, I will briefly summarize methods to query social networks using subgraph matching query paradigms. Though subgraph matching is intractable, I will briefly discuss cloud based approaches that can process complex subgraph queries on real social networks of over one billion edges in under a second. Second, I will discuss a class of problems called social network optimization problems - problems related to allocating resources across a social network, taking a diffusion model of some phenomena into account. Finally, I will discuss a class of problems called competitive diffusion problems - how does a phenomenon diffuse across a network when there are competing diffusions occurring at the same time. Parts of this talk reflect joint work with Matthias Broecheler, Andrea Pugliese, and Paulo Shakarian.

Bio Sketch: V.S. Subrahmanian is Professor of Computer Science and Director of the Center for Digital International Government and Co-Director of the Lab for Computational Cultural Dynamics at the University of Maryland where he has been on the faculty since 1989. He has worked extensively on databases and artificial intelligence and has co-authored over 200 papers as well as several books. He has served on the editorial board of several journals, has won numerous awards, and delivered invited talks at numerous conferences. His work has been extensively cited both in the academic literature as well as in the press.

Regular Papers

Getting the Best from Uncertain Data

Ilaria Bartolini, Paolo Ciaccia, and Marco Patella
DEIS - Università di Bologna, Italy
{i.bartolini,paolo.ciaccia,marco.patella}@unibo.it

Abstract. The skyline of a relation is the set of tuples that are not dominated by any other tuple in the same relation, where tuple u dominates tuple v if u is no worse than v on all the attributes of interest and strictly better on at least one attribute. Previous attempts to extend skyline queries to probabilistic databases have proposed either a weaker form of domination, which is unsuitable to univocally define the skyline, or a definition that implies algorithms with exponential complexity. In this paper we demonstrate how, given a semantics for linearly ranking probabilistic tuples, *the skyline of a probabilistic relation can be univocally defined*. Our approach preserves the three fundamental properties of skyline: 1) it equals the union of all top-1 results of monotone scoring functions, 2) it requires no additional parameter to be specified, and 3) it is insensitive to actual attribute scales. We also detail efficient sequential and index-based algorithms.

1 Introduction

Uncertain data management has recently become a very active area of research, due to the huge number of relevant applications in which uncertainty plays a key role, such as data extraction from the Web, data integration, biometric systems, sensor network readings, etc. Further, uncertainty might also occur as a result of data anonymization.

According to a commonly adopted model, uncertain data can be represented through *probabilistic relations*, in which each tuple has also a probability (confidence) to appear [11, 12]. A probabilistic relation compactly represents a set of *possible worlds*, i.e., subsets of tuples. In the general case, the formation of possible worlds is constrained by a set of generation rules, that are used to model correlation among tuples (e.g., a rule might state that two tuples are mutually exclusive).

In recent years, several works have focused on extending different query types to probabilistic databases. Among them, in this paper we concentrate on *skyline queries*, whose relevance in supporting multi-criteria decision analysis is well known [3]. The skyline of a relation R is the set of undominated (or Pareto-optimal) tuples in R, where tuple u dominates tuple v if u is no worse than v on all the attributes of interest, and strictly better than v on at least one attribute. The appeal of skyline queries comes from the observation that the skyline consists of all and only top-1 results obtainable from scoring functions that are monotone in the skyline attributes, thus providing users with an overall picture of what are the best alternatives in a relation. Further, unlike top-k queries, a skyline query does not require any input parameter to be specified. Not less important is also the fact that the skyline is insensitive to attributes' scales, being it only dependent on the relative ordering of tuples on each attribute.

As a motivating example, consider a traffic-monitoring application collecting data by means of a radar, a sample of which is shown in Figure 1.[1] Each radar reading has associated a Prob value, representing the overall confidence one has in the reading itself. A skyline query on the Time and Speed attributes would return those tuples (i.e., readings) that are, at the same time, the most recent ones and that concern high-speed cars. In the deterministic case it would be $\textsc{Sky}(R) = \{t_1, t_2\}$, as it can be easily verified from the figure on the right. In the probabilistic case, in which also the confidence of each tuple has to be considered, even *defining* what the skyline should be is challenging.

TID	Plate No	Time	Speed	Prob
t_1	X-123	11:50	145	0.4
t_2	W-246	11:40	160	0.3
t_3	Z-456	11:15	145	0.6
t_4	H-121	11:05	137	0.4
t_5	Y-324	11:00	140	0.6
t_6	X-827	10:50	135	0.4
t_7	C-442	10:45	155	0.5

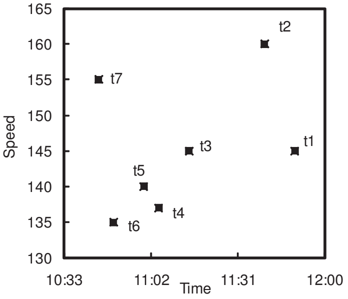


Fig. 1. A probabilistic relation

1.1 Related Work

The first work to consider skyline queries on probabilistic data has been [10]. There, the basic idea is to compute for each tuple u the probability, $\Pr_{\textsc{Sky}}(u)$, that u is undominated, and then rank tuples based on these *skyline probabilities*. Intuitively, $\Pr_{\textsc{Sky}}(u)$ equals the overall probability of the possible worlds W in which u is in the (deterministic) skyline of W. The p-skyline of a probabilistic relation is then defined as the set of tuples whose skyline probability is at least p. This approach is unable to preserve the basic skyline properties, since it requires an additional parameter (the p threshold), and has no apparent relationship with the results of top-1 queries. Subsequent works on the subject have provided efficient algorithms to compute all skyline probabilities [1], and shown how to compute p-skylines on uncertain data streams [14]. More recently, Lin et al. have proposed the *stochastic skyline operator* [9]. Unlike p-skyline, the stochastic skyline has the advantage of not requiring any parameter. However, this comes at the price of an algorithmic exponential complexity, since testing stochastic domination is an NP-complete problem. Further, the stochastic skyline equals only a subset of possible top-1 results, namely those arising from the *expectation of multiplicative scoring functions*.

1.2 Contributions

In this paper we address the problems of *defining and efficiently computing the skyline of a probabilistic relation*. We start by providing in Section 3 a formal

[1] A similar example was also used in previous works on top-k queries [12, 8].

definition of skyline, which is based on a generalization to the probabilistic case of the concept of domination among tuples. The P-domination relationship we introduce to this purpose is formally grounded in order theory, and satisfies all the properties the skyline has in the deterministic case. Since P-domination is parametric in the semantics used to rank probabilistic tuples, this implies that, whatever ranking semantics for top-k queries one wants to adopt, our skyline definition will be always consistent with it, which is a remarkable property.

In Section 4 we show how the skyline can be computed in $\mathcal{O}(n^3)$ time for a relation with n tuples, by detailing the analysis for the case in which the "expected rank" semantics is used for ranking tuples [5]. In Section 5 we describe algorithms aiming to reduce the actual response time. Experimental evaluation on large datasets shows the practical applicability of our approach.

For lack of space, we only consider probabilistic relations in which tuples are pairwise independent, i.e., no generation rule is present; however, our results can be also smoothly extended to the correlated case.

2 Preliminaries

We model a probabilistic relation R^p as a pair, $R^p = (R, p)$, where R is a relation in the standard sense, also called a *deterministic* relation, and p is a function that assigns to each tuple $u \in R$ a probability, $p(u) \in (0, 1]$. A *posssible world* W of R^p is any subset of tuples from R. The set of possible worlds of R^p is denoted $\mathcal{W}$. The probability of possible world W is computed as: $\Pr(W) = \prod_{u \in W} p(u) \prod_{v \notin W}(1 - p(v))$.

Given a (deterministic) relation R whose schema includes a set of numerical attributes $\mathcal{A} = \{A_1, A_2, \ldots, A_d\}$, the *skyline* of R with respect to $\mathcal{A}$, denoted $\textsc{Sky}_{\mathcal{A}}(R)$ or simply $\textsc{Sky}(R)$, is the set of *undominated* tuples in R. Assuming that on each attribute higher values are preferable, tuple u (Pareto-)dominates tuple v, written $u \succ v$, iff it is $u.A_i \geq v.A_i$ for each $A_i \in \mathcal{A}$ and there exists at least one attribute A_j such that $u.A_j > v.A_j$. Thus:

$$\textsc{Sky}(R) = \{u \in R \mid \nexists\, v \in R : v \succ u\} \tag{1}$$

If neither $u \succ v$ nor $v \succ u$ hold, then u and v are *indifferent*, written $u \sim v$.

A *scoring function* $s()$ on the attributes $\mathcal{A}$, $s : dom(\mathcal{A}) \to \Re$, is *monotone* iff $u.A_i \geq v.A_i$ $(i = 1, \ldots, d)$ implies $s(u) \geq s(v)$. Although it is folklore that $\textsc{Sky}(R)$ equals the union of top-1 results of monotone scoring functions, this is imprecise because of the non-deterministic nature of top-1 queries. For instance, consider the max function, which is monotone, and $R = \{(3, 4), (1, 4)\}$. Although $(3, 4) \succ (1, 4)$, it is $\max\{3, 4\} = \max\{1, 4\}$, thus $(1, 4)$ might be (non-deterministically) returned as the top-1 result. To obviate the problem, in this paper we only consider monotone functions that are also *domination-preserving*, i.e., $u \succ v$ implies $s(u) > s(v)$.[2] In the following, we always implicitly assume that a monotone function is also domination-preserving.

[2] Domination-preserving monotone functions are exactly those functions that Fagin et al. call *strictly monotone in each argument* [6].

3 The Skyline of a Probabilistic Relation

In order to define the skyline of a probabilistic relation we start by rewriting Equation 1 as:

$$\boxed{\textsc{Sky}(R^p) = \{u \in R \mid \nexists\, v \in R : v \succ_p u\ \}} \tag{2}$$

in which the only difference with the deterministic case is that $\succ$ is substituted by $\succ_p$. We call $\succ_p$ probabilistic domination, or *P-domination* for short. Note that $\succ_p$ is a binary relation in the standard sense, i.e., no probability is present in $\succ_p$.

In order to define P-domination so as to preserve all skyline properties, we approach the problem by considering things from an order-theoretic viewpoint. In order-theoretic terms, $\succ$ is a *strict partial order*, i.e., an irreflexive ($\forall u : u \not\succ u$) and transitive ($\forall u, v, t : u \succ v \wedge v \succ t \Rightarrow u \succ t$) relationship on the domain of skyline attributes. A *linear* order $\gtrdot$ is a strict partial order that is also *connected*, i.e., for any two distinct tuples u and v, either $u \gtrdot v$ or $v \gtrdot u$.[3] A linear order $\gtrdot$ is called a *linear extension* of $\succ$ iff $u \succ v \Rightarrow u \gtrdot v$, i.e., $\gtrdot$ is *compatible* with $\succ$. Notice that a linear extension of $\succ$ can be obtained by ordering tuples with a monotone scoring function and then breaking ties arbitrarily.

Let $\textsc{Ext}(\succ)$ denote the set of all linear extensions of $\succ$. A fundamental result in order theory, derived from *Szpilrajn's Theorem* [13], asserts that *any strict partial order* $\succ$ *equals the intersection of its linear extensions*, $\succ = \bigcap\{\gtrdot \mid \gtrdot \in \textsc{Ext}(\succ)\}$. This is the first ingredient needed to define P-domination.

Our second ingredient comes from the observation that *each linear order* $\gtrdot$ *on the tuples of* R *can be used to define a corresponding linear order on the probabilistic tuples of* R^p. Indeed, this has been the subject of several recent works aiming to support top-k queries on uncertain data, which has lead to different, alternative semantics for ranking tuples that come with both a score and a probability [12, 15, 5]. In abstract terms, each of these semantics can be viewed as a *probabilistic ranking function* Ψ that, given a linear order $\gtrdot$ on the tuples of R and a probability function p, yields a *probabilistic linear order* $\gtrdot_p = \Psi(\gtrdot, p)$ on the probabilistic tuples of R^p. In practice, any ranking semantics assigns to each tuple u a value $\psi(u)$, so that $u \gtrdot_p v$ iff $\psi(u) > \psi(v)$.[4]

We are now ready to define P-domination:

Definition 1 (P-domination). *Let $R^p = (R, p)$ be a probabilistic relation, and let $\succ$ be the Pareto-domination relationship on the tuples in R when considering the skyline attributes $\mathcal{A}$. Let Ψ be a probabilistic ranking function on R^p. For any two tuples u and v in R^p, we say that u* P-dominates *v, written $u \succ_p v$, iff for each linear extension $\gtrdot$ of $\succ$, with associated probabilistic linear order $\gtrdot_p = \Psi(\gtrdot, p)$, it is $u \gtrdot_p v$, that is:*

$$\boxed{u \succ_p v \iff u \gtrdot_p v,\ \ \forall \gtrdot_p = \Psi(\gtrdot, p), \gtrdot \in \textsc{Ext}(\succ)} \tag{3}$$

[3] To denote linear orders over tuples we use the symbol $\gtrdot$ in place of the usual $>$, and reserve the latter for the standard order on real numbers.

[4] In the most general case, Ψ might also depend on the actual scores of the tuples, rather that only on their ordering. This has no influence on the results we derive.

The diagram in Figure 2 summarizes how $\succ_p$ is conceptually obtained: from $\succ$ we obtain a set of linear orders, and for each of them a corresponding probabilistic linear order. The intersection of such probabilistic rankings yields P-domination.

$$\begin{tikzcd}
\succ \arrow[r, "\textsc{Ext}(\succ)"] \arrow[d, "\Psi{,}p"] & \{\gtrdot\} \arrow[d, "\Psi{,}p"] \\
\succ_p & \{\gtrdot_p\} \arrow[l, "\bigcap"']
\end{tikzcd}$$

Fig. 2. How P-domination is obtained

From Definition 1 three major results follow:[5]

Theorem 1. *For any probabilistic ranking function Ψ, the corresponding P-domination relationship $\succ_p$ is a strict partial order.*

Theorem 2. *Let $\textsc{Sky}(R^p)$ be the skyline of R^p, for a given probabilistic ranking function Ψ. A tuple u belongs to $\textsc{Sky}(R^p)$ iff there exists a monotone scoring function $s()$ such that u is the top-1 tuple according to the probabilistic linear order $\gtrdot_p = \Psi(\gtrdot, p)$, where $\gtrdot$ is the linear order induced by $s()$ on R.*

A further important property of $\textsc{Sky}(R^p)$ is that, as in the deterministic case, it is insensitive to actual attribute values, rather it only depends on the relative ordering on each skyline attribute.

Theorem 3. *Let $R^p = (R, p)$ be a probabilistic relation, and $S^p = (S, p)$ be another probabilistic relation, in which S is obtained from R through an isomorphism ϕ that preserves Pareto domination (i.e., for any two tuples $u, v \in R$ it is $u \succ v$ if and only if $\phi(u) \succ \phi(v)$), and $p(u) = p(\phi(u))$ for all $u \in R$. Then, for any probabilistic ranking function Ψ, it is $\textsc{Sky}(R^p) = \textsc{Sky}(S^p)$.*

4 Computing P-domination

Definition 1 cannot be directly used to check P-domination, since it requires to enumerate all linear extensions of the Pareto dominance relationship, and these can be exponential in the number of tuples.[6] In the following we first sketch how, independently of the specific probabilistic ranking function Ψ, P-domination can be checked without materializing the linear extensions of $\succ$, after that we detail the analysis for the case of in which Ψ is the “expected rank” semantics [5].

Consider a linear extension $\gtrdot$ of $\succ$, and let $\psi_{\gtrdot}(u)$ be the numerical value that Ψ assigns to tuple u. According to Definition 1, for $u \succ_p v$ to hold it has to be $\psi_{\gtrdot}(u) > \psi_{\gtrdot}(v)$ for all linear extensions $\gtrdot$ of $\succ$, that is:

$$u \succ_p v \iff \min_{\gtrdot \in \textsc{Ext}(\succ)} \left\{ \frac{\psi_{\gtrdot}(u)}{\psi_{\gtrdot}(v)} \right\} > 1 \tag{4}$$

[5] For lack of space, all formal results are stated without proof.

[6] If R^p consists of n pairwise indifferent tuples, then $\succ$ is empty and $\textsc{Ext}(\succ)$ has size $n!$, since each permutation is compatible with $\succ$.

The key idea for efficiently checking the above inequality is to determine which is the linear order that is the most unfavorable one for u with respect to v. If $\psi_{\gtrdot}(u) > \psi_{\gtrdot}(v)$ holds for this "extremal" order, then it will necessarily hold for all other orders compatible with $\succ$. Regardless of the specific probabilistic ranking function Ψ, the two relevant cases to consider here are:

$\mathbf{u} \succ \mathbf{v}$: When u dominates v, we can restrict the analysis to those linear orders for which it is $u \gtrdot v$; starting from this we analyze how other tuples should be arranged in the linear order so as to minimize the ratio $\psi_{\gtrdot}(u)/\psi_{\gtrdot}(v)$.

$\mathbf{u} \not\succ \mathbf{v}$: If u does not dominate v, then the worst case for u and the best one for v corresponds to a linear order in which: 1) $u \gtrdot t$ only for those tuples t that u dominates, and 2) $t' \gtrdot v$ only for those tuples t' that dominate v.

4.1 P-domination with Expected Ranks

According to [5], the result of a top-k query on a probabilistic relation R^p is based on the concept of *expected rank*. Given a linear order $\gtrdot$ on the tuples of R, the rank of u in a possible world W with $|W|$ tuples is the number of tuples in W that precedes u, that is:

$$\text{rank}_{W,\gtrdot}(u) = \begin{cases} |\ \{t \in W \mid t \gtrdot u\}\ | & \text{if } u \in W \\ |W| & \text{otherwise} \end{cases}$$

Thus, ranks range from 0 to $|W| - 1$, and tuples not in W have rank $|W|$. The expected rank of u is then defined as $ER_{\gtrdot}(u) = \sum_{W \in \mathcal{W}} \text{rank}_{W,\gtrdot}(u) \times \Pr(W)$.

As in [5], we consider that if two tuples have a same expected rank value, a tie-breaking rule is applied so that expected ranks define a linear order. Let $\gtrdot_p$ be such linear order, i.e.,: $u \gtrdot_p v$ iff $ER_{\gtrdot}(u) < ER_{\gtrdot}(v)$.

As explained in [5], the expected rank of a tuple u can be computed as:

$$ER_{\gtrdot}(u) = p(u) \times \sum_{t \gtrdot u} p(t) + (1 - p(u)) \times \sum_{t \neq u} p(t) \tag{5}$$

where the first term is the expected rank of u in a possible world in which u appears, and the second sum is the expected size of a possible world in which u does not appear.

Let P be the overall probability of all the tuples in R, $P = \sum_{t \in R} p(t)$, and let $H_{\gtrdot}(u) = \sum_{t \gtrdot u} p(t)$ be the overall probability of those tuples that are better than u according to $\gtrdot$. A key observation that will be exploited in the following is that, for any linear order $\gtrdot$ that extends $\succ$, it is $H_{\gtrdot}(u) \in [H^-(u), H^+(u)]$, where the two bounds are respectively defined as:

$$H^-(u) = \sum_{t \succ u} p(t) \qquad\qquad H^+(u) = \sum_{\substack{u \not\succ t \\ t \neq u}} p(t) = P - p(u) - \sum_{u \succ t} p(t)$$

Notice that $H^-(u)$ is the best possible case for u, in which only those tuples that dominate u are also better than u according to $\gtrdot$, whereas the worst possible case for u is given by a linear order in which u is better only of those tuples that it dominates. Equation 5 can then be compactly rewritten as:

$$ER_{\gtrdot}(u) = p(u) \times H_{\gtrdot}(u) + (1 - p(u)) \times (P - p(u))$$

According to Definition 1, it has to be $ER_{\gtrdot}(u) < ER_{\gtrdot}(v)$ for each linear order $\gtrdot$ that extends $\succ$, i.e.:

$$\max_{\gtrdot \in \textsc{Ext}(\succ)} \left\{ \frac{p(u) \times H_{\gtrdot}(u) + (1 - p(u)) \times (P - p(u))}{p(v) \times H_{\gtrdot}(v) + (1 - p(v)) \times (P - p(v))} \right\} < 1$$

Let $P_{u,v} = P - p(u) - p(v)$. Substituting, simplifying, and rearranging terms, above inequality can be equivalently written as:

$$u \succ_p v \Leftrightarrow \boxed{\frac{p(u)}{p(v)} > \max_{\gtrdot \in \textsc{Ext}(\succ)} \left\{ \frac{P_{u,v} + 1 - H_{\gtrdot}(v)}{P_{u,v} + 1 - H_{\gtrdot}(u)} \right\}} \tag{6}$$

The two cases to be considered for Equation 6 are dealt with as follows.

$\mathbf{u} \succ \mathbf{v}$: Since u dominates v, and domination is transitive, it is $H_{\gtrdot}(v) \geq H_{\gtrdot}(u) + p(u)$ for each $\gtrdot \in \textsc{Ext}(\succ)$. This ensures that the right-hand side of Equation 6 is strictly less than 1, which immediately yields the first P-domination rule:

$$\boxed{u \succ v \ \wedge \ \frac{p(u)}{p(v)} \geq 1} \tag{Rule 1}$$

Note that this perfectly matches the intuition that a more likely and better tuple should probabilistically dominate a less likely and worse tuple.

When $p(u) < p(v)$, we can maximize the right-hand side of Equation 6 as follows. For any tuple t such that $u \succ t$, yet t is indifferent to v, $t \sim v$, we set $v \gtrdot t$, so as not to increase the value of $H_{\gtrdot}(v)$. For a tuple t which is indifferent to both u and v there are two alternatives to consider: either $t \gtrdot u \gtrdot v$ or $u \gtrdot v \gtrdot t$. In the first case we would add $p(t)$ to both $H_{\gtrdot}(v)$ and $H_{\gtrdot}(u)$, but this would *lower* the ratio in the right-hand side of Equation 6. Thus, we conclude that the second alternative is the one to be chosen. Finally, consider a tuple t such that $t \succ v$, yet $t \sim u$. In this case we set $t \gtrdot u$, so as to increase the value of $H_{\gtrdot}(u)$ (notice that $H_{\gtrdot}(v)$ already includes $p(t)$, since $t \succ v$).

Combining the above cases, it is evident that it is $H_{\gtrdot}(v) = H^{-}(v)$. On the other hand, for $H_{\gtrdot}(u)$ we have to add to $H^{-}(u)$ the mass of probability of all those tuples t such that $t \succ v$ and $t \sim u$, that is: $H_{\gtrdot}(u) = H^{-}(u) + \sum_{\substack{t \succ v \\ t \sim u}} p(t)$. By partitioning the set of tuples that dominate v depending on their relationship with respect to u, the following identity is derived:

$$H^{-}(v) = H^{-}(u) + p(u) + \sum_{\substack{t \succ v \\ t \sim u}} p(t) + \sum_{\substack{t \succ v \\ u \succ t}} p(t)$$

Letting $IbP(u, v) = \sum_{\substack{t \succ v \\ u \succ t}} p(t)$ to stand for the *in-between mass of probability* of those tuples that dominate v and are dominated by u we obtain:

$$H_{\gtrdot}(u) = H^{-}(v) - IbP(u, v) - p(u)$$

from which we get the second P-domination rule:

$$\boxed{u \succ v \ \wedge \ \frac{p(u)}{p(v)} > \frac{P_{u,v} + 1 - H^{-}(v)}{P_{u,v} + 1 - H^{-}(v) + IbP(u,v) + p(u)}} \tag{Rule 2}$$

Rule 2 generalizes Rule 1, which is therefore redundant. However we keep it since, unlike Rule 2, it can be checked without the need to compute any bound.

$\mathbf{u} \not\succ \mathbf{v}$: P-domination can occur even when $u \not\succ v$, provided $p(u) > p(v)$. In this case it is immediate to see that the right-hand side of Equation 6 is maximized by setting $H_{\gtrdot}(v) = H^{-}(v)$ and $H_{\gtrdot}(u) = H^{+}(u)$, thus:

$$\boxed{u \not\succ v \ \wedge \ \frac{p(u)}{p(v)} > \frac{P_{u,v} + 1 - H^{-}(v)}{P_{u,v} + 1 - H^{+}(u)}} \tag{Rule 3}$$

Example 1. Table 1 lists the probabilities of the tuples in Figure 1, whose overall probability is $P = 3.2$, together with their H^{-} and H^{+} bounds. As an example of how bounds are computed consider tuple t_3. Since t_3 is dominated only by t_1 and t_2, it is $H^{-}(t_3) = p(t_1) + p(t_2) = 0.7$. The tuples dominated by t_3 are t_4, t_5, and t_6, thus $H^{+}(t_3) = P - p(t_3) - p(t_4) - p(t_5) - p(t_6) = 1.2$. A case to which Rule 1 applies concerns tuples t_1 and t_4, since it is $t_1 \succ t_4$ and $p(t_1) \geq p(t_4)$, thus $t_1 \succ_p t_4$. Rule 2 is used to discard tuple t_5, which is P-dominated by t_1 (notice that here it is $p(t_1) < p(t_5)$, and $IbP(t_1, t_5) = 0.6$). A case in which Rule 3 is satisfied regards tuples t_3 and t_2 (notice that t_2 is part of the deterministic skyline). An exhaustive analysis shows that $\textsc{Sky}(R^p) = \{t_1, t_3, t_7\}$. □

tuple	t_1	t_2	t_3	t_4	t_5	t_6	t_7
probability	0.4	0.3	0.6	0.4	0.6	0.4	0.5
H^{-}	0	0	0.7	1.3	1.3	2.3	0.3
H^{+}	0.8	0.4	1.2	2.4	2.2	2.8	2.7

Table 1. Probabilities and bounds for the dataset in Figure 1

5 Algorithms

The skyline of a probabilistic relation R^p consisting of n tuples can be computed in $\mathcal{O}(n^3)$ time, since checking P-domination between two tuples is in $\mathcal{O}(n)$. The basic idea to reduce the actual running time is to use a 2-phase algorithm, whose general schema goes as follows. In the first phase, for each tuple u we compute the bounds $H^{-}(u)$ and $H^{+}(u)$, which requires $\mathcal{O}(n^2)$ time overall. In the second phase we actually compare tuples, and also compute the in-between probabilities, $IbP(u, v)$, for all pairs of tuples such that $u \succ v$ yet $p(u) < p(v)$.

We consider several variants of this basic schema. As a preliminary observation, it has to be remarked that the pre-processing step of topologically sorting the input relation R^p, so that a tuple u dominating v can never follow v, which is commonly used in the deterministic case [2, 4] (since it leads to a reduction of the number of comparisons and simplifies the management of the result set, that can only grow in size), would not provide such guarantees in our scenario. This is because, as explained in Section 4, it could well be the case that $u \succ_p v$ even if $u \not\succ v$. However, as detailed below, sorting can be exploited to speed up the computation of the H^- and H^+ bounds and of the quantities $IbP(u, v)$.

The baseline algorithm for computing the bounds $H^-(u)$ and $H^+(u)$ precisely follows their definition, given in Section 4, thus tuples in R^p are sequentially accessed and compared with all already encountered tuples. The number of comparisons is thus $n(n-1)/2$. Topologically sorting R^p only slightly reduces the running time, since if v follows u in the order then we can only conclude that v is not needed to compute $H^-(u)$.

Once all bounds are computed, the second phase of the algorithm can start, in which tuples are actually compared. Algorithm 1 resembles the well-known BNL algorithm for computing the skyline of a (non-probabilistic) relation [3]. Each tuple u of R^p is compared to all the tuples v currently in the skyline: for this, the quantity $IbP(u, v)$ (or $IbP(v, u)$) is computed at line 5/6 (only if $u \succ v$, or $v \succ u$, and Rule 1 of P-domination fails). If $u \succ_p v$, then v can be dropped from $\text{Sky}(R^p)$ (line 7); otherwise, if $v \succ_p u$, then u cannot be part of the skyline (line 8) and the loop terminates.

Algorithm 1 Tuple comparison

```
Input: probabilistic relation $R^p$, each tuple $u$ in $R^p$ includes bounds $H^+(u)$ and $H^-(u)$
Output: $\text{Sky}(R^p)$, the skyline of $R^p$
1: $\text{Sky}(R^p) \leftarrow \emptyset$
2: for all tuples $u \in R^p$ do
3:     $insert \leftarrow$ true
4:     for all tuples $v \in \text{Sky}(R^p)$ do
5:         if $u \succ v \wedge p(u) < p(v)$ then $IbP(u, v) \leftarrow computeIbP(R^p, u, v)$
6:         else if $v \succ u \wedge p(v) < p(u)$ then $IbP(v, u) \leftarrow computeIbP(R^p, v, u)$
7:         if $u \succ_p v$ then $\text{Sky}(R^p) \leftarrow \text{Sky}(R^p) \setminus \{v\}$
8:         else if $v \succ_p u$ then $insert \leftarrow$ false, continue (goto 9)
9:     if $insert$ then $\text{Sky}(R^p) \leftarrow \text{Sky}(R^p) \cup \{u\}$
```

Again, a topological sort of R^p guarantees that the test at line 5 in Algorithm 1 is never satisfied, thus we could obtain a faster execution of the algorithm. The computation of the value $IbP(u, v)$ can be performed in a trivial way by following the definition in Section 4, i.e., by checking if any tuple t in R^p satisfies $u \succ t \succ v$. If R^p is topologically sorted, then all such tuples can only be found between u and v, i.e., if i and k, respectively, are the indices of u and v in the sorted R^p, then we need only to check those tuples t_j such that $i < j < k$. As an alternative implementation, we could also exploit a spatial index, able to efficiently solve window queries, i.e., to find all tuples included in a hyper-rectangular region of the attribute space $\mathcal{A}$. In particular, we use an R-tree [7] for retrieving all tuples in a window whose opposite vertices consist of the coordinates of tuples v and u, respectively: $IbP(u,v)$ can then be computed by simply summing up probabilities of result tuples.[7]

6 Experimental Evaluation

In this section we experimentally analyze the efficiency of the proposed algorithms for the computation of the skyline of a probabilistic relation. For this, we synthetically generated 100,000 4-D tuples with uniformly distributed coordinates and probability. We then contrasted the performance of the algorithms described in Section 5 when varying the data dimensionality d (only the first 2-4 coordinates are used for checking domination) and/or the data cardinality n (only a fraction of the dataset is used).

As a first result, we show in Table 2 the size of $\textsc{Sky}(R^p)$ for different values of d and n: this demonstrates the fact that, at least for these datasets, the skyline has always a reasonable size, thus it makes sense to actually investigate the efficiency of alternative algorithms for computing it.

$d \setminus n$	20K	40K	60K	80K	100K
2	24	29	29	32	30
3	126	154	173	186	172
4	435	619	666	781	792

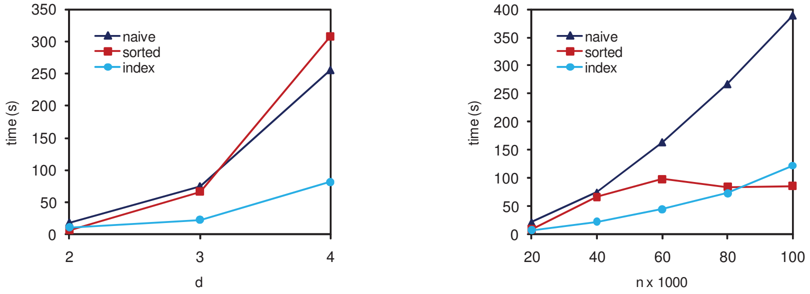
Table 2. The size of $\textsc{Sky}(R^p)$ for different values of d and n

In our next experiment, we evaluate the effect of topologically sorting the dataset in the first phase of the algorithm, when the H^- and the H^+ bounds are computed. As expected, sorting R^p only leads to a minor time saving: on average, the sort-based version of the algorithm is only 4% faster, with a maximum time saving of 10% (for $d = 3$ and $n = 40K$).

We then compare the performance of three variants of Algorithm 1. The variants we tested are as follows: the naive variant uses a simple loop for computing $IbP(u,v)$, sorted exploits a topological sort of R^p, so that only tuples between u and v are checked, and index uses an R-tree built on R^p. Figure 3 shows elapsed times for the three algorithms. As a first observation, we note that the index algorithm is consistently better than naive, saving around 70% of time, and that this does not depend on the data cardinality: such saving is the one provided by the index in computing the $IbP(u,v)$ values. A second, more interesting, evidence is that performance of the sorted algorithm actually improves when incrementing the dataset size: this behavior is likely due to the use of cache memory, since the comparison of consecutive tuples with a same skyline tuple requires checking almost the same sets of tuples, thus likely producing several cache hits.

Our final experiment investigates the effect of the three rules for checking P-domination between tuples. In Figure 4 (a) we show effectiveness of each rule

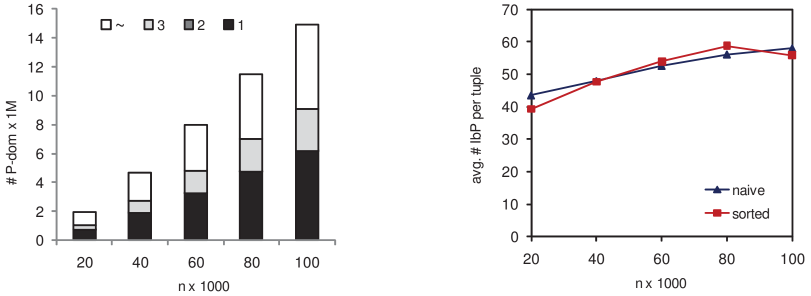
[7] Using a spatial index for the computation of the H^- and H^+ bounds would not be efficient, since it would require solving window queries with very low selectivity, unless $\mathcal{A}$ has a high dimensionality; in that case, however, the curse of dimensionality would hinder index performance.



(a) (b)

Fig. 3. Execution times for the three variants of Algorithm 1 vs. (a) dataset dimensionality ($n = 40K$) and (b) dataset cardinality ($d = 3$)

for the naive algorithm (according to our experiments, this is basically independent of the specific algorithm variant): P-domination tests linearly increase with n, as in the case of BNL-like algorithms; moreover, the most effective rule is the cheapest Rule 1 (about 40% of cases are solved with this rule), while the effectiveness of Rule 2 is less than 0.1%, so low that the graph is unable to show it. In about 40% of cases, finally, the compared tuples are indifferent. Figure 4 (b) shows the average number of $IbP(u, v)$ values that should be computed for a tuple u: clearly, this happens whenever both Rules 1 and 3 fail, as already noted in Section 5. As the figure suggests, sorting R^p has almost no effect on reducing the number of times $IbP(u, v)$ should be computed, but only, as previously observed, on the average number of tuples to be checked in each calculation.



(a) (b)

Fig. 4. Effectiveness of P-domination rules (a) and average number of IbP calculations per tuple (b) vs. dataset cardinality ($d = 3$)

7 Conclusions

In this paper we have presented a new definition of skyline for probabilistic relations, based on an appropriate definition of P-domination, i.e., domination between tuples having a confidence/probability value. We have also proved that, unlike previous definitions, ours maintains all the nice properties that skylines have in the deterministic scenario. We have provided alternative algorithms for the efficient computation of the skyline and evaluated their performance through some preliminary experiments over synthetically generated datasets.

Although we elaborated our analysis for the case of independent tuples, the definition of P-domination can be smoothly extended to the correlated case, i.e., where possible worlds are generated through a set of generation rules. This requires opportunely adapting domination rules in Section 4.1 and algorithms in Section 5, the latter maintaining the same time complexity of the independent case.

Besides a thorough experimentation with other datasets (either with different distributions of coordinates and probabilities or real ones, if available), our current and future work includes considering alternative formulations of resolution algorithms. As a matter of fact, all our algorithms share the same 2-phase structure: we expect to attain even better performance by comparing some tuples as early as possible.

References

1. Atallah, M.J., Qi, Y.: Computing all Skyline Probabilities for Uncertain Data. In: PODS 2009. pp. 279–287. Providence, RI (Jun 2009)
2. Bartolini, I., Ciaccia, P., Patella, M.: Efficient Sort-Based Skyline Evaluation. ACM TODS 33(4), 1–45 (2008)
3. Börzsönyi, S., Kossmann, D., Stocker, K.: The Skyline Operator. In: ICDE 2001. pp. 421–430. Heidelberg, Germany (Apr 2001)
4. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with Presorting. In: ICDE 2003. Bangalore, India (Mar 2003)
5. Cormode, G., Li, F., Yi, K.: Semantics of Ranking Queries for Probabilistic Data and Expected Ranks. In: ICDE 2009. pp. 305–316. Shanghai, China (Apr 2009)
6. Fagin, R., Lotem, A., Naor, M.: Optimal Aggregation Algorithms for Middleware. In: PODS 2001. pp. 216–226. Santa Barbara, CA (May 2001)
7. Guttman, A.: R-trees: A Dynamic Index Structure for Spatial Searching. In: SIGMOD 1984. pp. 47–57. Boston, MA (Jun 1984)
8. Li, J., Saha, B., Deshpande, A.: A Unified Approach to Ranking in Probabilistic Databases. In: VLDB 2009. pp. 502–513. Lyon, France (Aug 2009)
9. Lin, X., Zhang, Y., Zhang, W., Cheema, M.A.: Stochastic Skyline Operator. In: ICDE 2009. Hannover, Germany (Apr 2011)
10. Pei, J., Jiang, B., Li, X., Yuan, Y.: Probabilistic Skylines on Uncertain Data. In: VLDB 2007. pp. 15–26. Vienna, Austria (Sep 2007)
11. Sarma, A.D., Benjelloun, O., Halevy, A.Y., Widom, J.: Working Models for Uncertain Data. In: ICDE 2006. Atlanta, GA (Apr 2006)
12. Soliman, M.A., Ilyas, I.F., Chang, K.C.C.: Top-k Query Processing in Uncertain Databases. In: ICDE 2007. pp. 896–905. Istanbul, Turkey (Apr 2007)
13. Szpilrajn, E.: Sur l'Extension de l'Ordre Partiel. Fundamenta Mathematicae 16, 386–389 (1930)
14. Zhang, W., Lin, X., Zhang, Y., Wang, W., Yu, J.X.: Probabilistic Skyline Operator over Sliding Windows. In: ICDE 2009. pp. 1060–1071. Shanghai, China (Mar 2009)
15. Zhang, X., Chomicki, J.: On the Semantics and Evaluation of Top-k Queries in Probabilistic Databases. In: DBRank 2008. pp. 556–563. Cancun, Mexico (Apr 2008)

On Probabilistic Record Linkage: New Methods Compared to the Fellegi-Sunter Approach

Diego Zardetto[1], Monica Scannapieco[2], Luca Valentino[1], and Tiziana Catarci[3]

[1] Istat, Italy, {zardetto,luvalent}@istat.it
[2] Istat and Sapienza - Univ. Roma, Italy, {scannapi}@istat.it
[3] Sapienza - Univ. Roma, Italy, catarci@dis.uniroma1.it

Abstract. Record Linkage (RL) aims at identifying pairs of records coming from different sources and representing the same real world entity. Several methods have been proposed to face RL problems and many independent software implementations of traditional methods exist. However, none of the available systems seems to be at the same time fully automated and very effective. In this paper we describe and test a new RL software that, instead, possesses both these abilities: the MAERLIN system. MAERLIN implements a novel suite of methods for RL, based on Mixture Models. Such methods allow our system to obtain accurate and reliable results without relying on domain knowledge, thus not jeopardizing automation. The system adopts a two-component Beta mixture model and finds Maximum Likelihood estimates of mixture parameters by means of an original Perturbative Fitting technique. Then, it obtains a probabilistic clustering of record pairs into Matches and Unmatches by finding optimal classification rules under arbitrary matching constraints through a purposefully designed Evolutionary Algorithm. In this paper, we provide an overview of the MAERLIN system. Then, we describe the RELAIS toolkit, which includes a state-of-the-art implementation of the traditional Fellegi-Sunter method for probabilistic RL. Finally, we provide an extensive experimental analysis comparing MAERLIN to RELAIS. Specifically, we present several experiments on challenging real-world RL instances arising from distinct application domains. The obtained results show the significant effectiveness and robustness of the methods underlying MAERLIN and also reveal interesting findings arising from the aforementioned comparative evaluation.

1 Introduction

Record Linkage (RL) deals with the problem of identifying pairs of records coming from different sources and representing the same real world entity. Integration of different data sources and improvement of the quality of single sources are only some of the real application scenarios that need to solve the RL problem. In Official Statistics, to cite just a single example, the need of performing a RL task arises whenever one tries to integrate statistical survey data with data coming from administrative archives, due to lacking or unreliable common record identifiers. Several methods have been proposed to face RL problems and many independent software implementations of traditional methods exist. However, none of the available systems seems to be at the same time fully automated and very effective. Indeed, the following pattern emerges: *(i)* RL systems incorporating domain knowledge – e.g., via clerically prepared training sets or fine-tuned parameters – can reach high effectiveness, but at the price of jeopardizing automation; *(ii)* fully automated RL systems – like approximate join algorithms, generally biased toward the goal of high efficiency – tend to perform poorly with respect to accuracy (see, e.g., [6]). In this paper we describe and test a new RL software that, instead, possesses both the aforementioned abilities: the MAERLIN system.

MAERLIN (the acronym stands for Mixture-based Automated Effective Record LINkage) implements the novel suite methods (based on Mixture Models) proposed in [10] and has been developed in R [8]. MAERLIN structures the decision phase of a RL process into two consecutive tasks. First, it estimates mixture parameters by fitting the adopted model to observed distance measures between record pairs. Then, it obtains a probabilistic clustering of record pairs into Matches and Unmatches by exploiting the fitted model. More specifically, MAERLIN uses a mixture model with component densities belonging to the Beta family and finds Maximum Likelihood estimates of mixture parameters by means of an original Perturbative Fitting technique. Moreover, it can solve the clustering problem according to both Maximum Likelihood and Minimum Cost objectives under arbitrary matching constraints (One-to-One, One-to-Many, Many-to-One and Many-to-Many restrictions can all be handled). To accomplish this task, MAERLIN searches optimal decision rules through a purposefully designed constrained Evolutionary Algorithm.

In the paper we also outline the main functionalities of the RELAIS (REcord Linkage At IStat) system. RELAIS is a toolkit including techniques for each phase of a RL process, i.e. search space reduction, string comparison, decision, matching constraints enforcement, etc. The project was launched in 2006 and is currently at its 2.1 release[4], distributed as an open source project. RELAIS includes a state-of-the-art implementation of the traditional Fellegi-Sunter method [4] for probabilistic RL, and this has been selected as the matter of comparison with MAERLIN.

We have conducted an extensive experimental analysis comparing MAERLIN to RELAIS. Specifically, we present several experiments on challenging real-world RL instances. Such instances have been deliberately selected to verify MAERLIN robustness against variations of the main characteristics of the RL problem, including: data set sizes, match rate, number and discrimination power of variables used to compute distance measures, and error rates affecting such variables. For the same reason, the data we use as experimental test bed have been retrieved from sources belonging to distinct application domains, e.g., Official Statistics surveys, bibliographic databases, and e-commerce websites. The obtained results not only show the significant effectiveness and robustness of the methods underlying MAERLIN, but also reveal interesting findings arising from the aforementioned comparative evaluation.

2 MAERLIN: Overall Picture of the System

The goal of a RL system is to identify record pairs representing the same real world entity; such pairs are named Matches according to a consolidated jargon. At a very high

[4] www.istat.it/strumenti/metodi/software/analisi_dati/relais

level of abstraction, every RL workflow can be seen as the sequence of two fundamental processes: a *comparison* process followed by a *decision* process (see, e.g., Figure 1). The comparison process takes as input the datasets to be linked (generally two, though a single dataset is actually compared against itself in deduplication applications) and performs some kind of distance (or, equivalently, similarity) measure on record pairs. The subsequent decision process takes such computed measures as input and, by applying to them a rule of some kind, eventually classifies each record pair as belonging to the class of Matches (M) or to the one of Unmatches (U). Needless to say, in almost all real-world applications, both the comparison process and the decision process actually encompass several complex subprocesses (see, e.g., Figure 3 where a scheme of MAERLIN decision process is reported).

As will be concisely discussed in Section 3, most of the original methodological contributions implemented in the MAERLIN system have been embedded in its decision engine. On the basis of these methods, MAERLIN can be defined, adopting a traditional terminology, as a system for *(i) unsupervised*, *(ii) probabilistic*, *(iii) fully automated RL*. Indeed: *(i)* MAERLIN does not require any clerically prepared training set, as opposed to supervised techniques for RL (mostly based on machine learning algorithms); *(ii)* MAERLIN assumes that the pairwise measures computed in the comparison phase obey a well defined statistical model, and exploits the statistical inference machinery to draw conclusion on the unknown class-membership of each pair; *(iii)* MAERLIN does not require any human intervention to set and/or fine-tune crucial parameters, like similarity thresholds to be exceeded by matching record pairs.

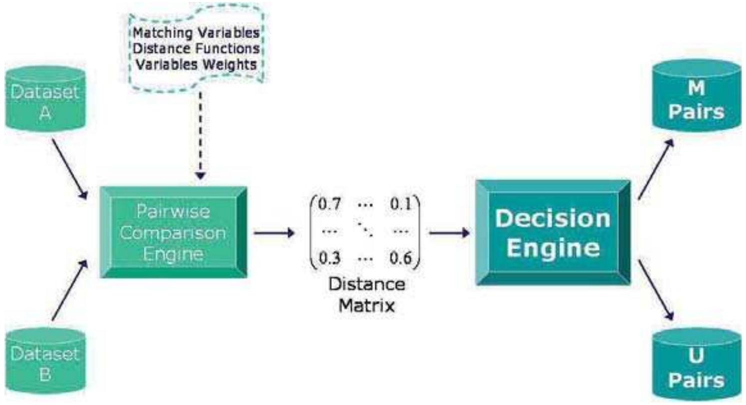


Fig. 1. MAERLIN (simplified) workflow for a general Record Linkage Process

As depicted in Figure 1, MAERLIN decision engine expects as input a matrix with values representing pairwise distances normalized in the $[0, 1]$ interval. Besides this weak functional requirement, the system does not rely on any restrictive assumption concerning the number or the type of the attributes (matching variables) to be used to compare records, or the individual distance functions to be applied to those attributes.

Therefore, MAERLIN comparison engine allows the user (see the dashed input in Figure 1): *(i)* to select an arbitrary set of matching variables among those common to the input datasets; *(ii)* to select and combine freely the distance functions to be used on individual matching variables (several string metrics are supported, e.g. Equality, Levenshtein, Jaro, JaroWinkler, etc.); *(iii)* to specify (optionally) a set of reliability weights for the matching variables. Whenever more than one matching variable is used, the system adopts a simple averaging procedure to obtain a scalar distance value: first distances measured on individual variables are transformed into standardized scores, then a weighted mean of such scores is computed, lastly the obtained scalar values are normalized in $[0,1]$.

As for large input datasets it is generally not feasible to compute pairwise distances for the whole cartesian product, MAERLIN provides a standard blocking technique to reduce the comparison space. This means that only distances corresponding to pairs with identical blocking variables are actually computed and passed to the decision engine. If the selected blocking variables are accurate (i.e., almost not affected by errors), such technique is expected to quickly filter-out pairs belonging, with high probability, to the U class.

3 MAERLIN: Overview of the Underlying Methods

MAERLIN implements a novel suite of methods for RL, based on mixture models. These are statistical models that allow to represent a probability distribution as a convex combination of other distributions (see, e.g., [7]). A thorough motivation and description of such methods has been already provided elsewhere. Due to space limitations, here we shall restrict ourselves to a very concise outline, referring the reader to [10] for further details.

Real-world data are always affected by a wide variety of unpredictable errors: this is precisely the reason why the RL problem is non-trivial. As already stated, RL methods invariably rely on distance (or similarity) measures between record pairs. Due to the stochastic nature of every real-world data generating process, such pairwise distances can be seen as (realizations of) a random variable. Thus, the intuition behind the use of mixtures models is that the observed distances arise from a superposition of two distinct probability distributions: the one stemming from the subpopulation of Matches and the other from that of Unmatches. The ultimate aim of this statistical perspective is to exploit the mixture model for classification purposes, i.e., to bring to light the hidden grouping of the pairs in the underlying M and U classes. The trick is simple: the distance is viewed as an *observable* auxiliary random variable that can be used to make inference on a *latent* interest random variable, namely the class-membership indicator of the pairs. The whole picture is founded upon the hypothesis that the probability distribution of the distance is significantly different inside the M and U classes. Luckily this is almost always the case in real application scenarios, because typically errors affect data at moderate rates. Whenever such condition holds, the shapes of the M and U distance densities are indeed very different: *(i)* Unmatches tend to be concentrated at higher distances than Matches, which furthermore generally exhibit their own distinctive peak at zero distance; *(ii)* M and U densities show only a relatively small overlap.

These qualitative features are so general that one can rightly consider them as a piece of *prior knowledge* about the underlying (unknown) *M* and *U* distance probability distributions: we refer to it as $\mathcal{PK}_1$. Besides $\mathcal{PK}_1$, another piece of prior knowledge is readily available in RL applications, namely that Matches are rare as compared to Unmatches. This property is easily understood for 1:1 RL[5], but remains true even when the data sets to be linked do contain duplicated records. We refer to this second kind of prior knowledge as $\mathcal{PK}_2$. Figure 2 exemplifies in a clear-cut way the excellent agreement between the aforementioned basic assumptions $\mathcal{PK}_1$ and $\mathcal{PK}_2$ and sample data coming from a real-world RL instance.

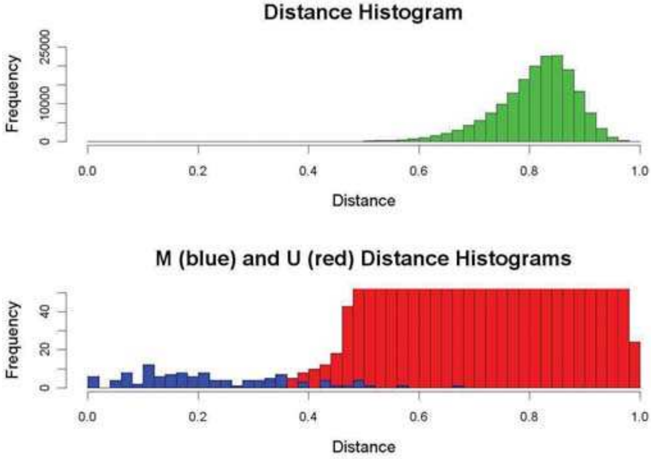


Fig. 2. Pairwise-distances coming from a real-world RL instance (Rest data sets, see Section 6). Upper panel: distance histogram of the whole unlabeled data (176,423 pairs). Lower panel: superimposed distance histograms for Match pairs (blue, dark) and Unmatch pairs (red, light); note that a 500 times y-axis zoom was needed to detect the feeble signal arising from the few Matches (112 pairs).

The decision engine coded inside MAERLIN is able to exploit $\mathcal{PK}_1$ and $\mathcal{PK}_2$ successfully when facing practical RL tasks. MAERLIN represents the probability density function of the distance as a two-component Beta[6] mixture. The system structures the decision phase of a RL process into two consecutive tasks, as schematically depicted

[5] Indeed, if data sets A and B do not contain duplicated records the Match rate, i.e. the ratio between the cardinalities of *M* and $A \times B$, cannot exceed the value $1/\max(|A|,|B|)$. This value is very small in almost all the RL problems, even when blocking techniques have been applied.

[6] The Beta is appropriate because: 1) it has bounded support; 2) it is flexible; 3) it can represent positively and negatively skewed distributions, in compliance with $\mathcal{PK}_1$; 4) it is controlled by shape parameters, so that $\mathcal{PK}_1$ can be translated easily into a set of constraints acting on the parameter space.

in Figure 3. First, it finds (constrained) Maximum-Likelihood estimates for the mixture parameters by fitting the model to the observed distance measures between pairs. Then, it obtains a probabilistic clustering of the pairs into Matches and Unmatches by exploiting the fitted model.

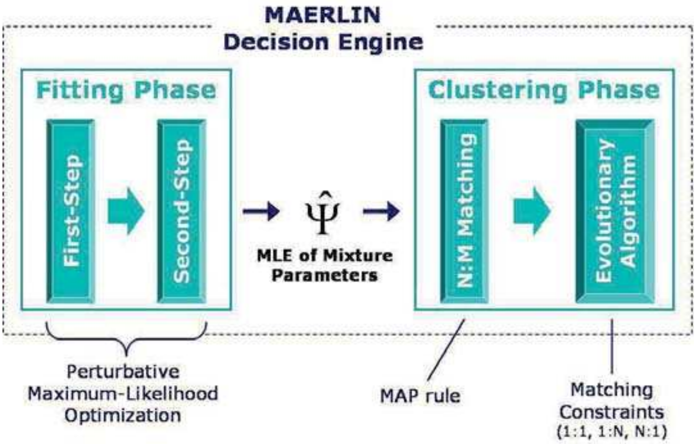


Fig. 3. A Schematic view of MAERLIN decision engine.

The fitting phase is the crucial one, as it implicitly determines the quality of the subsequent clustering results. However, it represents a very hard task; indeed, the problem of fitting a mixture model is always difficult, but it is even more severe in RL applications. This is due to the huge class-skew inherent in RL problems, where the very few (and unidentified) distance measures stemming from Matches risk to be completely overwhelmed by the bulk of those stemming from Unmatches. To overcome this difficulty MAERLIN exploits an original fitting technique inspired by Perturbation Theory (see, e.g., [1]) and designed to take advantage from both $\mathcal{PK}_1$ and $\mathcal{PK}_2$. The technique is coded as a Two-Step algorithm, with the *M* class mixing weight (which is guaranteed to be $\ll 1$, due to $\mathcal{PK}_2$) playing the role of the perturbative expansion parameter[7]. The First-Step concentrates on the *U* component mixture parameters and is specifically aimed at "factorizing" the leading contribution arising from Unmatches. The Second-Step strives to increase the Likelihood achieved in the previous step by using the remaining mixture parameters in a "smart way"; that is, *M* density parameters are tuned in such a way as to better fit the behavior of the distance distribution exactly in those regions where, thanks to $\mathcal{PK}_1$, values stemming from Matches are more likely to be found.

In the clustering phase MAERLIN searches an optimal classification rule such that each pair can be assigned, based on its observed distance value, either to the *M* or to the

[7] Stated in Perturbation Theory jargon: First-Step and Second-Step optimization algorithms are respectively in charge of solving the zeroth-order and first-order perturbative approximations of the original constrained ML optimization problem.

U class. The system can minimize either the probability of classification error (Maximum Likelihood objective) or, alternatively, the expected classification cost (Minimum Cost objective), while satisfying arbitrary matching constraints (1:1, 1:N, N:1 or N:M)[8]. If no constraints are imposed (i.e. for N:M matching), the applied classification rules depend in a quite straightforward way on posterior estimates of class membership probabilities and reflect classical Decision Theory results (see, e.g., [3]). For instance, the Maximum Likelihood objective leads to the well known "Maximum a Posteriori (MAP) rule", see Figure 3. When, on the contrary, matching constraints are imposed, MAERLIN faces directly the full-complexity constrained optimization problem by means of a purposefully designed Evolutionary Algorithm [10].

4 Overview of the RELAIS Toolkit

The RELAIS toolkit idea is based on the consideration that the RL process is application dependent. Indeed, available tools do not provide a satisfying answer to the various requirements that different applications can exhibit. A RL process typically consists of different phases; the implementation of each phase can be performed according to a specific technique or on the basis of a specific decision model. For instance, choosing which decision model to apply is not immediate: the usage of a probabilistic decision model can be more appropriate for some applications but it can be less appropriate for others, for which an empirical decision model could prove more successful (or more easy to use). Furthermore, even using the same decision model, in different application scenarios, a comparison function could fit better than others. Therefore, we claim that no RL process, deriving from the choice and combination of a specific technique for each phase, is the best for all applications. The RELAIS toolkit is composed by a collection of techniques for each phase of the RL procedure that can be dynamically combined in order to build the best RL workflow. As an example, if it is known that the datasets to compare have poor quality, it is suitable the usage of comparison functions ensuring high precision (e.g. Jaro distance); as a further example, if no specific error-rates are required by the application, it can be appropriate the usage of an empirical decision model. Some phases of the RL process can be missing: for instance the search space reduction phase makes sense only for huge data volumes, or for applications that have time constraints. The principal RELAIS functionalities (see [9] for details) are:

- Data profiling, in which a set of quality metadata (completeness, accuracy, consistency, categories, frequency distribution, entropy) are calculated starting from input data. These metadata help the user in the critical phase of choosing the best blocking or matching variables.
- Search Space Reduction which, besides the cross product of the input datasets, makes available a blocking method and the Sorted Neighbourhood method.
- Comparison Functions, currently including Equality, Numeric Comparison, 3-Grams, Dice, Jaro, JaroWinkler, Levenshtein and Soundex.

[8] These constraints arise whenever one knows in advance that both (1:1) or either (1:N or N:1) of the data sets to be matched do not contain duplicates. N:M matching actually means absence of constraints.

- Deterministic Linkage, which permits to specify complex boolean decision rules. It can be adopted, in alternative or in combination with probabilistic methods.
- Probabilistic Linkage, which implements the Fellegi-Sunter method [4].
- Optimal One-to-One Matching, in which one-to-one matching constraints between the two input data sets are enforced by solving a Linear Programming problem trough the Simplex method.
- Greedy One-to-One Matching, applicable if the optimal solution is not able to reach a result due to computational limitations. With this strategy, local choices, based on a sort of the matching probabilities, are performed to enforce the one-to-one matching constraints.

As anticipated, the RELAIS functionality that will be used for comparison with MAERLIN is the traditional Fellegi-Sunter probabilistic RL method.

5 Probabilistic RL approaches: MAERLIN versus Fellegi-Sunter

On the whole, i.e. when taking into account both the fitting phase and the clustering phase, MAERLIN approach differs very much from the Fellegi-Sunter (FS) method. Since the FS method is by far the best known approach in the probabilistic RL literature, and as in Section 6 we shall experimentally compare MAERLIN with the state-of-the-art software implementation of FS provided by RELAIS, here we quickly list some of the differences:

Modeled Variable. As opposed to MAERLIN scalar distance, FS uses a k-vector variable whose components represent agreement/disagreement outcomes obtained when comparing record pairs on k matching fields. Thus, despite FS based softwares are usually able to exploit continuous string-similarity functions, they require the user to specify agreement thresholds in order to eventually map continuous values to 0 or 1 (by the way, this also applies to RELAIS).

Statistical Model. FS assumes that the components of the comparison vector are conditionally independent; the resulting model can, thus, be thought as a two-component mixture where each component is a product of k Bernoulli distributions. Modern FS implementations, including RELAIS, generally rely on the EM algorithm [2] to estimate the $2k + 1$ parameters of the model. This is to be compared to MAERLIN two-component Beta mixture model, whose 5 parameters are estimated via our Two-Step Perturbative Fitting algorithm.

Classification Rule. FS decision model has a third class (besides M and U), namely the Possible Match class. Unlike MAERLIN, FS classification rule is neither based on Maximum Likelihood nor on Minimum Cost, but rather minimizes the expect number of Possible Matches at fixed misclassification error rates. As a matter of fact, systems based on FS, including RELAIS, are generally unable to automatically apply such rule, therefore requiring a human intervention to set the crucial classification thresholds. This is at odds with MAERLIN, whose decision engine is parameter-free and fully automated.

Matching Constraints. Few papers on FS based applications, e.g. [5], tackled the 1:1 problem by using Simplex-based algorithms (as anticipated, this is also the choice adopted by RELAIS). Such solution has to be compared to MAERLIN Clustering Evolutionary Algorithm which is able to manage every kind of matching constraints.

6 Experiments

Here we present some experiments on challenging real-world RL instances to validate our system. All experiments have been run in an ordinary PC environment, equipped with: Windows XP 64 Operating System, 4 GB RAM, 2 GHz CPU. We shall describe 5 RL instances involving very different data sources. Such instances have been delibcharately selected to verify MAERLIN robustness against variations of the main characteristics of the RL problem. These include: data set sizes, Match rate (i.e. fraction of pairs that are Matches), type of records to be matched, number and discrimination power of variables used to compute distance measures, error rates affecting such variables. For the same reason, the data we use as experimental test bed have been retrieved from sources belonging to distinct application domains, e.g., Official Statistics surveys, bibliographic databases, and e-commerce websites. Finally, we shall provide a detailed comparison between MAERLIN and RELAIS.

6.1 Experimental Setup

We first introduce the quality measures we adopt to assess the effectiveness of a RL system, then we describe the testing strategy we use for comparing MAERLIN to RELAIS, and finally we provide a concise description of the studied RL instances.

Quality measures. We choose to rely on traditional Precision (Prec) and Recall (Rec) measures. Whenever a single quality measure is needed, we select the F-measure, F=2/(Prec-1+Rec-1). Notice that the F-measure is a *conservative* quality measure, as it can reach an high value only when both Precision and Recall are high.

Comparison protocol. The main issues influencing our testing strategy are related to the intrinsic asymmetry between the automated, parameter-free nature of MAERLIN, and the need of incorporating into RELAIS some kind of domain-knowledge in the form of user-specified thresholds (recall the discussion of Section 5). To get rid of this asymmetry, we determined a comparison protocol that deliberately gives a big advantage to RELAIS, thus producing a severe test bed for our MAERLIN system. The protocol is as follows. An expert RELAIS user was allowed to perform, on every selected RL task, multiple runs with several different parameter settings (matching variables + similarity functions + agreement thresholds + classification thresholds). After each run, he was allowed to evaluate the quality scores obtained by RELAIS and to modify accordingly the parameters to be used in the next run, thus progressively tuning them. At the end of the cycle, only the best parameter configuration was retained, and the corresponding Precision, Recall, and F-measure results accepted and reported as RELAIS benchmark scores. On the contrary, no tuning was allowed when running MAERLIN, not even in the comparison phase: MAERLIN was constrained to use the same matching variables

Table 1. Relevant Features of RL instances

RL Instance	Data Origin	Matching Variables	Data/Error Nature	Pairs ($n_A \times n_B$)	Number of Matches	Match Rate
Rest	Riddle	name, address, city, type	Real/Real	176,423 (331 × 533)	112	$6.3 \cdot 10^{-4}$
Cens	SecondString	surname, name, midinit, number, street	Artificial/Artificial	176,008 (392 × 449)	327	$1.9 \cdot 10^{-3}$
Bib	Leipzig Univ	title, author, year	Real/Real	6,001,104 (2,294 × 2,616)	2,224	$3.7 \cdot 10^{-4}$
E-comm	Leipzig Univ	name, price, description	Real/Real	1,157,776 (1,076 × 1,076)	1,076	$9.3 \cdot 10^{-4}$
PES	Istat	surname, name, birth.dd, birth.mm, birth.yyyy, sex	Real/Real	32,876,434,096 (180,133 × 182,512)	172,621	$5.3 \cdot 10^{-6}$

and distance functions corresponding to the best parameter configuration found for RELAIS, without any possibility of specifying reliability weights for the variables. For the sake of absolute clarity, we stress here that *no way* can the described protocol surreptitiously determine an advantage for MAERLIN, indeed: *(i)* RELAIS best choices for matching variables and distance functions are generally found to be *suboptimal* for MAERLIN (unsurprisingly as MAERLIN and FS methods differ both in the modeled variable and in the adopted statistical model); *(ii)* the most influential parameters in the RELAIS tuning/optimisation cycle, that is the agreement thresholds and (mainly) the classification thresholds, are simply *absent* from MAERLIN framework.

RL instances. Table 1 reports some basic information concerning the RL instances we selected, to which we refer as **Rest**, **Cens**, **Bib**, **E-comm** and **PES**. All such instances imply 1:1 constraints and, with the only exception of **Cens**, involve real-world data. These problems are all hard, as indicated by (though not exclusively due to) their very low Match rates. The **Rest** instance involves restaurant data affected by real-world errors, coming from Zagat's and Fodor's guidebooks (available at the RIDDLE[9] repository). For this instance, four matching variables were used and the Levenshtein distance was applied to all. The **Cens** datasets, originally provided by W. Winkler, contain synthetic census-like records (available with the SECONDSTRING package[10]). The corresponding RL instance relies on five matching variables and applies the Levenshtein distance. The **Bib** instance deals with bibliographic data covering the same sets of computer science conferences and journals, retrieved from the DBLP and ACM digital libraries (available at the website of the Leipzig University[11]). Three matching variables were used for comparison, again by applying uniformly the Levenshtein distance. The **E-comm** datasets (again retrieved from the Leipzig University website) contain records describing products of the same category available for sale at the Abt.com and Buy.com online shops. Due to the unstructured nature of the original web sources, the quality of **E-comm** data is very low (heterogeneous representations of product names and descriptions, misspellings, missing product prices, extraction errors). RELAIS faced this

[9] www.cs.utexas.edu/users/ml/riddle/index.html

[10] www.cs.utexas.edu/users/ml/riddle/data/secondstring.tar.gz

[11] dbs.uni-leipzig.de/en/research/projects/object_matching

Table 2. Precision, Recall and F-measure Results: MAERLIN vs. RELAIS

RL Instance	$\mathrm{Prec}_{\mathrm{MAERLIN}}$	$\mathrm{Prec}_{\mathrm{RELAIS}}$	$\Delta_{\mathrm{Prec}}(\%)$	$\mathrm{Rec}_{\mathrm{MAERLIN}}$	$\mathrm{Rec}_{\mathrm{RELAIS}}$	$\Delta_{\mathrm{Rec}}(\%)$	$\mathrm{F}_{\mathrm{MAERLIN}}$	$\mathrm{F}_{\mathrm{RELAIS}}$	$\Delta_{\mathrm{F}}(\%)$
Rest	0.925	0.839	+10.2%	0.875	0.884	−1.0%	0.899	0.861	+4.4%
Cens	0.994	0.952	+4.4%	0.988	0.979	+0.9%	0.991	0.965	+2.6%
Bib	0.987	0.986	+0.1%	0.970	0.949	+2.2%	0.978	0.967	+1.2%
E-Comm	0.968	0.288	+236.3%	0.455	0.642	−29.0%	0.619	0.398	+55.8%
PES	0.999	0.998	+0.1%	0.992	0.962	+3.1%	0.996	0.980	+1.6%
Average* Performance	0.976	0.944	+3.4%	0.956	0.944	+1.3%	0.966	0.943	+2.4%

RL task by using 3-grams on `name` and `description` and a numeric comparison function on `price`. As the 3-grams distance is not already supported in MAERLIN, our system exploited instead the cosine distance between TF-IDF weights vectors. The **PES** instance involves data coming from the Post Enumeration Survey (PES) carried out by the Italian National Institute of Statistics to estimate the coverage rate of the 2001 population Census. Therefore, this RL task required the matching of two lists of people, the first collected by the Census and the second by the PES. Six matching variables were used and again the Levenshtein distance was chosen. **PES** comparison-space was so huge (about 33 *billions* of pairwise distances, see Table 1) that both systems had to perform a preliminary blocking step. The enumeration area code was selected as blocking variable. From a computational complexity point of view, the net result was to transform the original, global RL task (which was not affordable) into a sequence of smaller, independent RL subtasks, one for each block. The overall number of processed blocks was 1,098. Correspondingly, the size of the comparison-space decreased from about 33 *billions* to about 86 *millions* pairs.

6.2 Results

The results of our experiments are collectively shown in Table 2, with $\Delta_{\{\mathrm{Prec,Rec,F}\}}$ expressing the percent performance gain (or loss) of MAERLIN versus RELAIS with respect to a given quality measure. It has to be stressed that, since for **E-comm** we could not employ the same distance functions when running MAERLIN and RELAIS, and as this is a possible source of bias when comparing the systems, we excluded that instance from the computation of average performances (whence the superscript *).

A first look to the average Precision (0.976), Recall (0.956), and F-measure (0.966) scores achieved by MAERLIN immediately reveals the remarkable effectiveness of our system. Moreover, MAERLIN exhibits also a very good robustness, with F-measures scores falling significantly below 0.9 only for the **E-comm** instance. Turning the attention to the comparative evaluation, we observe that: *(i)* MAERLIN always outperforms RELAIS with respect to Precision; *(ii)* MAERLIN outperforms RELAIS with respect to Recall in 3 cases out of 5; *(iii)* MAERLIN F-measure scores are always higher than the ones achieved by RELAIS. These results are clearly summarized in the average performance values (last row of Table 2), with MAERLIN showing some effectiveness gain as compared to RELAIS for all the adopted quality measures. The small amount of such relative improvements (+3.4% Precision, +1.3% Recall, +2.4% F-measure) should not be misleading: indeed, our comparison protocol was deliberately biased in favor of

RELAIS, which in turn is a very good RL system in itself. The **PES** instance demonstrates the practical feasibility of MAERLIN approach when dealing with very large amounts of data. Indeed, besides excellent quality scores, our system also exhibits a very satisfactory behavior with respect to computational efficiency: the overall execution time for processing about 86 millions of pairs, partitioned into 1,098 blocks, was 293 minutes (i.e. less than 5 hours) corresponding to an average processing time of about $2 \cdot 10^{-4}$ seconds per pair. Even if MAERLIN superseded once more RELAIS (+55.8% F-measure), the **E-comm** task turned out to be challenging for both systems. This is not surprising, given the unstructured nature of the original web sources and the resulting extremely poor quality of the variables used for comparison, whose values are actually bulky and heterogeneous agglomerates of words. Apparently, the task of linking data crawled from the web needs further improvements, perhaps to figure out some new and more sophisticated distance function.

7 Conclusions

In this paper, we presented the MAERLIN RL system. Several original contributions enable MAERLIN to be at the same time effective and fully automated. We validated our system by testing its Precision, Recall and F-measure scores on challenging real-world problems arising from very different application domains. MAERLIN obtained excellent results, outperforming constantly the traditional Fellegi-Sunter approach as implemented by the RELAIS system. This proved as well the remarkable robustness of the methods implemented by the MAERLIN system. The system is currently in the alpha testing phase and is planned to be released as a standard R package on CRAN (the Comprehensive R Archive Network)[12]. Moreover, since RELAIS, as a toolkit, gives the possibility of integrating new techniques, we also plan to include MAERLIN into RELAIS as an alternative decision method for probabilistic RL.

References

1. Bender, C., Orszag, S.: Advanced Mathematical Methods for Scientists and Engineers: Asymptotic methods and perturbation theory. Springer, N. Y. (1999)
2. Dempster, A., Laird, N., Rubin, D.: Maximum-likelihood from incomplete data via the em algorithm. JRSS, SERIES B 39(1) (1977)
3. Duda, R., Hart, P., Stork, D.: Pattern Classification. John Wiley & Sons (2000)
4. Fellegi, I., Sunter, A.: A theory for record linkage. JASA 64 (1969)
5. Jaro, M.: Advances in record-linkage methodology as applied to matching the 1985 census of tampa, florida. JASA 84 (1989)
6. Kopcke, H., Thor, A., Rahm, E.: Evaluation of entity resolution approaches on real-world match problems. In: VLDB (2010)
7. McLachlan, G., Peel, D.: Finite Mixture Models. John Wiley & Sons (2000)
8. R-Development-Core-Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing (2011)
9. RELAIS-Development-Team: RELAIS Manual Version 2.1. Istat (2010)
10. Zardetto, D., Scannapieco, M., Catarci, T.: Effective automated object matching. In: ICDE (2010)

[12] cran.r-project.org

Answering Queries in a Relaxed Way

Davide Martinenghi[1] and Riccardo Torlone[2]

[1]Dip. di Elettronica e Informazione
Politecnico di Milano, Italy
martinen@elet.polimi.it

[2]Dip. di Informatica e Automazione
Università Roma Tre, Italy
torlone@dia.uniroma3.it

Abstract. Traditional information search in which queries are posed against a known and rigid schema over a structured database is shifting towards a Web scenario in which schemas are vague or absent. In this framework, query answering needs to be relaxed with the goal of matching user requests with accessible data. In this paper, we propose a logical model and an abstract query language as a foundation for querying data sets with vague schemas. The model is a natural extension of the relational model in which data domains are organized in taxonomies, simple classifications of values arranged in a hierarchical structure. The query language is a conservative extension of relational algebra where special operators allow the specification of relaxed queries over vaguely structured information.

1 Introduction

There are today many application scenarios in which user queries do not match the structure and the content of data repositories, given the nature of the application domain or because the schema is not available. This happens for instance in location-based search (find an opera concert in Paris next summer), multifaceted product search (find a cheap blu-ray player with an adequate user rating), and multi-domain search (find a DB conference for which a cheap travel solution exists). In these situations, the query is usually relaxed to accommodate user's needs, and query answering relies on finding the best match between the user request and the available data.

In spite of this trend towards "schema-agnostic" applications, the support of current database technology for query relaxation is quite limited. The only examples are in the context of semi-structured information, in which schemas and values are varied and/or missing [1]. Conversely, the above mentioned applications can benefit from applying traditional relational database technology enhanced with a support for the management of query relaxation.

To this aim, in this paper we present a logical data model and an abstract query language supporting query relaxation over relational data. Our approach relies on the availability of taxonomies, that is, simple ontologies in which terms used in schemas and data are arranged in a hierarchical structure according to a generalization-specialization relationship. The data model is a natural extension of the relational model in which data domains are organized in hierarchies, according to different levels of detail: this guarantees a smooth implementation of the approach with current database technology. In this model data and metadata can be expressed at different levels of detail. This is made possible by a partial order relationship defined both at the schema and at the instance level.

The query language is called Taxonomy-based Relational Algebra (TRA) and is a conservative extension of relational algebra. TRA includes two special operators that extend the capabilities of standard selection and join by relating values occurring in tuples with values in the query using the taxonomy. In this way, we can formulate relaxed queries that refer to attributes and terms different from those occurring in the actual database.

The rest of the paper is organized as follows. In Section 2 we present the data model. TRA, the query language for this model, is illustrated in Section 3. In Section 4 we compare our approach with related works and finally, in Section 5, we draw some conclusions and sketch future works.

2 A Data Model with Taxonomies

2.1 Hierarchical domains and t-relations

The basic construct of our model is the *hierarchical domain* or simply the *h-domain*, a collection of values arranged in a containment hierarchy. Each h-domain is described by means of a set of *levels* representing the domain of interest at different degrees of granularity.

Definition 1 (H-domain). *An* h-domain *h is composed of:*

- *a finite set $L = \{l_1, \ldots, l_k\}$ of* levels*, each of which is associated with a set of values called the* members *of the level and denoted by $M(l)$;*
- *a partial order $\leq_L$ on L having a bottom element, denoted by $\bot_L$, and a top element, denoted by $\top_L$, such that: (i) $M(\bot_L)$ contains a set of* ground *members whereas all the other levels contain members that represent groups of ground members, and (ii) $M(\top_L)$ contains only a special member $m_\top$ that represents all the ground members;*
- *a family* CM *of* containment mappings $\textsc{cmap}_{l_1}^{l_2} : M(l_1) \rightarrow M(l_2)$ *for each pair of levels $l_1 \leq_L l_2$ satisfying the following* consistency conditions*:*

Example 1. The h-domain *time* has a bottom level whose (ground) members are timestamps and a top level whose only member, `anytime`, represents all possible timestamps. Other levels can be day, week, month, quarter, season and year, where day $\leq_L$ month $\leq_L$ quarter $\leq_L$ year and day $\leq_L$ season. A possible member of the Day level is `23/07/2010`, which is mapped by the containment mappings to the member `07/2010` of the level month and to the member `Summer` of the level season.

As should be clear from Definition 1, in this paper we consider a general notion of taxonomy in which, whenever $l_1 \leq_L l_2$ for two levels in an h-domain, then the set of ground members for l_1 is contained in the set of ground members for l_2.

Actually, a partial order $\leq_M$ can also be defined on the members M of an h-domain h: it is induced by the containment mappings as follows.

Definition 2 (Poset on members). *Let h be an h-domain and m_1 and m_2 be members of levels l_1 and l_2 of h, respectively. We have that $m_1 \leq_M m_2$ if: (i) $l_1 \leq_L l_2$ and (ii) $\text{CMAP}_{l_1}^{l_2}(m_1) = m_2$.*

Example 2. Consider the h-domain of Example 1. Given the members $m_1 =$ 29/06/2010 and $m_2 =$ 23/08/2010 of the level day, $m_3 =$ 06/2010 and $m_4 =$ 08/2010 of the level month, $m_5 =$ 2Q 2010 and $m_6 =$ 3Q 2010 of the level quarter, $m_7 =$ 2010 of the level year, and $m_8 =$ Summer of the level season, we have: $m_1 \leq_M m_3 \leq_M m_5 \leq_M m_7$, $m_2 \leq_M m_4 \leq_M m_6 \leq_M m_7$, and $m_1 \leq_M m_8$ and $m_2 \leq_M m_8$.

The main construct of the data model is the t-relation, a natural extension of a relational table built over taxonomies of values.

Definition 3 (T-relation). *Let H be a set of h-domains. We denote by $S = \{A_1 : l_1, \ldots, A_k : l_k\}$ a* t-schema *(schema over taxonomies), where each A_i is a* distinct *attribute name and each l_i is a level of some h-domain in H. A* t-tuple *t over a t-schema $S = \{A_1 : l_1, \ldots, A_k : l_k\}$ is a function mapping each attribute A_i to a member of l_i. A* t-relation *r over S is a set of t-tuples over S.*

Given a t-tuple t over a t-schema S and an attribute A_i occurring in S on level l_i, we will denote by $t[A_i : l_i]$ the member of level l_i associated with t on A_i. For a subset S' of S, we will denote by $t[S']$ the restriction of t to S'. Finally, for simplicity, we will often not make any distinction in the following between the name of an attribute of a t-relation and the name of the corresponding h-domain.

Example 3. As an example, a t-schema over the h-domains *time*, *location* and *weather conditions* can be the following: $S = \{Time : \text{day}, Location : \text{city}, Weather : \text{brief}\}$. A possible t-relation over this schema is the following:

$r_1 =$

Time: day	Location: city	Weather: brief	
11/05/2010	Rome	Sunny	$t_{1,1}$
24/04/2009	Milan	Cloudy	$t_{1,2}$
24/07/2010	New York	Showers	$t_{1,3}$

Then we have: $t_{1,1}[\text{Location:city}] =$ Rome.

A partial order relation on both t-schemas and t-relations can be also defined in a natural way.

Definition 4 (Poset on t-schemas). *Let S_1 and S_2 be t-schemas over a set of h-domains H_1 and H_2 respectively. We have that $S_1 \leq_S S_2$ if: (i) $H_1 \subseteq H_2$, and (ii) for each $A_i : l_i \in S_1$ there is an element $A_i : l_j \in S_2$ such that $l_i \leq_L l_j$.*

Definition 5 (Poset on t-tuples). *Let t_1 and t_2 be t-tuples over S_1 and S_2 respectively. We have that $t_1 \leq_t t_2$ if: (i) $S_1 \leq_S S_2$, and (ii) for each $A_i : l_i \in S_1$ there is an element $A_i : l_j \in S_2$ such that $t_1[A_i : l_i] \leq_M t_2[A_i : l_j]$.*

Note that, in these definitions, we assume that levels of the same h-domain occur in different t-schemas with the same attribute name: this strongly simplifies the notation that follows without loss of expressibility. Basically, it suffices to use as attribute name the role played by the h-domain in the application scenario modeled by the t-schema.

Example 4. Consider the following t-relations:

$S_1 =$	Title:cultural-event	Time:day	Location:theater	
$r_1 =$	Romeo & Juliet	13/04/2011	La Scala	$t_{1,1}$
	Carmen	24/05/2011	Opéra Garnier	$t_{1,2}$
	Requiem	28/03/2011	La Scala	$t_{1,3}$
	La bohème	09/01/2011	Opéra Garnier	$t_{1,4}$

$S_2 =$	Title:event	Time:quarter	Location:city	Ticket:website	
$r_2 =$	Concert	1Q 2011	Milan	tkts.com	$t_{2,1}$
	Ballet	2Q 2011	Milan	tkts.com	$t_{2,2}$
	Sport	3Q 2011	Rome	allsport.it	$t_{2,3}$
	Opera	2Q 2011	Paris	billet.fr	$t_{2,4}$

Then, it is easy to see that: (i) $S_1 \leq_S S_2$, and (ii) $t_{1,1} \leq_t t_{2,2}$, $t_{1,2} \leq_t t_{2,4}$, $t_{1,3} \leq_t t_{2,1}$, and $t_{1,4} \leq_t t_{2,4}$.

In the following, for the sake of simplicity, we will often make no distinction between the name of an attribute and the corresponding level.

3 Querying with Taxonomies

In this section we present TRA (Taxonomy-based Relational Algebra) an extension of relational algebra over t-relations. This language provides insights on the way in which data can be manipulated taking advantage of available taxonomies over those data. Moreover, for its procedural nature, it can be profitably used to specify query optimization. The goal is to provide a solid foundation to querying databases in a relaxed way using taxonomies.

Similarly to what happens with the standard relational algebra, the operators of TRA are closed, that is, they apply to t-relations and produce a t-relation as result. In this way, the various operators can be composed to form the *t-expressions* of the language.

TRA is a conservative extension of basic relational algebra (RA) and so it includes its standard operators: selection (σ), projection (π), and natural join ($\bowtie$). It also includes some variants of these operators that are obtained by combining them with the following two new operators.

Definition 6 (Upward extension). *Let r be a t-relation over S, A be an attribute in S over a level l, and l' be a level such that $l \leq_L l'$. The* upward extension *of r to l', denoted by $\hat{\varepsilon}_{A:l}^{A:l'}(r)$, is the t-relation over $S \cup \{A : l'\}$ defined as follows: $\hat{\varepsilon}_{A:l}^{A:l'}(r) = \{t \mid \exists t' \in r : t[S] = t', t[A : l'] = \textsc{cmap}_l^{l'}(t'[A : l])\}$*

Definition 7 (Downward extension). *Let r be a t-relation over S, A be an attribute in S over a level l, and l' be a level such that $l' \leq_L l$. The* downward extension *of r to l', denoted by $\check{\varepsilon}^{A:l}_{A:l'}(r)$, is the t-relation over $S \cup \{A : l'\}$ defined as follows: $\check{\varepsilon}^{A:l}_{A:l'}(r) = \{t \mid \exists t' \in r : t[S] = t', t'[A : l] = \mathrm{CMAP}^{l}_{l'}(t[A : l'])\}$*

For simplicity, in the following we will often simply write $\hat{\varepsilon}^{l'}_{l}$ or $\check{\varepsilon}^{l'}_{l}$, when there is no ambiguity on the attribute name associated with the corresponding levels.

Example 5. Consider the t-relations r_1 and r_2 from Example 4. The result of $\hat{\varepsilon}^{\mathsf{city}}_{\mathsf{theater}}(r_1)$ is the following t-relation.

$S_3 =$	Title:cultural-event	Author:artist	Time:day	Location:theater	Location:city	
$r_3 =$	Romeo & Juliet	Prokofiev	13/04/2011	La Scala	Milan	$t_{3,1}$
	Carmen	Bizet	24/05/2011	Opéra Garnier	Paris	$t_{3,2}$
	Requiem	Verdi	28/03/2011	La Scala	Milan	$t_{3,3}$
	La bohème	Puccini	09/01/2011	Opéra Garnier	Paris	$t_{3,4}$

The result of $\check{\varepsilon}^{\mathsf{quarter}}_{\mathsf{month}}(r_2)$ is the following t-relation.

$S_4 =$	Title:event	Time:quarter	Location:city	Time:month	
$r_4 =$	Concert	1Q 2011	Milan	Jan 2011	$t_{4,1}$
	Concert	1Q 2011	Milan	Feb 2011	$t_{4,2}$
	Concert	1Q 2011	Milan	Mar 2011	$t_{4,3}$
	Sport	3Q 2011	Rome	Jul 2011	$t_{4,4}$
	Sport	3Q 2011	Rome	Aug 2011	$t_{4,5}$
	Sport	3Q 2011	Rome	Sep 2011	$t_{4,6}$
	...	...	...	...	

The main rationale behind the introduction of the upward extension is the need to relax a query with respect to the level of detail of the queried information. For example, one might want to find events taking place in a given country, even though the events might be stored with a finer granularity (e.g., city). Similarly, the downward extension allows the relaxation of the answer with respect to the level of detail of the query. For instance, a query about products available in a given day may return the products available in that day's month. Both kinds of extensions meet needs that arise naturally in several application domains.

For this purpose, we introduce two new operators for the selection that leverage the available taxonomies; they can reference an h-domain that is more general or more specific than those referenced by the underlying data.

Definition 8 (Upward selection). *Let r be a t-relation over S, A be an attribute in S defined over l, m be a member of l' with $l \leq_L l'$, and $\theta \in \{=, <, >, \leq, \geq, \neq\}$: the* upward selection *of r with respect to $A\,\theta\,m$ on level l, denoted by $\hat{\sigma}_{A:l\,\theta\,m}(r)$, is the t-relation over S defined as follows:*

$$\hat{\sigma}_{A:l\,\theta\,m}(r) = \{t \in r \mid \mathrm{CMAP}^{l'}_{l}(t[A : l])\,\theta\,m\}$$

Definition 9 (Downward selection). *Let r be a t-relation over S, A be an attribute in S defined over l, m be a member of l' with $l' \leq_L l$, and $\theta \in \{=, <, >, \leq, \geq, \neq\}$: the* downward selection *of r with respect to $A\,\theta\,m$ on level l, denoted by $\check{\sigma}_{A:l\,\theta\,m}(r)$, is the t-relation over S defined as follows:*

$$\check{\sigma}_{A:l\,\theta\,m}(r) = \{t \in r \mid \mathrm{CMAP}^{l}_{l'}(m)\,\theta\,t[A : l]\}$$

In the following, we will often simply write $\hat{\sigma}_{A\,\theta\,m}$ and $\check{\sigma}_{A\,\theta\,m}$, without explicitly indicating the name of the level, when this is unambiguously determined by the corresponding attribute.

Example 6. Consider again the t-relations r_1 and r_2 from Example 4. We have that: $\hat{\sigma}_{\mathsf{City}=\mathtt{Milan}}(r_1) = \{t_{1,1}, t_{1,3}\}$ and $\check{\sigma}_{\mathsf{Day}=\mathtt{13/03/2011}}(r_2) = \{t_{2,1}\}$.

It can be easily seen that these operators can be obtained by composing the upward or downward extension, the (standard) selection, and the projection operators, as shown in (1) and (2) below.

$$\hat{\sigma}_{A:l\,\theta\,m}(r) = \pi_S(\sigma_{A:l'\,\theta\,m}(\hat{\varepsilon}^{A:l'}_{A:l}(r))) \tag{1}$$

$$\check{\sigma}_{A:l\,\theta\,m}(r) = \pi_S(\sigma_{A:l'\,\theta\,m}(\check{\varepsilon}^{A:l}_{A:l'}(r))) \tag{2}$$

Finally, we introduce two new join operators. Their main purpose is to combine information stored at different levels of granularity.

Definition 10 (Upward join). *Let r_1 and r_2 be two t-relations over S_1 and S_2 respectively, and let S be an upper bound of a subset $\bar{S}_1$ of S_1 and a subset $\bar{S}_2$ of S_2. The* upward join *of r_1 and r_2 with respect to S on $\bar{S}_1$ and $\bar{S}_2$, denoted by $r_1 \hat{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2$, is the t-relation over $S_1 \cup S_2$ defined as follows:*

$$r_1 \hat{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2 = \{t \mid \exists t_1 \in r_1, \exists t_2 \in r_2, \exists t' \, over \, S : t_1[\bar{S}_1] \leq_t t', \\ t_2[\bar{S}_2] \leq_t t', t[S_1] = t_1, t[S_2] = t_2\}$$

Definition 11 (Downward join). *Let r_1 and r_2 be two t-relations over S_1 and S_2 respectively, and let S be a lower bound of a subset $\bar{S}_1$ of S_1 and a subset $\bar{S}_2$ of S_2. The* downward join *of r_1 and r_2 with respect to S on $\bar{S}_1$ and $\bar{S}_2$, denoted by $r_1 \check{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2$, is the t-relation over $S_1 \cup S_2$ defined as follows:*

$$r_1 \check{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2 = \{t \mid \exists t_1 \in r_1, \exists t_2 \in r_2, \exists t' \, over \, S : t' \leq_t t_1[\bar{S}_1], \\ t' \leq_t t_2[\bar{S}_2], t[S_1] = t_1, t[S_2] = t_2\}$$

Example 7. Consider the t-relation r_1 from Example 4 and the following t-relation.

$S_5 =$	Company:airline-company	Location:airport	
$r_5 =$	Alitalia	Linate	$t_{5,1}$
	Air France	Roissy	$t_{5,2}$

The result of $r_1 \hat{\bowtie}_{\mathsf{city}} r_5$ is the following t-relation:

$S_6 =$	Event:cultural-event	Author:artist	Time:day	Location:theater	Company:airline-company	Location:airport	
$r_6 =$	Romeo & Juliet	Prokofiev	24/04/2011	La Scala	Alitalia	Linate	$t_{6,1}$
	Carmen	Bizet	24/05/2011	Opéra Garnier	Air France	Roissy	$t_{6,2}$
	Requiem	Verdi	24/03/2011	La Scala	Alitalia	Linate	$t_{6,3}$
	La bohème	Puccini	09/01/2011	Opéra Garnier	Air France	Roissy	$t_{6,4}$

Now, consider the following t-relations.

$S_7 =$	Loc:theater	Time:year	Price:money	
$r_7 =$	La Scala	2011	150	$t_{7,1}$

$S_8 =$	Loc:theater	Time:month	Discount:perc.	
$r_8 =$	La Scala	03/2011	10%	$t_{8,1}$
	La Scala	06/2011	20%	$t_{8,2}$

The result of $r_7 \check{\bowtie}_{\mathsf{theater,day}} r_8$ is the following t-relation:

$S_9 =$	Loc:theater	Time:year	Price:money	Time:month	Discount:perc.	
$r_9 =$	La Scala	2011	150	03/2011	10%	$t_{9,1}$
	La Scala	2011	150	06/2011	20%	$t_{9,2}$

Also in this case, both the upward join and the downward join can be obtained by combining the upward extension or the downward extension, and the (standard) join. Equation (3) below shows this for the upward join, where $S = \{A^1 : l^1, \ldots, A^n : l^n\}$, $S_i \supseteq \bar{S}_i \supseteq \{A^1 : l^1_i, \ldots, A^n : l^n_i\}$ for $i = 1, 2$, and P is a predicate requiring pairwise equality in both sides of the join for all fields added by the extensions.

$$r_1 \hat{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2 = \pi_{S_1 S_2}(\hat{\varepsilon}^{A^1:l^1}_{A^1:l^1_1} \cdots \hat{\varepsilon}^{A^n:l^n}_{A^n:l^n_1}(r_1) \bowtie_P \hat{\varepsilon}^{A^1:l^1}_{A^1:l^1_2} \cdots \hat{\varepsilon}^{A^n:l^n}_{A^n:l^n_2}(r_2)) \qquad (3)$$

Equation (4) below shows this for the downward join, where $S \supseteq \{A^1 : l^1, \ldots, A^n : l^n\}$, $S_i \supseteq \bar{S}_i \supseteq \{A^1 : l^1_i, \ldots, A^n : l^n_i\}$ for $i = 1, 2$, and P is as above.

$$r_1 \check{\bowtie}_{S:\bar{S}_1,\bar{S}_2} r_2 = \pi_{S_1 S_2}(\check{\varepsilon}^{A^1:l^1_1}_{A^1:l^1} \cdots \check{\varepsilon}^{A^n:l^n_1}_{A^n:l^n}(r_1) \bowtie_P \check{\varepsilon}^{A^1:l^1_2}_{A^1:l^1} \cdots \check{\varepsilon}^{A^n:l^n_2}_{A^n:l^n}(r_2)) \qquad (4)$$

As in the standard relational algebra, it is possible to build complex expressions combining several TRA operators thanks to the fact that TRA is closed, i.e., the result of every application of an operator is a t-relation.

In [8] we have identified a number of equivalence rules for the operators of TRA that provide a formal foundation for the algebraic optimization of relaxed queries in our model. For instance, it is possible to prove the following equivalence indicating that a upward selection can be "pushed" through an upward join on the side that involves the attribute-level pair used in the selection.

$$\hat{\sigma}_{A:l\,\theta\,m}(r_1 \hat{\bowtie}_{C_{up}:C_1,C_2} r_2) = (\hat{\sigma}_{A:l\,\theta\,m} r_1) \hat{\bowtie}_{C_{up}:C_1,C_2} r_2 \qquad (5)$$

4 Related work

The approach proposed in this paper is focused on the relaxation of queries to a less restricted form with the goal of accommodating user's needs. This problem has been investigated in several research areas under different perspectives.

In the database area, query relaxation has been addressed in the context of XML and semi-structured databases, with the goal of combining database style querying and keyword search [1] and for querying databases with natural language interfaces [7]. Malleable schemas [4, 9] deals with vagueness and ambiguity in database querying by incorporating imprecise and overlapping definitions of data structures. An alternative formal framework relies on multi-structural databases [5], where data objects are segmented according to multiple distinct criteria in a lattice structure and queries are formulated in this structure. The majority of these approaches rely on non-traditional data models, whereas we refer on a simple extension of the relational model. Moreover, none of them consider relaxation via taxonomies, which is our concern.

Query relaxation is also used in location-based search [3], but in the typical IR scenario in which a query consists of a set of terms and query evaluation is focused in the ranked retrieval of documents. This is also the case of the approach in [2], where the authors consider the problem of fuzzy matching queries to items. Actually, in the information retrieval area, which is however clearly different from ours, document taxonomies have been already used in, e.g., [6], where the authors focus on classifying documents into taxonomy nodes and developing the scoring function to make the matching work well in practice.

5 Conclusion

In this paper, we have presented a logical model and an algebraic language as a foundation for querying databases using taxonomies. In order to facilitate the implementation of the approach with current technology, they rely on a natural extension of the relational model. The hierarchical organization of data allows the specification of queries that refer to values at varying levels of details, possibly different from those available in the underlying database. We have also studied the interaction between the various operators of the query language as a formal foundation for the optimization of taxonomy-based queries.

Several interesting directions of research can be pursued within the framework presented in this paper. In particular, we plan to develop methods for the automatic identification of the level in which two heterogeneous t-relations can be joined for integration purposes. Also, we are currently studying the impact of our model on the complexity of query answering. On the practical side, we plan to study how the presented approach can be implemented, in particular whether materialization of taxonomies is convenient.

References

1. B. Fazzinga, S. Flesca, and F. Furfaro. XPath Query Relaxation Through Rewriting Rules. In *IEEE TKDE*, vol. 99, 2010.
2. A. Z. Broder, M. Fontoura, V. Josifovski, and L. Riedel. A semantic approach to contextual advertising. In *Proc. of SIGIR*, pages 559-566, 2007.
3. Y.-Y. Chen, T. Suel, and A. Markowetz. Efficient query processing in geographic web search engines. In *Proc. of SIGMOD*, pag. 277–288, 2006.
4. X. Dong and A. Y. Halevy. Malleable schemas: A preliminary report. In *Proc. of WebDB*, pages 139–144, 2005.
5. R. Fagin, R. V. Guha, R. Kumar, J. Novak, D. Sivakumar, and A. Tomkins. Multi-structural databases. In *Proc. of PODS*, pag. 184–195, 2005.
6. M. Fontoura, V. Josifovski, R. Kumar, C. Olston, A. Tomkins, and S. Vassilvitskii. Relaxation in text search using taxonomies. *Proc. of VLDB*, 1(1): 672–683, 2008.
7. Y. Li, H. Yang, and H. V. Jagadish. NaLIX: A generic natural language search environment for XML data. *TODS* 32(4): art. 30, 2007.
8. D. Martinenghi and R. Torlone. Querying Databases with Taxonomies. In *Proc. of ER*, pag. 377–390, 2010.
9. X. Zhou, J. Gaugaz, W. Balke, and W. Nejdl. Query relaxation using malleable schemas. In *Proc. of SIGMOD*, pag. 545–556, 2007.

A Fast and Accurate Algorithm for Hierarchical Clustering on Euclidean Distances (Extended Abstract)

Elio Masciari[1], Giuseppe M. Mazzeo[1], and Carlo Zaniolo[2]

[1] ICAR-CNR – Institute of Italian National Research Council
{masciari,mazzeo}@icar.cnr.it
[2] UCLA – University of California Los Angeles
zaniolo@cs.ucla.edu

Abstract. A simple hierarchical clustering algorithm is proposed that is faster and more accurate than existing algorithms, including k-means and its recently proposed refinements. The algorithm consists of a divisive phase and an agglomerative phase; during these two phases, the samples are repartitioned using a least quadratic distance criterion possessing unique analytical properties that we exploit to achieve a very fast computation. CLUBS derives optimal clusters without requiring input from users, and it is robust and impervious to noise, while providing better speed and accuracy than methods, such as BIRCH, that are endowed with the same critical properties.

1 Introduction

Many clustering algorithms have been proposed over the years [6], including algorithms for partition-based clustering (e.g. *k-means*[8]), density based clustering (e.g. *DBScan*[3]), hierarchical methods (e.g. *BIRCH*[11]) and grid-based methods (e.g. *STING* [10]), to mention only a few. Even so, the clustering remains a many-facet problem, and there is no single algorithm that ensures *speed, repeatability,* and *quality* of results. For instance, the old k-means algorithm remains one of the most widely used algorithms due to its superior speed. But k-means does not achieve good *repeatability* of results, since different choices of the of the initial k points will produce different results, thus falling short of the 'unsupervised learning' ideal. Indeed, k-means' shortcomings have motivated much research work [9, 1, 2], seeking to reduce the variability of the results by careful seeding of the initial k points. These approaches also improves the quality of the results, by maximizing inter-cluster distances, and minimizing the intra-cluster similarities (formal measures of quality will be discussed later). As noted by various authors and textbooks ([7, 6]) hierarchical clustering algorithms can produce the best performances both in terms of clustering quality and repeatability, and find important results in domains such as biology, medicine, and image processing, but they also tend to be computationally expensive. This has motivated a significant research line that attempted to overcome the computational inefficiencies of these algorithms while preserving their desirable properties. For instance in [5] a fast agglomerative clustering method is proposed that uses approximate nearest neighbor graph for reducing the number of distance calculations required and thus the complexity of the clustering algorithm. In this paper we instead observe that, for the frequent situations where Euclidean distance is used as the measure of similarity, its analytical properties can be exploited to design a hierarchical clustering algorithm that achieves superior performance in the three dimensions of speed, repeatability and cluster quality.

The new algorithm is called CLUBS (for CLustering Using Binary Splitting) and operates in two phases which are both carried out in a completely unsupervised mode. The first phase of CLUBS is divisive, as the original set is split recursively into miniclusters through successive binary splits: the algorithm's second phase is agglomerative since these miniclusters are recombined into the final result.

2 Background

We are given a set D of N points in a d-dimensional space; we assume without loss of generality, that all dimensions of D have the same size. The number of non-zero elements of D will be denoted as N. A *range* ρ_i on the i-th dimension of D is an interval $[l..u]$, such that $1 \leq l \leq u \leq n$. Boundaries l and u of ρ_i are denoted by $lb(\rho_i)$ (*lower bound*) and $ub(\rho_i)$ (*upper bound*), respectively. The size of ρ_i will be denoted as $size(\rho_i) = ub(\rho_i) - lb(\rho_i) + 1$. A *block* b (of D) is a d-tuple $\langle \rho_1, \ldots, \rho_d \rangle$ where ρ_i is a range on the dimension i, for each $1 \leq i \leq d$. Informally, a block represents a "hyper-rectangular" region of D. A block b of D with all zero elements is said to be a *null block.* The volume of a block $b = \langle \rho_1, \ldots, \rho_d \rangle$ is given by $size(\rho_1) \times \ldots \times size(\rho_d)$ and will be denoted as $vol(b)$. Given a point in the multidimensional space $\mathbf{x} = \langle x_1, \ldots, x_d \rangle$, we say that $\mathbf{x}$ belongs to the block b (written $\mathbf{x} \in b$) if $lb(\rho_i) \leq x_i \leq ub(\rho_i)$ for each $i \in [1..d]$.

Given a block $b = \langle \rho_1, \ldots, \rho_d \rangle$, let x be a coordinate on the i-th dimension of b such that $lb(\rho_i) \leq x < ub(\rho_i)$. Coordinate x divides the range ρ_i of b into $\rho_i^{low} = [lb(\rho_i)..x]$ and $\rho_i^{high} = [(x+1)..ub(\rho_i)]$, thus partitioning b into $b^{low} = \langle \rho_1, \ldots, \rho_i^{low}, \ldots, \rho_d \rangle$ and $b^{high} = \langle \rho_1, \ldots, \rho_i^{high}, \ldots, \rho_d \rangle$. The pair $\langle b^{low}, b^{high} \rangle$ is said to be the *binary split* of b along the dimension i at the position x; dimension i and coordinate x are said to be the *splitting dimension* and the *splitting position*, respectively.

Informally, a binary partition can be obtained by performing a binary split on D (thus generating the two sub-blocks D^{low} and D^{high}), and then recursively partitioning these two sub-blocks with the same binary hierarchical scheme.

Definition 1. *Given a d-dimensional data distribution D with volume n^d, a* binary partition *BP of D is a binary tree such that the root of BP is the block $\langle [1..n], \ldots, [1..n] \rangle$ and for each internal node p of BP the pair of children of p is a binary-split of p.*

Given a dataset $\mathcal{DS}$ cluster analysis aims at producing a clustering $C = \{C_1, \cdots, C_n\}$ that is a subset of the set of all subsets of $\mathcal{DS}$ such that C contains disjoint (non-overlapping) subsets, covering the whole object set (we refer in this paper exclusively to hard clustering problem, where every data point belongs to one and only one cluster). Consequently, every point $x \in \mathcal{DS}$ is contained in exactly one and only one set C_i. These sets C_i are called clusters.

Definition 2. *Let* C_s *be a cluster (set) of* N d*-dimensional points. Let* $\mathbf{S} = (S_1, \ldots, S_d) = \sum_{\mathbf{p}\in C_s} \mathbf{p}$ *be the vector representing the sum of points in* C_s*. The center of* C_s *is* $\mathbf{C_s^0} = \frac{\mathbf{S}}{N}$*. Let* $\mathbf{Q} = (Q_1, \ldots, Q_d)$*, where* $Q_i = \sum_{\mathbf{p}\in C} p_i^2$*, be a vector whose* i^{th}*coordinate is the sum of the squared* i^{th}*coordinates of the points in* S*. The SSQ of* C_s *is defined as:*
$SSQ(C_s) = \sum_{\mathbf{p}\in C_s} dist^2(\mathbf{p}, \mathbf{C_s^0}) = \sum_{\mathbf{p}\in C} \sum_{i=1}^{d} (p_i - C_s^0)^2 = \sum_{i=1}^{d} \sum_{\mathbf{p}\in C} (p_i - C_s^0)^2 = \sum_{i=1}^{d} \sum_{\mathbf{p}\in C} (p_i^2 - 2 \cdot p_i \cdot C_s^0 + (C_s^0)^2) = \sum_{i=1}^{d} \sum_{\mathbf{p}\in C} p_i^2 - 2 \cdot C_s^0 \cdot \sum_{\mathbf{p}\in C} p_i + N \cdot (C_s^0)^2)$
Since N *is the number of points in* C *and* $\sum_{\mathbf{p}\in C} p_i = C_s^0 \cdot N$
we have that: $\sum_{i=1}^{d} \sum_{\mathbf{p}\in C} p_i^2 - \sum_{i=1}^{d} \frac{(\sum_{\mathbf{p}\in C} p_i)^2}{N}$.
Finally from the definition of Q_i *and* S_i *we obtain:*
$SSQ(C_s) = \sum_{i=1}^{d} (Q_i - \frac{S_i^2}{N})$ (1)

Therefore, to quickly compute the SSQ of a cluster, we need only to store $\mathbf{Q}$, $\mathbf{S}$, and N. To select an optimal binary partition, we need to estimate the SSQ reduction obtained by this split. Suppose to split a cluster C_s at a given position j, represented by $\langle Q, S, N \rangle$ into two clusters $C_{s'}$ and $C_{s''}$ represented respectively by $\langle Q_1, S_1, N_1 \rangle$, and $\langle Q_2, S_2, N_2 \rangle$. Thus, the SSQ reduction is the non negative value ΔSSQ that can be obtained by simply summing up the SSQ difference in all dimensions (the mathematical steps are omitted for the sake of simplicity):

$$\Delta SSQ(i, j) = \sum_{i=1}^{d} \Delta SSQ_i(j) \quad (3)$$

3 Our proposal

Among hierarchical algorithms, bottom-up approaches tend to be more accurate but have a higher computational cost than the top-down approaches. Thus, in CLUBS, the agglomerative step is only used on mini-clusters generated by a first divisive process. Top-down partitioning exploiting greedy algorithms has been widely used in multidimensional data compression due to its efficiency. Here we use a similar divisive approach to minimize the SSQ among the data belonging to clusters. Thus, our clustering algorithm consists of two steps, where in the first step we use binary hierarchical partitioning to produce a set of mini-clusters and in the second step, we pairwise merge the mini-clusters so obtained in a bottom-up fashion. In both steps the clusters are defined by a hierarchical partition of the multi-dimensional space. The partition can be compactly represented by a binary tree, where: *i)* each node is associated with a range of the multi-dimensional domain; *ii)* the root is associated with the whole data domain; *iii)* for each inner node n, its children are associated with a pair of ranges representing a (rectangular) partition of n. Each node maintains summary information about points inside its range, to expedite the clustering computation.

The top-down splitting works as follows. As auxiliary structure, we maintain a priority queue of clusters whose elements are ordered on the basis of the SSQ of each cluster. At each iteration, the algorithm performs the following two steps: A) select the cluster C_s that exhibits the highest SSQ (i.e. the one on top of the priority queue), and then B) partition this C_s in such a way that the SSQ reduction, denoted $\mathit{\Delta} SSQ$, is maximized. For step B, we use formula (3) to compute $\mathit{\Delta} SSQ(i,j)$ for each dimension i and for each cutting position j; then we choose the position j that guarantees the maximum $\mathit{\Delta} SSQ$. This computation can be done very efficiently since we pre-compute Q and S, and therefore we need a single scan of the data. We repeat these two steps, A and B above, while $\mathit{\Delta} SSQ$ is greater than the average SSQ. We recall that the partition (i.e., the cluster tree) is built by exploiting a greedy strategy. To this end, the tree is constructed top-down, by means of leaf-node splitting. At each step, the leaf with the largest SSQ is chosen, and it is split as to maximize the SSQ reduction, denoted $\mathit{\Delta} SSQ$. Being SSQ a measure of a range skewness, we perform splits as long $\mathit{\Delta} SSQ$ remains "significant". After the early splits that yield large SSQ reductions, the values of $\mathit{\Delta} SSQ$ become smaller and smaller, until after n splits both SSQ and $\mathit{\Delta} SSQ$ become 0 (since each point has become its own cluster). Thus, the average SSQ reduction per split is SSQ_0/n, and we will compare this value against the current δSSQ to decide when we should stop splitting.

The splitting process just described is tied to the grid partitioning and thus may cause a non-optimal splitting of some clusters. The successive phase overcomes this limitation since the merging is performed considering all the possible pairs of adjacent mini-clusters, and recombining those that offer the best SSQ reduction. This agglomerative process offers significant advantages. One is that it merges clusters in different grid partitions, thus overcoming non optimal splits obtained in the first phase. The second critical advantage is that the computational complexity of this bottom-up step is very low since the number of merging steps is related to the number of clusters that is very low compared to usual dataset sizes. The final advantage is that this phase also halts automatically, producing an algorithm that does not require any seeding or other parameters from the user.

Figure 1 provides a more formal description of the CLUBS algorithm. We use the *initializeTree* to load the dataset into the root of the auxiliary tree structure BT exploited for partitioning. Once the tree structure has been initialized the *topdownsplitting* step starts. In particular, the root of BT is added to a priority queue whose ordering criterion is based on the SSQ values of clusters stored in the queue. The initial cluster assignment performed by *initializeClusters* is composed by the root r of BT and the initial SSQ is the one computed on r. The function *computeAverageDeltaSSQ* averages the actual SSQ for all the points in the cluster. The function *computeWeightedDeltaSSQ* is applied to the cluster C_s that is currently on top of the priority queue. The $weighted\mathit{\Delta}_{SSQ}$ is computed as the average gain of SSQ obtained by splitting C_s as explained above for $\mathit{\Delta}_{SSQ}$; thus, we pre-compute the partial sums (S and Q) for a given splitting point (w.r.t., the coordinates ordering) and reassign the splitting point on the

```
Input:
A dataset $DS$ of $n$ points
Output:
A set of clusters $C$.
Vars:
An auxiliary binary tree $BT$;
An initial cluster assignment $C'$.
Method: CLUBS
$BT :=$ initializeTree($DS$);
$C' :=$ topDownSplitting($BT$);
$C :=$ bottomUpMerging($C'$);
return $C$;
```

```
Function topDownSplitting($BT$) : $C'$;
Vars:
A priority queue $PQ$;
A boolean $finished$;
A double $\Delta_{SSQ}$;
A double $avgDeltaSSQ$;
Method:
$PQ :=$ add($BT.root()$);
$C' =$ initializeClusters;
$finished = false$;
$avg\Delta_{SSQ} =$ computeAverageDeltaSSQ();
while $!finished$ do begin
    $C_s = PQ.get()$;
    $weighted\Delta_{SSQ} =$ computeWeightedDeltaSSQ($C_s$);
    if ($weighted\Delta_{SSQ} > avg\Delta_{SSQ}$) then
        $C' :=$ update($C'$);
    else $finished = true$;
end while
return $C'$;
```

```
Function bottomUpMerging($C'$) : $C$;
Vars:
A pair of cluster $Pair$;
A double $avgDeltaSSQ$;
A double $minInc$;
Method:
$C := C'$;
$Pair :=$ selectBestPair($C'$);
$minInc :=$ computeSSQIncrease($Pair$);
$avgDeltaSSQ =$ computeAverageDeltaSSQ();
while $minInc < avgDeltaSSQ$ do begin
    $C :=$ merge($Pair$);
    $Pair :=$ selectBestPair($C$);
    $minInc :=$ computeSSQIncrease($Pair$);
end while;
return $C$;
```

Fig. 1. The CLUBS clustering algorithm

basis of these partial sums. In order to improve the effectiveness of splits, the value of Δ_{SSQ} is raised to a power of p, $p < 1$, thus obtaining $weighted\Delta_{SSQ}$ value. If $weighted\Delta_{SSQ}$ is greater than $avgDeltaSSQ$ computed by *computeAverageDeltaSSQ* then we proceed with the split, otherwise we do not. We use values of p that are less than 1, since for $p \geq 1$ we would end up splitting clusters where the gain does not exceeds the Average Δ_{SSQ} associated with a random distribution. This would result in a large number of small clusters, where both intra-cluster and inter-cluster distances are small. We instead seek values of p that reduce the former while magnifying the latter, after a deep experimental evaluation not reported here due to space limitations, we determined that the best value is $p = 0.8$. When no more top-down splits are possible, the *topDownSplitting* ends and we begin the *bottomUpMerging*. In order to obtain more compact clusters, we select (by running *selectBestPair*) the pair of clusters that, if merged, yields the least SSQ increase (that is assigned to $minInc$ by function *computeSSQIncrease*). This merging step is repeated until $minInc$ becomes larger than $avgDeltaSSQ$.

4 Experimental Evaluation

An extensive set of experiments was executed to evaluate the performance of CLUBS. In particular, we compared our method with BIRCH [11], K-means++ [1](we refer to it as KM++) and k*-means [2] (we refer to it as SMART). For the last three algorithms (i.e., the k-means based ones) we performed 20 runs (same as [1]) and we report the average and worst values for these runs. Comparisons with the basic k-means algorithm were also executed; they confirmed the significant improvements brought by KM++ [1], which we will thus use to compare against. All algorithms were implemented using the Java language Version 6 on a Intel Core Duo 2GHz equipped with 4GB RAM.

Our test suite encompasses a large number of widely used benchmarks over a wide spectrum of different characteristics. The first group of datasets are all publicly available[3] and can be divided in three subgroups as follows: *Birch:* This benchmark contains three datasets having 100,000 tuples arranged in 100 clusters of different shapes[4]; *A:* This dataset contains synthetic 2-d data with varying number of clusters and vectors. In particular A_1 contains 3000 tuples arranged in 20 clusters, A_2 contains 5250 tuples arranged in 35 clusters and A_3 contains 7500 tuples arranged in 50 clusters; *S:* This dataset contains synthetic 2-d data with 5000 tuples and 15 Gaussian clusters with varying degrees of overlap. We also validated CLUBS on experimental datasets from UCI [4], including the *US Census Data (1990)* consisting of 2458285 tuples over 68 attributes cleaned and preprocessed for mining.

We performed several experiments in different stressing settings where CLUBS performances were very good, due to space limitations we report here a subset of these experiments. Table 1 shows the running times of the various algorithms on our synthetic datasets as we increase the the number of tuples from $3,000$

[3] http://cs.joensuu.fi/sipu/datasets/

[4] We keep the dataset names used by the authors, while we use all caps for the names of algorithms.

to $100,000$: the datasets are assuming 30 clusters and a Gaussian distribution for points. We see that, as the number of tuples grows, CLUBS outperforms the other methods in terms of speed: KM++ is a close second and SMART and BIRCH are significantly slower–typically by a factor of four or five. Table 1 compares the speed and accuracy of our four algorithms on *Birch*, A and S datasets. We report the running times in seconds, and the accuracy measured by the SSQ value produced (the smaller the SSQ the higher the accuracy, i.e., the quality of clusters). Although the datasets present very different characteristics and statistics, CLUBS consistently outperforms the other algorithms, in both speed and accuracy, as shown in Table 1.

Table 1. Performances for our test datasets

Algorithm	***Birch Datasets***					
	$Birch_1$		$Birch_2$		$Birch_3$	
	SSQ	time	SSQ	time	SSQ	time
CLUBS	5.05E+13	23.703	1.77E+12	33.406	2.51E+13	21.281
BIRCH	5.67E+13	89.671	1.78E+12	93.433	2.86E+13	87.247
KM++	6.41E+13	28.441	1.76E+12	36.118	3.01E+13	27.874
SMART	8.91E+13	78.145	1.81E+12	88.154	3.66E+13	88.643
Algorithm	***A Datasets***					
	A_1		A_2		A_3	
	SSQ	time	SSQ	time	SSQ	time
CLUBS	1.43E+10	0.875	2.41E+10	1.063	3.71E+10	1.375
BIRCH	1.51E+10	1.469	2.46E+10	2.86	3.78E+10	5.012
KM++	1.57E+10	0.991	2.51E+10	1.112	3.82E+10	1.216
SMART	2.11E+10	1.332	2.67E+10	2.434	3.85E+10	4.781
Algorithm	***S Datasets***					
	S_1		S_2		S_3	
	SSQ	time	SSQ	time	SSQ	time
CLUBS	7.34E+12	8.125	7.87E+12	10.812	6.42E+12	12.313
BIRCH	8.01E+12	49.774	8.99E+12	52.446	6.55E+12	56.881
KM++	7.87E+12	7.214	8.81E+12	10.443	7.11E+12	12.741
SMART	9.15E+12	25.153	1.08E+13	37.134	7.75E+13	42.166

The best accuracy is obtained for the $Birch_2$ data set where the data is arranged in a large number of small clusters; this large number results in a smaller SSQ than for other data sets where there are fewer clusters. The best speed is obtained for $Birch_3$, where the data set is arranged in many big clusters, a situation that reduces the number of partition steps taken by the algorithms. Finally, the $Birch_1$ data set contains well-spaced clusters that are less compact: this results in the faster speed, and larger SSQ values seen in Table 1. The A data sets contain a relatively small number of tuples (ranging from 3000 to 7500) and several compact clusters uniformly distributed in the 2-D space. This situation makes the outcomes for dataset A less sensitive to the initial center assignment when executing k-means and other algorithms based on k-means. Nevertheless, as shown in Table 1, CLUBS provides better performance in both execution times and accuracy. For this data sets, BIRCH obtains an accuracy that is nearly as good as CLUBS, but it is significantly worse in terms of speed.

In the S data sets, the samples are arranged in several (up to 15) clusters which are globular in shape inasmuch as they have been generated using a Gaussian distribution. This is the ideal situation for the KM++ algorithm [1] that in fact achieves somewhat better average speed than CLUBS on data sets S_1 and S_2 (but not on S_3). For these larger data sets, the BIRCH algorithm runs significantly slower. In terms of accuracy, CLUBS consistently outperforms the other algorithms. Also, it is important to remember that for KM++, Table 1 reports the average speed: the worst case speed can be significantly worse, particularly in the presence of noise, as it will be discussed in later sections.

Table 2. Performances for UCI Census datasets

Algorithm	***UCI Dataset***	
	SSQ	time
CLUBS	4.47E+10	43.114
BIRCH	4.96E+10	112.634
KM++	5.84E+11	52.667
SMART	6.05E+11	109.459

In Table 2, we report the results obtained on the UCI Census Dataset [4], consisting of about 2.5 millions of relational tuples with 68 attributes. The table shows clearly that CLUBS outperforms all other algorithms in terms of accuracy and speed. For instance, on this large data set, CLUBS is three time faster than BIRCH, an algorithm that was designed for scalability. Even more remarkably, CLUBS is an order of magnitude more accurate than KM++, a very significant difference in view of the fact that KM++ was given as input $k = 162$, i.e., the optimal number of clusters determined by CLUBS. Thus, the results on this large and widely used dataset confirm the superior performance of CLUBS over other clustering methods, both in terms of accuracy and speed (even when the crippling problem of determining the correct k for KM++ is ignored).

References

1. D. Arthur and S. Vassilvitskii. k-means++: the advantages of careful seeding. In *In Proc. of Symposium on Discrete Algorithms (SODA)*, pages 1027–1035, 2007.
2. Y.M. Cheung. k*-means: A new generalized k-means clustering algorithm. *Pattern Recognition Letters*, 24(15):2883–2893, 2003.
3. M. Ester, H.P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *In Proc. of Knowledge Discovery And Data Mining (KDD)*, 1996.
4. A. Frank and A. Asuncion. UCI machine learning repository, 2010.
5. P. Franti, O. Virmajoki, and V. Hautamaki. Fast agglomerative clustering using a k-nearest neighbor graph. *IEEE Trans. Pattern Anal. Mach. Intell.*, 28:1875–1881, November 2006.
6. J. Han and M. Kamber. *Data Mining: Concepts and Techniques.* Morgan Kaufmann, 2000.
7. A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31, September 1999.
8. J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *5-th Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.
9. R. Ostrovsky, Y. Rabani, L. J. Schulman, and C. Swamy. The effectiveness of lloyd-type methods for the k-means problem. In *In Proc. of IEEE Symposium on Foundations of Computer Science (FOCS)*, 2006.
10. W. Wang, J. Yang, and R. R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *In Proc. of Very Large Data Base (VLDB)*, pages 186–195, 1997.
11. T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *In Proc. of Special Interest Group on Management Of Data (SIGMOD)*, pages 103–114, 1996.

Clustering of Process Schemas by Graph Mining Techniques (Extended Abstract)

Claudia Diamantini, Domenico Potena, and Emanuele Storti

Dipartimento di Ingegneria Informatica, Gestionale e dell'Automazione "M. Panti",
Università Politecnica delle Marche - via Brecce Bianche, 60131 Ancona, Italy
{diamantini, potena, storti}@diiga.univpm.it

Abstract. In this work, we focus on the analysis of process schemas in order to extract common substructures. In particular, we represent processes as graphs, and we apply a graph-based hierarchical clustering technique to group similar sub-processes together at different levels of abstraction. We discuss different representation choices of process schemas that lead to different outcomes.

1 Introduction

Process Mining (PM) is the application of inductive techniques to extract general knowledge about business processes from process instances. In state of the art research, instances are traces of running processes recorded in the event logs of ERP, Workflow Management Systems or other enterprise systems, and the goal of PM is to distill a structured process description, from the set of real executions, representing the *process schema* [1]. This mining task can be exploited to support process mapping activities, conformance checking, detection of anomalous behaviors, and so forth [2].

In this paper we consider a novel process mining task: *given a set of process schemas, discover clusters of similar (sub-)processes and their relationships*. Although clustering of log traces has received notable attention in the literature as a means to enhance the quality of discovered process schemas [3,4,5], to the best of our knowledge the application of clustering techniques to process schemas themselves is completely new. It can find application in enterprise integration at the process level; as a matter of fact, given a set of processes with the same goals and implemented in different companies, the proposed PM task helps to enlighten overlapping, complementary and heterogeneous structures, which are useful patterns to drive the integration. Furthermore, recurrent common subprocesses can be exploited to define reference prototype processes and best practices (or common bad practices).

In order to achieve the goal, we discuss the application of SUBDUE [6], a hierarchical graph clustering algorithm. Graph clustering techniques have been considered since process schemas have a inherent graph structure, while hierarchical clustering in general, and SUBDUE in particular, allows to account for the inherent abstraction structure typical of processes (from very general macro-processes down to simple activities). Although process schemas can be seen as graphs, the application of

SUBDUE requires some choices in terms of how to represent complex flow control structures, like parallel and alternative execution of activities or merging.

To this end, after presenting the methodology in Section 2 we discuss different representation choices. Then in Section 3 some evaluation indexes are introduced and experimental results are reported. Finally, Section 4 briefly discusses the results and possible applicative scenarios.

2 Methodology

Given a set of directed graphs $G_i = <N_i, A_i>$ where N_i is the set of nodes and $A_i \subseteq N_i x N_i$ is the set of (possibly labeled) arcs, SUBDUE generates a clustering lattice of typical substructures. In its exact matching version, graphs are iteratively analyzed to discover at each step a cluster of isomorphic substructures. The cluster is then used to compress the graphs, by substituting to each occurrence of the substructure a single node. The compressed graphs are presented to SUBDUE again, and the process is repeated until no more compression is possible. The output clusters turn out to define a lattice where the clusters are linked if a cluster appears in the definition of another. At each step, the substructure is chosen on the basis of its compression capability, measured by the Minimum Description Length (MDL) heuristics. The description length of a graph is measured by the number of bits needed to represent its adjacency matrix. The algorithm has been successfully applied to analyze structured objects in several domains (see http://ailab.wsu.edu/subdue/) thanks to the flexibility it gives to represent complex objects in terms of mathematical graph structures, and suggesting it as a promising technique to analyze process schemas.

A process schema describes the flow of work performed by a certain number of actors. The kinds of flow include simple sequences of activities (SEQ), and operators used to model parallelization (hereafter called SPLIT) and merging (JOIN) of activities. In particular, a SPLIT-AND means that the end of an activity starts all the linked activities, while in a SPLIT-XOR only one will be executed. Symmetrically, a JOIN-AND indicates that an activity begins when all the previous activities are terminated; while in a JOIN-XOR the completion of a single activity is needed. Figure 1 shows an example of process using some of the described operators in BPMN notation.

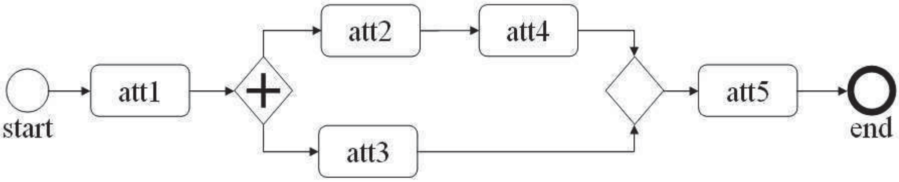
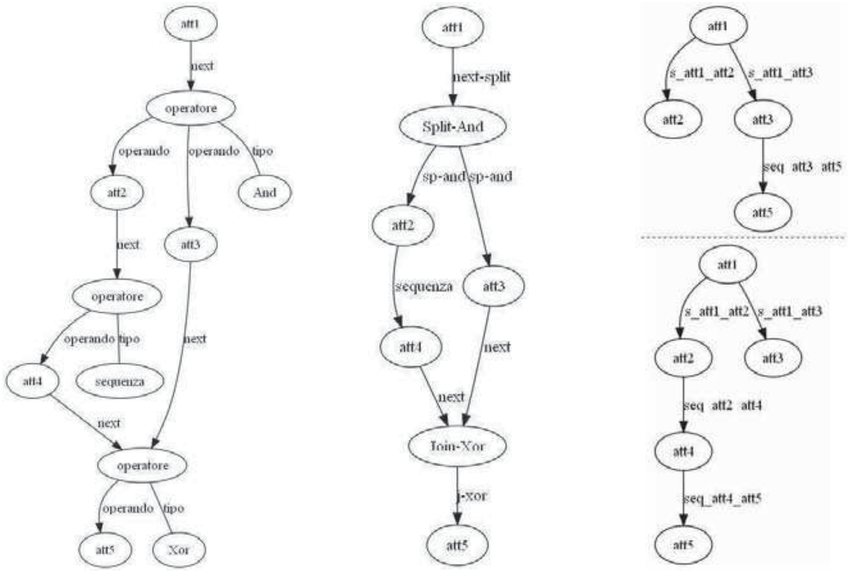


Fig. 1. An example of process schema. Activity *att1* is followed by both *att2* and *att3* (SPLIT-AND), and *att5* is started when *att4* or *att3* are completed (JOIN-XOR).

The application of SUBDUE to business processes requires to perform a mapping from the richer process graph to simpler directed graphs. As we will see, different representation choices may influence the final clustering result. While it seems straightforward to represent the SEQ operator by an arc in the graph, the representation of other operators is not straightforward. We present here three different models, named A, B, and C respectively, and characterized by an increasing level of compactness of the graph, without loss of information. In the A model, any operator is represented by a node called *operator*, which is linked to another node specifying the AND or XOR nature of the operator. In this model join and split are distinguishable by the number of ingoing and outgoing arcs (one outgoing arc and several ingoing arcs for join, the opposite for split). In the B model the node *operator* is replaced by different nodes one for each kind of operator. Finally, the C model simplifies the graph by removing both join and split nodes: since JOIN-XOR and SPLIT-XOR operators represent different alternative executable paths, one for each ingoing (outgoing) activity of a join (split) operator, XOR nodes can be removed by individuating all the possible alternative paths in the process, and generating a graph for each path. In this way, there is no ambiguity about the AND nature of arcs leaving (entering) a node, so AND nodes can be removed too. Figure 2 shows the representation of the process in Figure 1 with respect to A, B and C models. Note that the three representations hold the same information, and the last produces two compact graphs (one for each xor path). Note also the use of labeled arcs in the C model of Figure 2 to maintain information about domain and range nodes. This is necessary to guarantee the correct interpretation of the final lattice after the compression performed by SUBDUE. It is straightforward to see that these representation strategies can be simply extended to include other BPMN constructs as well (in fact, the first two are directly related to the approach presented in [7]).



model A model B model C

Fig. 2. The representation of the process schema in Figure 1 in conformity with the three proposed models

3 Experimental Evaluation

We experimented the methodology on a set of prototype processes describing e-science activities. In particular, we use a set of data mining processes for the classification task produced in the KDDVM project (http://boole.diiga.univpm.it). Activities are chosen among 21 algorithms of different kind (classification, pre-processing and post-processing) to generate a set of 40 different prototype processes. We also present preliminary results of an experimentation over a set of real workflows taken from the e-science project MyExperiment (http://www.myexperiment.org), which defines a collaborative web platform enabling researchers to share, annotate, comment, discover workflows.

In order to evaluate the resulting SUBDUE lattice with different representation strategies and the potentiality of the approach, we introduce some indexes: completeness, representativeness and significance. *Completeness* measures the number of original graph elements still present in the final lattice[1]. It is expressed as $C = \frac{N_O + A_O}{N_I + A_I}$, where I is the set of input graphs and O is the final lattice. Node completeness is also considered. While completeness measures a quality of the whole lattice, the other indexes allows to individually evaluate each cluster.

The *representativeness* of a substructure measures the number of input graphs holding the given substructure at least once. More precisely, representativeness of the substructure S_i is: $R(S_i) = \frac{G(S_i)}{G}$, where $G(S_i)$ is the number of processes holding S_i in graph G. High values of $R(S_i)$ indicate S_i as a typical subprocess.

Finally, *significance* is a qualitative index that evaluates the meaning of a cluster with respect to the domain. This index allows us to disregard those clusters that are very representative, but do not contain useful knowledge.

	A Model	B Model	C Model
Completeness	97%	94%	92%
Nodes Completeness	99%	99%	98%
Representativeness of high level clusters	7%– 67%	8%-31%	8%-40%
Significance of top level clusters	- -	+	+ +

Table 1. Comparison of lattices obtained from graphs represented in accordance with the A, B and C models.

In Table 1, we synthetically show results of experimentations over KDDVM prototype processes in terms of indexes values. In particular, clusters indexes are reported only for high level clusters, which represent the most common substructures. From Table 1, it results that all models are characterized by high completeness, even if C model leads to a slight decrease in the value of such index. The low significance of top level clusters obtained using A model is due to the fact that most frequent sub-

[1] As a matter of fact, during the lattice generation, SUBDUE discards those substructures having low compression capability. This may lead to loose some node or arc.

structures are nodes representing individual operators, without references to involved activities. The highest values of representativeness for A model also depends on the high frequency of top level clusters. The C model is that allowing to achieve overall best results, reporting as top level clusters high-frequency substructures that are common in input graphs and are significant in the domain: they are actually knowledge patterns.

Figure 3 shows some of these knowledge patterns. We can see that the most used classification algorithms in the set of data mining processes are BVQ and C4.5. Furthermore, the practice of applying pre-processing algorithms to remove missing values and reduce the dimensionality of datasets emerges as typical patterns. We conclude by noting that SUB_9 and SUB_4 enlighten a not well-formed pattern, since removeMissingValue is performed after LDA. This is not a clustering error, rather it enlighten some problems in input process schemas.

The experimentation performed on MyExperiment is based on a subset of graphs, selected from the whole collection of scientific workflows, The platform contains workflows for a variety of systems, most notably Taverna [8], which represents workflows through the XML-based dataflow-centric language Scufl. Given the huge amount of workflows available, we restricted the analysis on Taverna workflows within the "protein" domain. Scufl workflows are made of processors (i.e.: nodes), which represent logical services, and data links, which map data sources to data sinks. In Scufl, processors can belong to several types, among which WSDL (a remote web service), local (a class from local Java libraries), script (a hand-written piece of Java code, usually aimed at data transformation), stringconstant (a literal constant) and others. Besides these, a special kind of processor is the *workflow*, that is a subworkflow nested in its parent workflow.

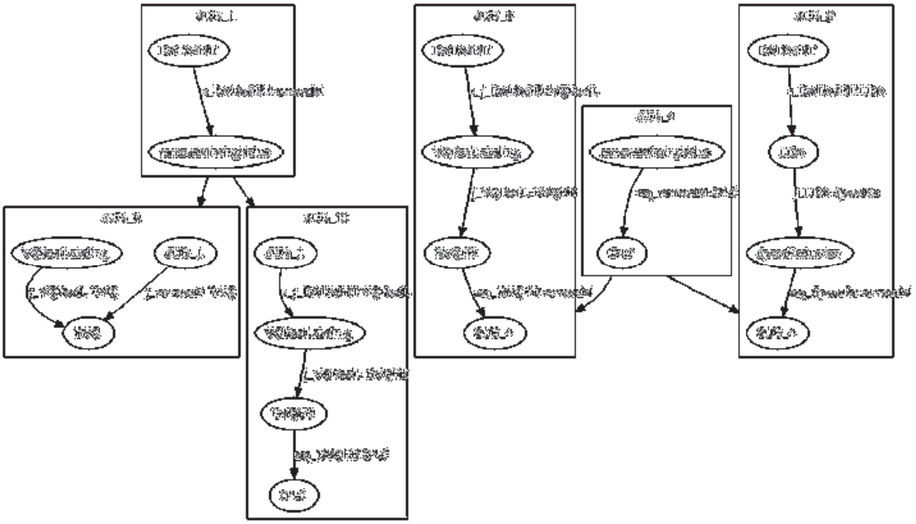


Fig. 3. First two levels of the lattice generated using C model.

The experimental dataset was generated by (1) extracting all Taverna workflows annotated with the tag "protein", (2) parsing their Scufl descriptors to extract processors and links, and (3) performing some preprocessing steps. In particular, since SUBDUE can manage only 1-level graphs, during the parsing the original workflows have been rewritten by flattening the nested subworkflow structure. Processors like stringconstant and Script have been ignored, since they are not considered useful for the analysis. As a matter of fact, the former can be considered as input variables, while the latter is code usually inserted to adapt a processor's output to an input, specific to a single workflow. Finally, some types of processors occur very rarely, so that they bring very little information. In particular we decided to discard seqhound and apiconsumer processors, which occur in less than 1% of the dataset.

After the preprocessing steps, the dataset contains 33 workflows, 432 nodes (57.18% of type "local" and 42.82% of type "WSDL") and 372 links.

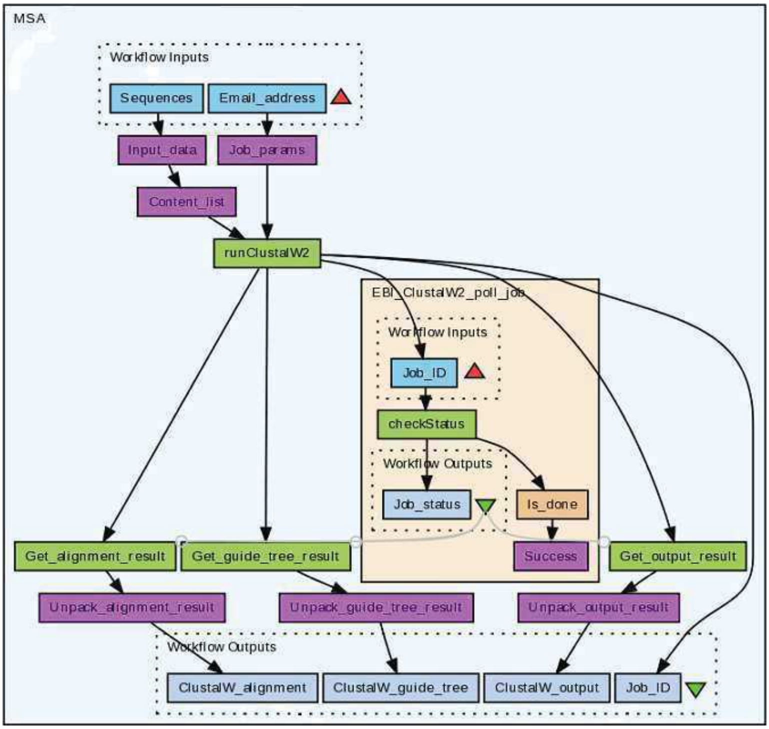


Fig. 4. The most typical sub-workflow in the MyExperiment dataset. runClustalW2, checkStatus, Get_% are web services, Input_data, Job_params, Content_list, Success and Unpack_% are local classes.

As input file for SUBDUE we generated a single graph containing every workflow from the dataset, coded through the C model. Experimentations were carried on with different parameters values. The best results showed a lattice formed by 107 substructures, in which the 3 most frequent substructures are contained in the 21% of workflows. As an example, in Figure 4 we show the sub-workflow corresponding to SUB_1, the most typical substructure found. It represents the typical usage schema for ClustalW2, a popular multiple sequence alignment algorithm used in bioinformatics to align DNA and protein sequences in multiple sequence formats. The workflow shows the following steps, which are frequent when ClustalW2 is to be executed: input preparation, submission of a job to the web service, check of the job status, download of the result and its postprocessing. Typical flow structures like the one shown in Figure 4, can be useful for supporting a non-expert user in the correct use of elements for the composition of a new experiment.

4 Discussion

The paper presents preliminary results about the feasibility of a graph-based clustering approach to recognize similarities among business processes, and to select significant prototypes. In particular, different representation alternatives of a business process for the application of SUBDUE algorithm have been discussed and evaluated.

The evaluation on real business processes has been made difficult by the lack of a sufficient number of process schemas, hence we turned to a specialized domain like data mining, exploiting processes automatically generated by an ontology-based composer tool. Nevertheless, this activity allowed to gain useful insights on the method and on the particular domain as well. For instance, from the analysis of the generated lattice we were able to recognize typical patterns of the KDD methodology and we gained insights about some missing or wrong information in the ontology guiding the activity of process generation. Also a promising, although preliminary, evaluation on a real e-science domain has been presented. In both experiments processes are designed to operate within specific platforms, so there is no heterogeneity issue. Indeed, when processes are designed by different authors, both name conflicts and structure conflicts can occur. In particular, structure conflicts are usually due to the granularity level adopted in the design of the process, e.g. the use of sub-processes instead of flat workflows. If the case of heterogeneity a schema preprocessing is needed before applying the methodology proposed in the present work. These semantic aspects are beyond the aims of this paper, and will be addressed in future works.

The proposed method can find application in a variety of activities related to business process management: first, it can be exploited to individuate similarities and differences in the implementation of certain processes at different companies, enlightening overlaps, complementarities and heterogeneities, hence supporting enterprise integration at the process level. Second, recurrent common substructures can be exploited to define reference prototype processes and best practices (or common bad practices). Third, the method can be exploited to organize a process repository to enhance search and retrieval. We plan to gather a sufficient number of business processes in order to concretely deal with these applications, and to extend the e-science experimentation to the whole MyExperiment repository.

Bibliography

1. Van der Aalst, W.M.P. and Weijters, A.J.M.M. Process mining: a research agenda, Computers in Industry 53, 231–244 (2004).
2. Van der Aalst, W.M.P. Process Mining: Discovery, Conformance and Enhancement of Business Processes, Springer (2011).
3. Greco, G., Guzzo, A., Pontieri, L. and Saccà, D. Discovering Expressive Process Models by Clustering Log Traces, IEEE Transactions on Knowedge and Data Engineering, Vol. 18, No. 8, 1010-1027 (2006)
4. Song, M., Gunther, C.W. and Van der Aalst, W.M.P. Trace Clustering in Process Mining, Business Process Management Workshops, Lecture Notes in Business Information Processing, Vol. 17 (2009)
5. Jagadeesh, R. P., Bose, C. and Van der Aalst, W.M.P. Trace Clustering Based on Conserved Patterns: Towards Achieving Better Process Models, Business Process Management Workshops, Lecture Notes in Business Information Processing, Vol. 43, 170-181 (2010)
6. Jonyer, I., Cook, D. and Holder, L. Graph-Based Hierarchical Conceptual Clustering, in: Journal of Machine Learning Research, Vol. 2, 19–43 (2001).
7. Ouvans, C., Dumas, M., ter Hofstede, A.H.M. and Van der Aalst, W.M.P. From BPMN Process Models to BPEL Web Services, 2006. International Conference on Web Services, pp. 285-292, Chicago, IL (2006).
8. D. Hull, K. Wolstencroft, R. Stevens, C. Goble, M. Pocock, P. Li, and T. Oinn, Taverna: a tool for building and running workflows of services, Nucleic Acids Research, 34, 729-732 (2006).

Supporting Roll-Up and Drill-Down Operations over OLAP Data Cubes with Continuous Dimensions via Density-Based Hierarchical Clustering

Michelangelo Ceci[1], Alfredo Cuzzocrea[2], and Donato Malerba[1]

[1] Dipartimento di Informatica, Università degli Studi di Bari “Aldo Modo”
via Orabona, 4 - I-70126 Bari - Italy
{ceci,malerba}@di.uniba.it
[2] ICAR-CNR and University of Calabria
Via P. Bucci, 41C, I-87036 Rende, Cosenza, Italy
cuzzocrea@si.deis.unical.it

Abstract. In traditional OLAP systems, roll-up and drill-down operations over data cubes exploit fixed hierarchies defined on discrete attributes that play the roles of dimensions, and operate along them. However, in recent years, a new tendency of considering even continuous attributes as dimensions, hence hierarchical members become continuous accordingly, has emerged mostly due to novel and emerging application scenarios like sensor and data stream management tools. A clear advantage of this emerging approach is that of avoiding the beforehand definition of an ad-hoc discretization hierarchy along each OLAP dimension. Following this latest trend, in this paper we propose a novel method for effectively and efficiently supporting roll-up and drill-down operations over OLAP data cubes with continuous dimensions via a density-based hierarchical clustering algorithm. This algorithm allows us to hierarchically cluster together dimension instances by also taking fact-table measures into account in order to enhance the clustering effect with respect to the possible analysis. Experiments on two well-known multidimensional datasets clearly show the advantages of the proposed solution.

1 Introduction

Traditional OLAP data cubes are defined on top of discrete dimensions that expose fixed hierarchies [4]. To this end, attribute domains of these dimensions are first discretized, and then processed simultaneously in order to obtain the final cube, given a certain measure [4]. Despite this well-consolidated methodology, a recent trend focuses the attention on the problem of effectively and efficiently computing OLAP data cubes defined on top of continuous dimensions (e.g., [5, 7, 6]), as the latter are more suitable to capture real-life dynamics rather than the discrete case. Nevertheless, computing such kind of data cubes poses severe challenges, and several alternatives, such as approximation paradigms (e.g., [7]), have been studied to face-off drawbacks deriving from this challenge. On the other hand, OLAP has also been recognized not only as a “last-stage” technology, but also as an important enabling technology that allows us to enhance the expressive power and the quality of retrieved results of a number of Data Warehousing and Mining techniques (e.g., [2]).

At the convergence of these two relevant challenges of Data Warehousing and Mining research, in this paper we propose and experimentally assess a novel framework, called OLAPBIRCH, whose main goal consists in integrating the clustering algorithm BIRCH [8] and OLAP. This integration permits to boost the benefits of both methodologies into a complex knowledge discovery framework for next-generation applications ranging from analytics to sensor-and-stream data analysis and social network analysis. OLAPBIRCH relies on top of a complex methodology according to which the capability of the clustering algorithm BIRCH are combined with OLAP in order to build a complex hierarchical data structure, called CF-Tree, whose nodes contain clusters retrieved by BIRCH from the target dataset and are organized in an OLAP-like fashion. This permits to exploit all the deriving benefits such as multidimensional and multi-resolution data exploration, roll-up and drill-down operations, interactive exploration, and so forth [4]. Particularly, supporting roll-up and drill-down operations play a significant role, due to the fact that CF-Tree materializes the retrieved clusters at each node, hence this allows us to significantly speed-up the response time due to executing these critical OLAP operations with respect to the baseline case represented by computing new clusters from pre-existent ones at each roll-up (or drill-down) operation. As an important contribution to actual research, OLAPBIRCH considers continuous dimensions instead than classical discrete ones, which is relevant in OLAP (e.g., [5, 7, 6]).

In more details, the proposed approach integrates a revised version of BIRCH in order to obtain a hierarchical organization of dimension instances according to similarity computed both on dimension values (from dimension tables) and measure values (from the fact table). We consider the BIRCH algorithm because, in its original formulation, it shows tree important peculiarities: *i)* Efficiency: The algorithm time complexity is linear in the number of instances to cluster and has a constant space complexity. *ii)* Hierarchical: The algorithm allows us to obtain a hierarchical clustering of instances. *iii)* Incremental: As new instances are given to the algorithm, the hierarchical clustering is revised and adapted by keeping into account memory constraints. All these properties well fit to work with data warehouses, where problems coming from the huge amount of data and require for efficient and incremental solutions. For the specific goal we tackle in this paper, it is also necessary to resort to a hierarchical clustering solution that would permit to give to the OLAP users the opportunity to perform roll-up and drill-down operations over continuous attributes. At this aim, we can exploit the peculiarity of BIRCH that provides high balanced hierarchies that become necessary in OLAP frameworks.

The paper is organized as follows. In the next Section we present the proposed framework OLAPBIRCH. In Section 3, we present an empirical evaluation of the proposed framework an finally, in Section 4 we draw some conclusions.

2 OLAPBIRCH: Combining BIRCH and OLAP

In this section, for the sake of completeness, we first describe the BIRCH algorithm and then we describe proposed modifications that permit to integrate BIRCH in an OLAP framework.

2.1 BIRCH

The BIRCH algorithm [8] works on a hierarchical data structure that the authors call CF tree (Clustering Feature Tree). This data structure permits to partition the incoming data points in an incremental and dynamic way. Each node in the CF tree is called Clustering Feature: Given n data points in a cluster, each of which represented according to a d-size feature vector, CF vector of the cluster is defined as a triple $CF = (n, LS, SS)$, where LS is the linear sum and SS is the square sum of data points. The CF vectors are sufficient to compute information about subclusters like centroid, radius and diameter. They satisfy an important additivity condition, i.e. if $CF_1 = (n_1, LS_1, SS_1)$ and $CF_2 = (n_2, LS_2, SS_2)$ are the clustering features for sets of points S_1 and S_2 respectively, then $CF_1 + CF_2 = (n_1 + n_2, LS_1 + LS_2, SS_1 + SS_2)$ is the clustering feature for the set $S_1 \cup S_2$.

A CF tree is a balanced tree whose structure is similar to that of a B+tree and depends on two parameters: the branching factor B and a user defined threshold T that represents the maximum cluster diameter. Each non-leaf represents a cluster consisting of all the subclusters represented by its entries. In particular, a non-leaf node N_j contains at most B entries of the form $[CF_i, c_i]_{i=1,..,B}$, where c_i is a pointer to the $i - th$ child node of N_j and CF_i is the clustering feature of the cluster identified by c_i. A leaf node contains at most L (typically $L = B$) entries each of the form $[CF_i]$. In the leaves, each node has two pointers $prev$ and $next$ which are used to chain all leaf nodes together. The tree size depends on the T value: the larger the T, the smaller the tree.

The algorithm BIRCH builds a CF tree in four steps. In the first step, BIRCH iteratively receives single data points and builds an initial CF-tree. A point is inserted by inserting the corresponding CF value into the closest leaf. If an entry in the leaf can absorb the new point without violating the threshold T condition, its CF is updated. Otherwise, a new entry is created in the leaf node, and, if the leaf node then contains more than L entries, it and maybe its ancestors are split. In this phase, in order to satisfy RAM constraints, BIRCH frequently rebuilds the whole CF tree by increasing the threshold T and tries to merge as many CF nodes as possible. The rebuild happens sufficiently fast since all needed data is already in RAM. At the same time outliers are removed from the tree and are stored to disk. The algorithm starts with maximum precision at $T = 0$ and as the CF tree grows larger than the available memory, it iteratively tries to find suitable cluster sizes by increasing T to be larger than the smallest distance between two entries in the tree. In the second step, the algorithm condenses the CF tree to a desirable size depending on the clustering algorithm employed in step three. This can involve removing outliers and further merging of clusters. In the third step, the algorithm employs a *global* clustering algorithm using the CF tree's leaves as input. This step, as claimed in [8], permits to avoid the undesirable effect of the skewed input order, and splitting triggered by space constraints. In this phase, the CF vectors allow for effective distance metrics computation. In the last step, a labeling procedure is performed. This means that, if desired, the actual data points can be associated with the generated clusters.

In the implementation we provide, the used distance measure is the variance increase distance [8] defined as follows:

Definition 1 (Variance Increase Distance).
Let C_1 and C_2 be two clusters, where $C_1 = \{x_i\}_{i=1..n_1}$ and $C_2 = \{x_i\}_{i=n_1+1..n_2}$. The variance increase distance between C_1 and C_2 is defined as:

$$D = \sum_{k=1}^{n_1+n_2} \left(x_k - \frac{\sum\limits_{l=1}^{n_1+n_2} x_l}{n_1+n_2} \right)^2 - \sum_{i=1}^{n_1} \left(x_i - \frac{\sum\limits_{l=1}^{n_1} x_l}{n_1} \right)^2 - \sum_{i=n_1+1}^{n_2} \left(x_i - \frac{\sum\limits_{l=n_1+1}^{n_2} x_l}{n_2} \right)^2$$

In our implementation, the clustering algorithm for the third step is the well-known DBSCAN [3] algorithm that performs a density based clustering. Density based clustering is performed on cluster centroids (that can be easily computed from the CF vectors and represent aggregated data) and allows us to further aggregate data that show similar peculiarities.

2.2 OLAPBIRCH

The integration of the implemented BIRCH algorithm in the OLAP solution we present, is not a trivial task since different issues have to be considered: First, OLAP queries can consider all the levels of the hierarchy and not only the last level. This means that it is necessary to have refined clusters not only in the last level of the hierarchy, but also in intermediate levels. Second, in OLAP frameworks, the user is typically able to control size of hierarchies, but this is not possible in the original BIRCH algorithm. Third, although the last step of the BIRCH algorithm is not mandatory, this step is necessary in our framework in order to simplify the computation of OLAP queries. Fourth, in order to avoid the combinatorial explosion that is typical in multidimensional clustering, it is necessary to focus only on interesting continuous dimension attributes.

In order to face with the first issue, we revised the clustering algorithm in order to allow the system to run the *global* clustering algorithm also in intermediate nodes of the tree. At this purpose, we extended the CF tree structure by providing pointers $prev$ and $next$ to each internal node. This allows us to linearly scan a each single level of the tree. In Figure 1, we report a graphical representation of the CF tree structure used in the proposed framework.

As for the second issue, in addition to the memory space constraints, we consider also an additional constraint that forces tree rebuilding when a maximum number of levels (MAX_LEV) is exceeded. This is coherent with the goal of having a limited number of levels, as in classical OLAP systems.

As for the third issue, given the maximum number of levels MAX_LEV and the branching factor B, it is possible to use a numerical representation of the complete path of clusters for each dimension instance so that the classical B+tree index structure can be used in order to allow efficient computation of range queries[5]. The representation is in the form $< d_1 d_2 \ldots d_{MAX\_LEV} >$, where each d_i is a sequence of $\ulcorner log_2 B \urcorner$ bits that permits to identify each subcluster. The number obtained in this way is then used to perform roll-up and drill-down operations.

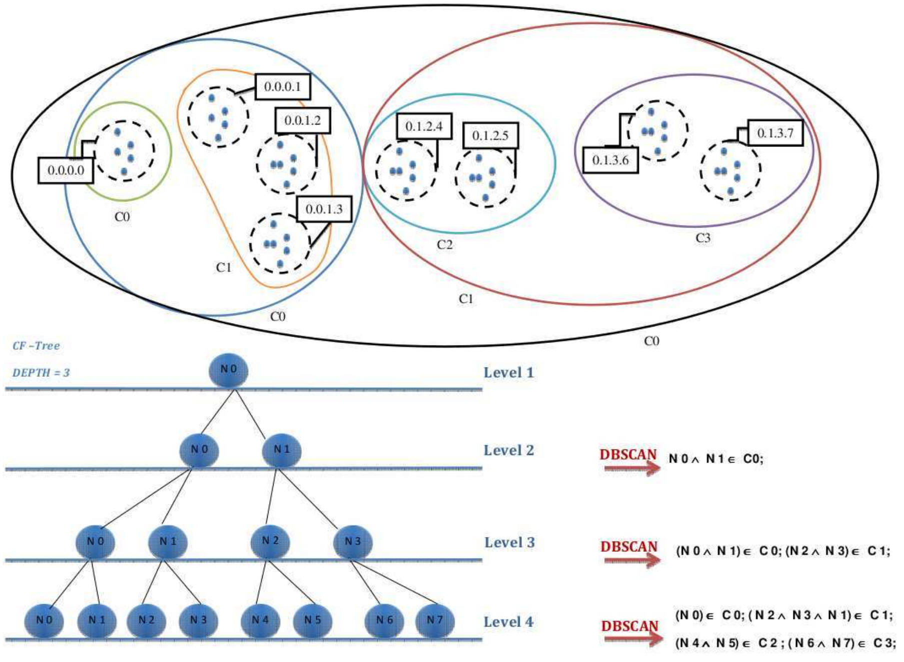


Fig. 1. OLAPBIRCH: an example of CF tree.

Finally, as for the fourth issue, in order to integrate the algorithm in an OLAP framework, we defined a language that permits to specify the attributes to be considered in the clustering phase. At this purpose, we have exploited the Mondrian[3] project that permits to represent a multidimensional schema of a data warehouse by means of an XML file. This file permits to define a mapping between the multidimensional schema and tables and attributes stored in the database. Main elements in this XML file are: the data source, cubes, measures, the fact table, dimensions and hierarchies.

For our purposes, we have modified the DTD in order to allow different types of hierarchies. The modified portion of the DTD is:

```
<!ELEMENT Hierarchy ((\%Relation;)?,(Level)*,
          (MemberReaderParameter)*,(Attribute)+, (Depth))>
   <!ATTLIST Hierarchy
          hasAll (true|false) #REQUIRED
          allMemberName CDATA #IMPLIED
          allMemberCaption CDATA #IMPLIED
          primaryKey CDATA #IMPLIED
          primaryKeyTable CDATA #IMPLIED
```

[3] http://sourceforge.net/projects/mondrian/files/mondrian/

```
          defaultMember CDATA #IMPLIED
          memberReaderClass CDATA #IMPLIED>
<!ELEMENT Attribute EMPTY>
    <!ATTLIST Attribute
          name CDATA #IMPLIED
          table CDATA #REQUIRED
          column CDATA #REQUIRED
          nameColumn CDATA #REQUIRED
          type (Numeric) Numeric #REQUIRED>
<!ELEMENT Depth EMPTY>
    <!ATTLIST Depth value (Numeric) Numeric #REQUIRED>
```

The DTD so modified permits to add two new elements ($< Attribute >$ and $< Depth >$) to the elements defined in $< Hierarchy >$. The $< Attribute >$ element permits to define one or more attributes to be used in the clustering procedure. Properties that can be defined in the $< Attribute >$ tag are: name - attribute name; table - table that contains the attribute; column: database column name; nameColumn: database column name (alias); type: SQL attribute type. The $< Depth >$ element permits to specify the maximum depth of the CF-tree.

The CF-tree is updated when a new dimension tuple is saved in the data warehouse while DBSCAN is run only when OLAP queries are executed and clusters are not updated. This permits to focus our attention only to levels that are actually used in the queries. It is noteworthy that, differently from [7], the global clustering is run on compact representations of data and does not pose efficiency problems.

Example 1. Le us consider the database schema reported in Figure 2 where $lineitem$ is the fact table and $orders$ is a dimensional table. By selecting, in the XML file, the attributes $orders.o\_totalprice$ and $orders.o\_orderpriority$:

```
< Attribute name=”totalprice” table=”orders” column=”
      o_totalprice” nameColumn=”o_totalprice”
      type=”Integer”/>
< Attribute name=”orderpriority” table=”orders” column=”
      o_orderpriority” nameColumn=”o_orderpriority”
      type=”Integer” />
< Depth value=”20”/>
```

we have that the OLAP engine performs clustering on the following database view:

```
SELECT l_quantity , l_extendedprice , l_discount , l_tax ,
          o_totalprice , o_orderpriority
FROM lineitem , orders
WHERE l_orderkey = o_orderkey
```

3 Experimental Evaluation and Analysis

In order to evaluate the effectiveness of the proposed solution, we performed experiments on two real world datasets. The first dataset is the SPAETH Cluster Analysis

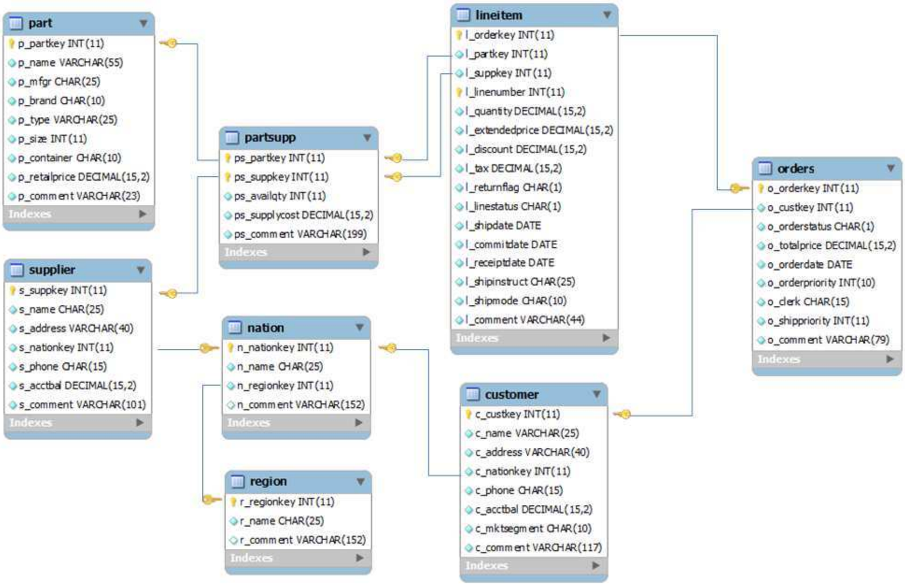


Fig. 2. TPC-H database schema

Datasets[4], a small dataset that allows us to visually evaluate the quality of extracted clusters. The second dataset is the well-know TPC-H benchmark (version 2.1.0)[5]. In Figure 2 we report the relational schema of TPC-H implemented on PostgreSQL, which we used as supporting DBMS. The TPC-H database holds data about the ordering and selling activities of a large-scale business company. For experiments we used the 1GB version of TPC-H [1] containing more that 1×10^6 tuples in the fact table. On this last dataset, we performed experiments on the scalability on the algorithm and we collected results in terms of running times and cluster quality. Clustering is performed on the fact table measures as well as on the the attributes orders.o_totalprice and orders.o_orderpriority as specified in Example 1. In order to force the OLAPBIRCH framework to work in the worst case scenario, running times are obtained by forcing the system to work when the examples are given one by one. The cluster quality is measured according to the *weighted average cluster diameter square measure*:

$$Q = \sum_{i=1..K} n_i(n_i - 1)D_i^2 / \sum_{i=1..K} n_i(n_i - 1) \tag{1}$$

where K is the number of obtained clusters, n_i is the cardinality of the i-th cluster and D_i is the diameter of the i-th cluster. The smaller the Q value, the higher the cluster quality.

In Figure 3, we report a graphical representation of obtained clusters. As we can see, the global clustering (DBSCAN) is necessary in order to have good quality clus-

[4] http://people.sc.fsu.edu/∼jburkardt/datasets/spaeth/spaeth.html

[5] Transactions Processing Council Benchmarks. Available from: http://www.tpc.org.

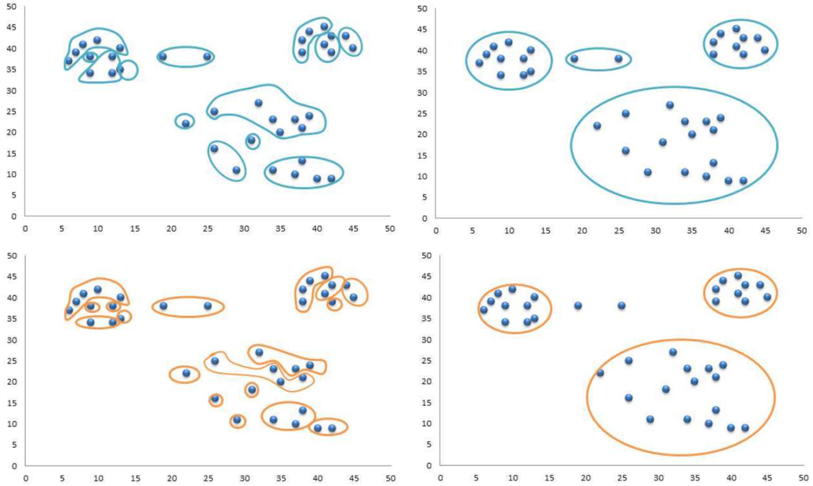


Fig. 3. Clustering effect on Spaeth dataset. CF-tree is obtained with B=L=2. Left: OLAPBIRCH without DBSCAN, Right: OLAPBIRCH with DBSCAN; Top: $LEVEL = 6$, Bottom: $LEVEL = 7$. Points outside clusters are considered outliers.

No of points	Running time (s)	Q	No of rebuilds
60×10^3	776	0.08	5
100×10^3	1819	0.07	5
500×10^3	41,646	0.018	5
1.1×10^6	172,800	0.039	9

Table 1. TPC-H: scalability results. MAX_LEV= 20, $B = L = 2$

ters. Moreover, by increasing the depth of the tree, it is possible to have more detailed clusters.

Results obtained on the TPC-H database are reported in Table 1. From these results, it is possible to see that the quality of the clusters does not deteriorate when the number of examples increases. We can also see that the number of times that the CF-tree is rebuilt is very small, even for huge datasets.

Figure 4 shows a different perspective of the obtained results. In particular, it shows that there is strong correlation between the region dimension (that is not considered during the clustering phase) and the obtained clusters. This means that numerical properties of the orders change in distribution between the regions where the order is performed.

4 Conclusions and Future Work

In this paper we have presented the framework OLAPBIRCH. This framework integrates a clustering algorithm in an OLAP engine in order to support roll-up and rill-down operations on numerical dimensions. OLAPBIRCH integrates a revised version of the BIRCH clustering algorithm that permits to incrementally revise hierarchical

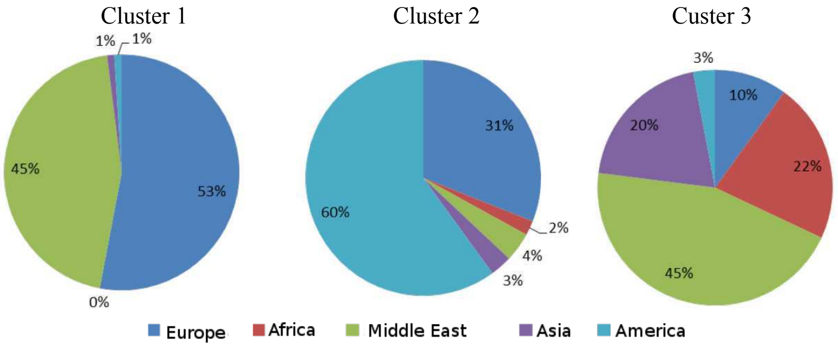


Fig. 4. TPC-H: Data distribution over the Region dimension.

clustering for all the levels of the hierarchy. Preliminary results show the effectiveness of the proposed solution on large real world datasets. For future work we intend to compare our framework with competitive frameworks and prove its applicability in answering to range queries and we intend to run experiments by varying input parameters in order to give better insights on their definition.

Acknowledgment

This work is partial fulfillment of the research objective of ATENEO-2009 project "Estrazione, Rappresentazione e Analisi di Dati Complessi".

References

1. A. Cuzzocrea. Improving range-sum query evaluation on data cubes via polynomial approximation. *Data Knowl. Eng.*, 56(2):85–121, 2006.
2. A. Cuzzocrea and P. Serafino. Clustcube: An olap-based framework for clustering and mining complex database objects. In *SAC*, 2011.
3. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *KDD*, pages 226–231, 1996.
4. J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub totals. *Data Min. Knowl. Discov.*, 1(1):29–53, 1997.
5. D. Gunopulos, G. Kollios, V. J. Tsotras, and C. Domeniconi. Selectivity estimators for multidimensional range queries over real attributes. *VLDB J.*, 14(2):137–154, 2005.
6. N. Karayannidis and T. K. Sellis. Hierarchical clustering for olap: the cube file approach. *VLDB J.*, 17(4):621–655, 2008.
7. J. Shanmugasundaram, U. M. Fayyad, and P. S. Bradley. Compressed data cubes for olap aggregate query approximation on continuous dimensions. In *KDD*, pages 223–232, 1999.
8. T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In H. V. Jagadish and I. S. Mumick, editors, *SIGMOD Conference*, pages 103–114. ACM Press, 1996.

Decomposition Technique for the Inverse Frequent Itemset Mining Problem

Antonella Guzzo[1], Luigi Moccia[2], Domenico Sacca[1], and Edoardo Serra[1]

DEIS[1], University of Calabria, 87036, Rende, Italy
ICAR-CNR[2], 87036, Rende,Italy
guzzo@deis.unical.it, moccia@icar.cnrt.it, sacca@unical.it, eserra@deis.unical.it

Abstract. The Inverse Frequent itemset Mining (IFM) is the problem of computing a transaction database D satisfying specified support constraints on a given set S of itemsets, that are typically the frequent ones. Earlier studies focused on investigating computational and approximability properties of this problem, that is NP-hard. However, this classical formulation of IFM does not enforce any constraint on the other itemsets (i.e., the ones that are not listed in S); D may therefore happen to contain additional (and, perhaps, unsuspected or even undesired) frequent itemsets. A possibility for removing this anomaly is to introduce a more general formulation of IFM in which the supports of itemsets that do not belong to S are explicitly constrained by a given threshold in order not to eventually get unexpected ”frequent” itemsets. This paper investigates this formulation, shows how it can be encoded as an integer linear program, and introduces a no-integer version of it solvable by a decomposition technique, that is a method designed to handle optimization problems with a huge number of variables by a using a limited memory space. As the decomposition technique requires at each step the solution of an auxiliary NP-hard integer linear program, a constructive heuristic for this auxiliary problem has also been defined, which enjoys very good scaling, thereby paving the way for its application over real-life scenarios.

1 Introduction

Transaction databases are databases where each tuple, called *transaction*, is defined as a subset of an underlying fixed set of *items* $\mathcal{I}$ (e.g. in market data, where a transaction corresponds to one purchase event and each item is a salable product). A popular mining task over transaction databases is to single out the set of the *frequent itemsets*, i.e., all the subsets of $\mathcal{I}$ (called *itemsets*) which are contained in a significant fraction (user-specified as a minimum support threshold) of the given transactions [7]. This problem attracted relevant research efforts in recent years, and several solution approaches and generalizations have indeed been discussed in the literature. In some cases, however, the perspective of the frequent itemset mining problem is naturally inverted: given a set S of frequent itemsets whose supports must be within specified ranges, the goal is to construct a transaction database D (if any) over which these itemsets are actually frequent. This problem, called *Inverse Frequent itemset Mining* (IFM) problem, has been recently introduced in the context of defining generators for benchmarks of mining algorithms [9, 6, 5], and has been subsequently reconsidered in privacy preserving contexts [10, 11], where the goal is to publish some data while avoiding disclosing sensitive or private knowledge. From a technical viewpoint, earlier studies on the inverse frequent itemset mining problem mainly investigated its computational properties, by charting a precise picture of the conditions under which it becomes intractable (see , e.g., [1, 2, 9]), and by observing that the problem is $\mathcal{NP}$-hard even if one looks for approximate solutions [10]. In this paper, the inverse frequent set mining problem is considered from a pragmatic point of view using the basic inverse frequent itemset mining setting of [9], where the "frequency" of any itemset in the database is measured in terms of its support, i.e., as the number of the transactions in which it occurs[1]. We first introduce an extension of IFM, calledσ-IFM, that removes the anomaly of the classical formulation that does not enforce any constraint on the other itemsets (i.e., the ones that are not listed in S) so that the computed transaction database D may contain additional (and, perhaps, unsuspected or even undesired) frequent itemsets. Thus, our formulation overcomes a severe limitation for a practical usage of IFM. We also concentrate on studying a heuristic solution approach that is able to scale over large instances, that often arise in real applications.

Within the extended framework for inverse frequent itemset mining discussed above, our contribution is then to study an efficient and effective solution approach based on linear programming. In more detail,

- We first propose a novel formulation of the inverse frequent set mining problem, σ-IFM, where the itemsets not in S are constrained to be infrequent – computation of the problem is still $\mathcal{NP}$-hard and, hence, unlikely to be efficiently solvable (*Section 2*).
- We describe how the σ-IFM can be encoded as an integer linear program (*Section 3.1*) and show that the relaxed version (i..e., non integer linear program) of it enjoys two important properties: (i) the size of the approximated solution is polynomial and (ii) accuracy of the solution can be measured in terms of input parameters (*Section 3.2*).
- We shown how the relaxed version of the σ-IFM can be solved by column generation which is a classical technique of large-scale linear programming. The proposed method requires at each step the solution of an auxiliary integer linear program which we prove being NP-hard. We devise an improved formulation and a constructive heuristic for this auxiliary problem (*Section 4*).
- Finally, we conduct a thorough experimental activity over both syntectic and real-life data. Results evidence that our approach emerged to be effective for real application scenarios. In addition, the results show that our approach enjoys very good scaling, thereby paving the way for its application over real-life scenarios (*Section 5*).

[1] Note that other approaches to the inverse frequent set mining problem (e.g., [1, 2]) considered the actual frequency, i.e., the support divided by the total number of transactions; however, as discussed in [9], supports convey more information than frequencies and hence the perspective of [9] is adopted here.

2 Problem Formalization

Let $\mathcal{I} = \{o_1, \ldots, o_n\}$ be a finite domain of elements, also called *items*. Any subset $I \subseteq \mathcal{I}$ is called an itemset over $\mathcal{I}$. The universe of itemsets $U_{\mathcal{I}}$ is the set of all non-empty itemsets over $\mathcal{I}$. A database $\mathcal{D}$ over $\mathcal{I}$ is bag of itemsets, each one usually called *transaction*. The number of transactions in $\mathcal{D}$ is denoted by $|\mathcal{D}|$. Given a database $\mathcal{D}$ over $\mathcal{I}$, for each itemset I over $\mathcal{I}$ ($I \in U_{\mathcal{I}}$), the *support* of I, denoted by $\sigma^{\mathcal{D}}(I)$, is the number of transactions containing I, and the number of duplicates of I, denoted by $\delta^{\mathcal{D}}(I)$, is the the number of transactions equal to I. We say that I is a *frequent* itemset in $\mathcal{D}$ if I's support is no less then a given support threshold *minsupp*. Finding all the frequent itemsets in $\mathcal{D}$ is the well-known *frequent itemset mining problem*. The *anti-monotonocity property* holds for supports: given two itemsets I and J with $I \subset J$, $\sigma^{\mathcal{D}}(J) \subseteq \sigma^{\mathcal{D}}(I)$.

We denote the set of natural numbers by $\mathbb{N}_0$ that will be used for bound. We also introduce the symbol ∞ to denote an unlimited bound and define $\mathbb{N}_0'$ as $\mathbb{N}_0 \cup \{\infty\}$ — we therefore assume that for each $i \in \mathbb{N}_0$, $i < \infty$ holds. Finally, we denote the set of pairs $\{(a, b) : a \in \mathbb{N}_0, b \in \mathbb{N}_0', a \leq b\}$ by $\mathbb{N}^2$.

Thanks to the above notation, we now formally introduce the σ-IFM problem:

Definition 1. Let:

i S be a given set of itemsets over the items in $\mathcal{I}$

ii S' and S'' denote respectively $\{I \in \mathcal{I} \setminus S : \exists I' \in S \text{ s.t. } I \subset I'\}$ and $U_{\mathcal{I}} \setminus (S \cup S')$

iii $\Gamma_\sigma = \{(I, \sigma^I_{min}, \sigma^I_{max}) : I \in S, (\sigma^I_{min}, \sigma^I_{max}) \in \mathbb{N}^2\}$ be a given set of triples assigning a minimum and the maximum support to each itemset in S

iv $\sigma'' \in \mathbb{N}_0'$ be the maximum support threshold for all itemsets in S''

v $\sigma' \in \mathbb{N}_0'$ be the maximum additional support threshold for all itemsets in S', i.e., for each $I \in S'$, the maximum support threshold of I is $\sigma' + \sum_{J \in S_I} \sigma^J_{max}$, where S_I is the set of all minimal itemsets in S containing I, i.e., $S_I = \{J \in S \mid I \subset J$ and $\nexists K \in S : K \subset J \wedge I \subset K\}$

vi **size** $= (size_1, size_2) \in \mathbb{N}^2$ denote the minimal and the maximal dimension for a database.

Then, the *σ-generic inverse frequent itemset mining problem* on $\mathcal{I}$, S, Γ_σ, σ'', σ' and **size**, shortly denoted as σ-IFM, consists of finding a database $\mathcal{D}$ over $\mathcal{I}$ such that the following conditions hold (or eventually state that there is no such a database):

$$\forall I \in S : \sigma^I_{min} \leq \sigma^{\mathcal{D}}(I) \leq \sigma^I_{max} \tag{1}$$

$$\forall I \in S'' : \sigma^{\mathcal{D}}(I) \leq \sigma'' \tag{2}$$

$$\forall I \in S' : \sigma^{\mathcal{D}}(I) \leq \sigma' + \textstyle\sum_{J \in S_I} \sigma^J_{max} \tag{3}$$

$$size_1 \leq |\mathcal{D}| \leq size_2. \tag{4}$$

If $\sigma' = \sigma" = \infty$, $size_1 = 0$ and $size_2 = \infty$ then the problem reduces to the *classical inverse frequent itemset mining problem*, IFM for short. □

The above constraints do formalize the intuitive points we have previously discussed. Indeed, the constraint (1) states that σ^I_{min} and σ^I_{max} induce a range of admissible supports for each itemset $I \in S$. The constraint (2) imposes that each intemset which neither belongs to the set S nor is a subset of some itemset in S must have a support $\sigma^{\mathcal{D}}(I)$ less or equal than the threshold σ''. By Constraint (3), itemsets in S' (i.e., subsets of some itemset in S) have support bounds that takes into account the ones of their immediate super-sets in S because of anti-monotonicity. Finally, the size of the database, i.e., the total number of transactions, is constrained in a range indicated by the constraint (4).

The complexity of the decision version of σ-IFM (*decision* σ-IFM) is $\mathcal{NP}$-hard.

Proposition 1 *Deciding whether there exists a solution for σ-IFM is $\mathcal{NP}$-hard and in $\mathcal{PSPACE}$.*

The proof is carried out by showing that σ-IFM is a special case of classical IFM, whose complexity has been analyzed in literature and has been proved to be $\mathcal{NP}$-hard and in $\mathcal{PSPACE}$ (see [9]). To this end, we introduce two meaningful subsets of S'' and S', called respectively $\mathtt{LUB}_{S''}$ and $\mathtt{LUB}_{S'}$, containing all least upper bounds of the itemsets in S:

$$\mathtt{LUB}_{S''} = \{I \cup \{x\} \mid I \in (S \cup \{\emptyset\}), x \in (\mathcal{I} - I)\} \cap S''. \tag{5}$$

$$\mathtt{LUB}_{S'} = \{I \cup \{x\} \mid I \in (S \cup \{\emptyset\}), x \in (\mathcal{I} - I)\} \cap S'. \tag{6}$$

Observe that the sum of the sizes of $\mathtt{LUB}_{S'}$ and $\mathtt{LUB}_{S''}$ is at most $(m+1) \times n$. As proven next, the generic support constraints can be tested just for the itemsets in $\mathtt{LUB}_{S''}$ and $\mathtt{LUB}_{S'}$ in rather than for all itemsets in S'' and S' respectively, whose number is instead exponential in n.

3 Linear Program Formulation of σ-IFM

3.1 The σ-IFM as an Integer Linear Program

The σ-IFM looks for a mapping between each itemset belonging to the universe $U_{\mathcal{I}}$ and the integers in $\{0, ..., |\mathcal{D}|\}$, which expresses the number of transactions of each itemset occurring in the resulting database $\mathcal{D}$ (complying with the constraints in Definition 1). We next encode this problem as an integer linear program.

W.l.o.g., we select any ordering of $U_{\mathcal{I}}$, say $\{I_1, \ldots, I_{2^n-1}\}$ and we assume that the input itemsets are the first $m \leq 2^n - 1$ ones in the above ordering, i.e., $S = \{I_1, \ldots, I_m\}$. Let $V = \{1, ..., m\}$ and $T = \{1, \ldots, 2^n - 1\}$.

Let l^σ and u^σ be two vectors respectively in $\mathbb{N}_0^m$ and in $\mathbb{N}_0'^m$ storing the lower and upper support constraints, i.e., $l_i^\sigma = \sigma_{min}^{I_i}$ and $u_i^\sigma = \sigma_{max}^{I_i}$, for each $1 \leq i \leq m$. Accordingly, we define l^δ and u^δ as two vectors respectively in $\mathbb{N}_0^m$ and in $\mathbb{N}_0'^m$ storing the lower and upper duplicate constraints, i.e., $l_i^\delta = \delta_{min}^{I_i}$ and $u_i^\delta = \delta_{max}^{I_i}$, for each $1 \leq i \leq m$.

We define u^δ as a vector in $\mathbb{N}_0'^n$ storing the generic upper duplicate constraints for each itemset size k, $1 \leq k \leq n$, i.e., $u_k^\delta = f_\delta^k$. Let $'\#'$ be a function that for each i, $1 \leq i \leq 2^n - 1$ returns the length of (i.e., the number of item in) I_i — obviously this function is computed in time linear in n, that is the size of $\mathcal{I}$, the maximal itemset.

W.l.o.g., we assume that the itemsets in $\texttt{LUB}_{S''}$ (see equation 5 above) are numbered from I_{m+1} to $I_{m+q''}$, where $q'' \leq (m+1) \times n$ and, then, $m + q'' \leq 2^n - 1$; we denote the indices $\{m+1, \ldots, m+q\}$ by Q''. Accordingly, we assume that the itemsets in $\texttt{LUB}_{S'}$ (see equation 6 above) are numbered from $I_{m+q''+1}$ to $I_{m+q''+q'}$, where $q' \leq (m+1) \times n$ and $m+q''+q' \leq 2^n-1$; we denote the indices $\{m+q''+1, \ldots, m+q''+q'\}$ by Q'. In addition, for each $i \in Q'$, u'_i denotes $\sum_{I_j, j \leq m, I_i \subset I_j} u^\sigma_j$.

Let x be a vector of $2^n - 1$ non-negative integer variables whose intended meaning is that its i-th coordinate, x_i, denotes the number of duplicates for the transaction I_i. Moreover, consider a $(2^n - 1) \times (2^n - 1)$ matrix A where each entry $a_{ij} \in 0, 1$ is associated with the pair of itemsets I_i and I_j with the intended meaning that $a_{ij} = 1$ if and only if $I_j \supseteq I_i$ holds.

We next introduce an objective function measuring the cost of violating any constraint using a number of real variables denoted by the symbol v with suitable adornments: the vectors v^l and v^u of $2^n - 1$ non-negative real variables, the vector v'' of q'' non-negative real variables, the vector v' of q' non-negative real variables and the two scalar non-negative real variables v^{s_1} and v^{s_2}. Then, the problem can be recast in terms of the following integer linear program:

$$\texttt{ILP}\text{: minimize} \sum_{i \in V} (v^l_i + v^u_i) + \sum_{i \in Q''} v''_i + \sum_{i \in Q'} v'_i + v^{s_1} + v^{s_2} \tag{7}$$

subject to

$$v^l_i + \sum_{j \in T} a_{ij} x_j \geq l^\sigma_i \qquad i \in V \tag{8}$$

$$v^u_i - \sum_{j \in T} a_{ij} x_j \geq -u^\sigma_i \qquad i \in V \tag{9}$$

$$v''_{i-m} - \sum_{j \in T} a_{ij} x_j \geq -\sigma'' \qquad i \in Q'' \tag{10}$$

$$v'_{i-m-q'} - \sum_{j \in T} a_{ij} x_j \geq -\sigma' - u'_i \qquad i \in Q' \tag{11}$$

$$v^{s_1} + \sum_{j \in T} x_j \geq size_1 \tag{12}$$

$$v^{s_2} - \sum_{j \in T} x_j \geq -size_2 \tag{13}$$

$$v^l_i, v^u_i \geq 0 \qquad i \in V, \tag{14}$$

$$v''_i \geq 0 \qquad i \in Q, \tag{15}$$

$$v^{s_1}, v^{s_2} \geq 0. \tag{16}$$

$$x_i \in \mathbb{N}_0 \qquad i \in T. \tag{17}$$

Whenever the bound of some of the constraints (9), (10), (11) and (13) is equal to ∞, the corresponding constraint is just removed. The non-negative real variables v^l_i, v^u_i ($\forall i \in V$), v''_i ($\forall i, 1 \leq i \leq q'$), v'_i ($\forall i, 1 \leq i \leq q''$), v^{s_1} and v^{s_2} are *artificial* in the sense that their role is to absorb possible violations of the constraints (8) – (13): the minimization of its value entails the search for a solution with the minimal number of violations. For example, assume that $\sum_{j \in T} x_j < size_1$, then the variable v will be forced by (12) to a value at least equal to $size_1 - \sum_{j \in T}$, i.e. a value greater than zero. The objective function (7) minimizes the value of the artificial variables. Therefore, the optimal solution of the presented ILP consists in a database (as induced by variables x in the optimal solution) with minimal violation of the mimimal database size constraint. Let $X_{\texttt{ILP}}^{\text{OPT}_0}$ be a solution of $\texttt{ILP}$ (if any) for which the objective function is equal to 0 and $D[X_{\texttt{ILP}}^{\text{OPT}_0}]$ be the database defined by such a solution.

The next result proves that the ILP defined by the objective function (7), and by the constraints (8) – (17) is instrumental in solving the σ-IFM.

Proposition 2 *A database D satisfies the σ-*IFM *problem if and only if there exists an optimal solution $X_{\texttt{ILP}}^{\text{OPT}_0}$ for which $D = D[X_{\texttt{ILP}}^{\text{OPT}_0}]$.*

Let $b[X_{\texttt{ILP}}^{\text{OPT}_0}]$ be the number of non-zero elements in $X_{\texttt{ILP}}^{\text{OPT}_0}$.

Corollary 3

*i Deciding whether there exists a solution $X_{\texttt{ILP}}^{\text{OPT}}$ for which the objective function is equal to 0 is in $\mathcal{NP}$ if and only decision σ-*IFM *is in $\mathcal{NP}$;*
*ii if decision σ-*IFM *$\notin \mathcal{NP}$ then $b[X_{\texttt{ILP}}^{\text{OPT}_0}]$ is exponential in n and m.*

As shown in the next subsection, a solution with a polynomial number of non-zero elements can be obtained by the continuous relaxation of the integer restrictions, i.e. all the variables are allowed to be real numbers.

3.2 The σ-IFM as a Linear Program

The relaxation of the σ-IFM can be stated as a linear program with an exponential number of variables but only a polynomial number of constraints:

$$\texttt{LP}\text{: minimize} \sum_{i \in V}(v_i^l + v_i^u) + \sum_{i \in Q''} v_i'' + \sum_{i \in Q'} v_i' + v^{s_1} + v^{s_2} \tag{18}$$

subject to the constraints (8) – (16) of the $\texttt{ILP}$ problem and by replacing the constraint (17) with;

$$x_i \in \mathbb{R} \;\; i \in T. \tag{19}$$

Let $X_{\text{LP}}^{\text{OPT}}$ be a solution of $\texttt{LP}$ (if any) and $X_{\text{LP}}^{\text{OPT}_0}$ be a solution for which the objective function is equal to 0.

Proposition 4 *Given $v \in \mathbb{R}$, deciding whether there exists a solution $X_{\text{LP}}^{\text{OPT}}$ for which the objective function is less than or equal to v is in $\mathcal{NP}$.*

Corollary 5 *Given any solution X_{LP}^{OPT}, $b[X_{LP}^{OPT}]$ is polynomial in n and m.*

Thus, relaxing the integer constraints allows us to find an approximate solution with polynomial size, that is given by the database $D[X_{\mathtt{LP}}^{\mathrm{OPT}}]$ induced by $\mathtt{round}[X_{\mathtt{LP}}^{\mathrm{OPT}}]$, that is obtained from $X_{\mathtt{LP}}^{\mathrm{OPT}}$ by rounding the values of its elements. An upper bound for the accuracy of the approximated solution is given next.

Proposition 6 *Given an optimal solution $X_{\mathtt{LP}}^{\mathrm{OPT}_0}$ of* LP, $\mathtt{round}[X_{\mathtt{LP}}^{\mathrm{OPT}}]$ *is a solution of* ILP *whose objective function value is less than or equal to $(2 \times m + q' + q'' + 2) \times (m + q' + q\text{''} + 1)/2 = \mathcal{O}(m^2 \times n)$.*

Experimental results prove that in practice the approximate solution is instead far below the theoretical bound.

The relaxation of integer constraints opens an interesting perspective for an effective solution of σ-IFM. But finding an optimal solution of LP is not an easy task because of the exponential number of variables. In the next section we present an algorithm which at each step explicits a polynomial number of variables.

4 Column Generation Algorithm to solve LP

Column generation, see e.g [3] and [4], is a method of dealing with linear programs with a large number of variables. This method solves a linear program without explicitly including all columns, i.e., variables, in the coefficient matrix. Columns are dynamically generated by solving an auxiliary optimization problem called the *pricing problem*. In the following we present and discuss the main column generation algorithm.

The linear program to be solved is denoted as the *master problem* (MP). The linear program with only a subset of the MP columns is called the *restricted master problem* (RMP). From linear programming theory we know that if there is an optimal solution it exists an optimal solution corresponding to a basis of the coefficient matrix. The columns of an optimal basis are those strictly necessary, all other columns can be ignored. Thus, to solve the MP is equivalent to solve the RMP with only the columns defining an optimal basis.

The column generation method looks for an optimal basis as within the simplex algorithm. The simplex algorithm moves from a current basis to a new basis by removing from the basis one column and adding to the basis a column with a negative (here and in the following we refer to the minimization case) reduced cost (*iteration step*). Primal feasibility is maintained and the objective function is non-increasing during this search. The reduced cost of a column can be computed by using the current dual variables. If there is not a column with a negative reduced cost, then the simplex algorithm terminates and the current basis is proved to be optimal. This scheme is maintained within the column generation method. However, the task of providing a column with a negative reduced cost, or certifying that there is not such a column, is delegated to the pricing problem.

The column generation algorithm has an attractive characteristic: columns that exit the basis can be dropped from the RMP. By so doing the algorithm has bounded use of the space, proportional to the number of constraints, i.e. the dimension of a basis.

Proposition 7 *At each iteration step the size of RMP is polynomial in the size of the input.*

Preserving polynomial size throughout the whole execution has some drawback on time efficiency. An acceleration strategy consists in maintaining the non-basic columns in the RMPs. This can avoids re-generating some of them. It should be noted that finite convergence is guaranteed as within the simplex algorithm even when the non-basic columns are dropped from the RMP, see [3]. In the following we simplify the presentation of the column generation algorithm by omitting the description of the space management scheme. The column generation algorithm critically depends upon solving the pricing problem. As we will show in a forthcoming paper, it is unlikely that a polynomial algorithm exists for the pricing problem. However, we do not need to solve exactly the pricing problem at each iteration. We could as well deploy a heuristic algorithm. As long as the heuristic algorithm returns a column with negative reduced cost we can iterate the column generation by adding the new column to the RMP.

In the general case, an exact solution of the pricing problem is necessary to the stop criterion of the column generation, i.e. when we have to certify that there is not a column with a negative reduced cost. In our case, we deal with an artificial optimization problem where the objective function (7) registers the distance from feasibility, since we know that the searched database $\mathcal{D}$ corresponds to an objective function value of zero. We have, thus, an alternative stop criterion: if the heuristic algorithm fails in finding a new column and we have reached a satisfactory, user defined, level of unfeasibility we can end the algorithm. This alternative stop criterion limits solving exactly the pricing problem which is computationally expensive. We need to call the exact pricing routine only if the heuristic fails and we have not reached the satisfactory level of unfeasibility.

The simplex algorithm, in the worst-case, explores all the feasible bases. The number of bases is exponential in the number of constraints and variables. In our problem the number of variables is itself exponential, leading to worrying worst-case conditions. We note that alternative algorithms for linear programming with polynomial worst-case complexity do exists, e.g interior points methods, see [3]. However, these worst-case complexities depend upon both the number of constraints and the number of variables. Since we deal with a linear program with an exponential number of variables we cannot escape worst-case exponential complexity by using these tools. As we will show in a forthcoming paper, the number of iterations of the column generation algorithm, i.e. the number of times that we update and re-solve the RMP, is such that the algorithm can have practical use. This is coherent with the practice were the simplex algorithm shows a remarkably fast convergence. However, in terms of worst-case computational complexity we cannot rule out excessive computational times. Therefore, we have to provide a global time-limit for termination. Moreover, the exact pricing routine could also face excessive computational times. A specific time-limit for the exact pricing routine is hence provided.

In summary, the decomposition algorithm starts by initializing the RMP. We observe that populating the RMP with the artificial variables v suffices to RMP feasibility. However, this RMP will provide poor dual information. We then initialize the RMP by adding also the variables x_i with $1 \leq i \leq m$, which are likely to have strictly positive values in the optimal solution. Based on the RMP dual variables , we solve the pricing problem and if a column with a negative reduced cost has been founded, that column will be included in the RMP problem and reiterate the procedure. In our approach, we try to first solve heuristically the pricing problem, and if no column is provided, we use the the exact pricing routine. The algorithm stopped when a the global time-limit has been reached, or when the decomposition algorithm does not include any column with negative reduced in RMP, and the the current solution is hence optimal.

5 Experiments and Conclusion

The solution approach for σ-IFM described in the previous section has been implemented, and a thorough experiential activity has been conducted to assess its efficiency and effectiveness over two real datasets BMS-WebView-1 and BMS-WebView-2, which can be download from the KDD-Cup 2000 competition website [8].

Due to the space limitation, all the results will not be addressed here, while we focused on the efficiency and the accuracy of our approach by reporting the execution times and a comparison with the IPF method proposed in [11]. Specifically, in the experimentation perspective discussed in [11], it is argued that the effectiveness of inverse frequent itemsets mining algorithms has to be assessed by comparing the similarity between the synthetic dataset generated as solution of the inverse problem and the original one. We mention that the correspondent synthetic datasets for BMS-WebView-1 and BMS-WebView-2 are generated with the frequent itemsets computed on real dataset whit threshold supports 0.7 and 0.6, respectively.

To this aims, we used the *Jaccard*, *Dice*, and *Overlap* indexes (see [11] for more details) to compare similarities between original frequent itemsets and those occurring in the output dataset. Results on the two real input datasets are reported in Figure 1. The figure evidences that very high accuracy measures are obtained by our method.

The table of execution times, reported in Figure 2, shows that our method has reduced execution times for a large amount of constraints, measured in terms of the cardinality of S ($|S|$) and of Q'' ($|Q''|$) - note that we have set $|Q'| = 0$. As a large number of constraints generates an exponential number of variables (from 10^{22} to over 10^{240} in our experiments!), the method allows us to handle linear programs with an enormous number of variables using a reduced amount of space and time. Limitation on space can be guaranteed in all cases whereas exponential time is not. However, the actual performance of the method in our experiments suggests that exponential time is get in rare cases as it happens for the classical execution of the simplex algorithm.

References

1. Calders, T.: Computational complexity of itemset frequency satisfiability. In: in Proc. PODS04. pp. 143–154 (2004)
2. Calders, T.: Itemset frequency satisfiability: Complexity and axiomatization. Theor. Comput. Sci. 394(1-2), 84–111 (2008)

BMS-Web	support(%)	Jaccard		Dice		Overlap	
		σ-IFM	IPF	σ-IFM	IPF	σ-IFM	IPF
View-1 s=0.7	0.6	0.535	0.817	0.697	0.899	0.747	0.985
	0.7	1.0	0.940	1.0	0.969	1.0	0.992
	0.8	1.0	0.883	1.0	0.938	1.0	0.934
	0.9	1.0	0.893	1.0	0.944	1.0	0.954
	1.0	1.0	0.887	1.0	0.940	1.0	0.959
View-2 s=0.6	0.6	1.0	0.696	1.0	0.768	1.0	1.0
	0.7	0.994	0.708	0.997	0.739	1.0	0.964
	0.8	1.0	0.710	1.0	0.830	1.0	0.928
	0.9	0.991	0.722	0.995	0.838	1.0	0.976
	1.0	1.0	0.701	1.0	0.824	1.0	0.910

Fig. 1. *Comparison of σ-IFM accuracy vs. IPF method*

	BMS-WebView-1			**BMS-WebView-2**		
support(%)	$\lvert S\rvert$	$\lvert Q''\rvert$	time(s)	$\lvert S\rvert$	$\lvert Q''\rvert$	time(s)
0.2	798	36506	137.553	3683	167688	10923.81
0.3	435	25530	7.157	1340	61083	5266.576
0.4	286	16605	1.037	676	30343	58.588
0.5	201	11498	0.605	408	17642	6.939
0.6	162	8732	0.406	257	11255	1.515
0.7	133	6260	0.234	187	8614	0.625
0.8	105	4226	0.143	138	6927	0.296
0.9	90	3260	0.104	113	6051	0.221
1	77	2633	0.085	81	4827	0.109

Fig. 2. *Execution Times of σ-IFM*

3. Dantzig, G.B., Thapa, M.N.: Linear Programming 2: Theory and Extensions. Springer Series in Operations Research, Springer-Verlag, New York (2003)
4. Desaulniers, G., Desrosiers, J., Solomon, M.M. (eds.): Column Generation. Springer (2005)
5. Guo, Y., Tong, Y., Tang, S., Yang, D.: A fp-tree-based method for inverse frequent set mining. BNCOD 2006, LNCS 4042 pp. 152–163 (2006)
6. Guzzo, A., Saccà, D., Serra, E.: An effective approach to inverse frequent set mining. In: Proc. of the Int. IEEE Conf. on Data Mining (ICDM'09). pp. 806–811 (2009)
7. Han, J., Pei, J., Yi, Y.: Mining frequent patterns without candidate generation. In: Proc. Int. ACM Conf. on Management of Data (SIGMOD'00). pp. 1–12 (2000)
8. KDDCUP2000: http://www.ecn.purdue.edu/kddcup
9. Mielikainen, T.: On inverse frequent set mining. In: Society, I.C. (ed.) Proc. of 2nd Workshop on Privacy Preserving Data Mining (PPDM). pp. 18–23 (2003)
10. Wang, Y., Wu, X.: Approximate inverse frequent itemset mining: Privacy, complexity, and approximation. In: ICDM. pp. 482–489 (2005)
11. Wu, X., Wu, Y., Wang, Y., Li, Y.: Privacy-aware market basket data set generation: An feasible approach for inverse frequent set mining. In: Proc. 5th SIAM ICDM (2005)

Computing Multidimensional OLAP Data Cubes over Probabilistic Relational Data: A Decomposition Approach

Alfredo Cuzzocrea[1], Dimitrios Gunopulos[2], Saverio Manti[3]

[1] ICAR-CNR and University of Calabria, Italy
[2] Dept. of Informatics and Telecommunications
University of Athens, Greece
[3] DEIS Dept., University of Calabria, Italy
cuzzocrea@si.deis.unical.it, dg@di.uoa.gr, smanti@deis.unical.it

Abstract. Focusing on novel database application scenarios, where datasets arise more and more in *uncertain* and *imprecise* formats, in this paper we propose *a novel framework for efficiently computing multidimensional OLAP data cubes over probabilistic data*, which well-capture previous kinds of data. Several models and algorithms supported in our proposed framework are formally presented and described in details, based on well-understood *theoretical statistical/probabilistic tools*, which converge to the definition of the so-called *probabilistic OLAP data cubes*, the most prominent result of our research.

1 Introduction

Multidimensional OLAP data cubes [15] are powerful tools allowing us to support rich and multi-perspective analysis over large amounts of data sets, based on a multi-dimensional and multi-resolution vision of data. Efficiently computing OLAP data cubes over the input dataset (e.g., relational databases) is a well-known research challenge that has been deeply investigated during last decades (e.g., [17,1]), with alternate fortune. *Probabilistic data* (e.g., [3,8,12,13,19,4,2,20]) are becoming one of the most attracting kinds of data for database researchers, due to the fact such a format/formalism perfectly captures two novel, interesting classes of datasets that very often occur in modern database application scenarios, namely *uncertain* and *imprecise data*. Uncertain and imprecise data are indeed very popular, as uncertainty and imprecision affect the same processes devoted to collecting data from input data sources and make use of these data in order to populate the target database, like, for instance, in *sensory databases* [5].

Since probabilistic datasets are becoming very popular, it is natural and reasonable to define and introduce the problem of *efficiently computing OLAP data cubes over probabilistic data*, which has been firstly proposed in [5]. Basically, [5] introduces the *possible-world semantics* concept, according to which a probabilistic database D_P can be represented and processed (e.g., during query evaluation) via admitting the existence of *different possible-world databases* $D_{P,k}$. Each possible-world database $D_{P,k}$ is obtained from D_P via assigning *possible values* to probabilistic attributes in D_P, on the basis of confidence intervals and probabilities associated to these attributes. It would be clear enough to notice that this approach takes the risk of generating an *exponential* number of possible-world databases (i.e., $k \rightarrow e^{|D_P|}$, such that $|D_P|$ denotes the cardinality of D_P), even because a *combinatory dependence* among probabilistic attributes of D_P exists. Based on this model for representing probabilistic databases, [5] finally retrieves the output data cube over D_P, denoted by $C(D_P)$, via *estimating* aggregates (to be stored within data cube cells of $C(D_P)$) from the universe of possible-world databases derived from D_P by means of probabilistic/statistical *estimation tools and techniques* [18] and via detecting several *probability-inspired consistency conditions* [5].

While there is a wide and rich literature on the issue of *efficiently processing probabilistic databases* (e.g., [2,3,4,8,12,13,19,20]), despite the relevance of OLAP applications for next-generation *Data Warehousing* (DW) *and Business Intelligence* (BI) *systems* very few papers address at now the yet-interesting problem of computing OLAP data cubes over probabilistic data (e.g., [5,6,7]). Contrary to this actual trend, it is natural to foresee that this problem will play more and more a leading role in the context of DW and BI systems, due to the obvious popularity of uncertain and imprecise datasets (e.g., environmental sensor networks, data stream management systems, alarm and surveillance systems, RFID-based applications, supply-chain management systems).

Inspired by these motivations, starting from limitations of [5] in this paper we propose a framework for efficiently computing multidimensional OLAP data cubes over probabilistic data, which we name as *probabilistic data cubes*. The most distinctive contribution of our research is represented by a meaningful *decomposition approach* that allows us to extract the so-called *decomposed probabilistic database* from the input probabilistic database at the cost of a *sub-linear complexity* – the decomposed probabilistic database is then used to compute the final probabilistic data cube based on conventional *multidimensional aggregation methods* [15].

2 Modeling Probabilistic OLAP Data Cubes

Given a probabilistic database D_P modeled in terms of a collection of probabilistic relations R_i, i.e. $D_P = \{R_0, R_1, \ldots, R_{|D_P|-1}\}$, the problem we investigate in this research consists in effectively and efficiently computing a data cube over D_P, $C(D_P)$, given an input *data cube schema* [21] W. According to the nature of D_P, we properly define $C(D_P)$ as a probabilistic data cube (we detail this novel definition next). Now, focus the attention on the class of probabilistic databases considered in our research, which is inspired from fundamental works in [4,2,20]. Given a probabilistic relation R_i in D_P modeled in terms of a collection of attributes, i.e. $R_i = \{A_{i,0}, A_{i,1}, \ldots, A_{i,|R_i|-1}\}$, such that A_{i,k_j}, with k_j in {0, 1, …, $|R_i| - 1$}, denotes an attribute in R_i, two distinct sub-set of attributes in R_i can be identified. The first one, denoted by $R_i^E \subset R_i$, such that $R_i^E = \{A_{i,k_0}^E, A_{i,k_1}^E, \ldots, A_{i,k_{|R_i^E|-1}}^E\}$, with k_j in {0, 1, …, $|R_i| - 1$}, stores the sub-set of *exact attributes* in R_i, i.e. attributes in R_i whose values are exact. The second one, denoted by $R_i^P \subset R_i$, such that $R_i^P = \{A_{i,k_0}^P, A_{i,k_1}^P, \ldots, A_{i,k_{|R_i^P|-1}}^P\}$, with k_j in {0, 1, …, $|R_i| - 1$} stores the sub-set of probabilistic attributes in R_i, i.e. attributes in R_i whose values are probabilistic. Obviously, $R_i^E \cap R_i^P = \emptyset$. An exact attribute A_{i,k_j}^E in R_i^E is defined as follows: $A_{i,k_j}^E = \left\{V_{i,k_j}^E \middle| V_{i,k_j}^E \in \mathbb{D}_{i,k_j}^E\right\}$, where V_{i,k_j}^E denotes an exact value of A_{i,k_j}^E and $\mathbb{D}_{i,k_j}^E$ the domain of A_{i,k_j}^E, respectively. A probabilistic attribute A_{i,k_j}^P in R_i^P is defined as follows:

$$A_{i,k_j}^P = \left\{\langle [V_{i,k_{j,min}}^P, V_{i,k_{j,max}}^P], p_{i,k_j}\rangle \middle| V_{i,k_{j,min}}^P \in \mathbb{D}_{i,k_j}^P, V_{i,k_{j,max}}^P \in \mathbb{D}_{i,k_j}^P, V_{i,k_{j,min}}^P < V_{i,k_{j,max}}^P, 0 \le p_{i,k_j} \le 1\right\} \quad \textbf{(1)}$$

such that: (*i*) $V^P_{i,k_{j,min}}$ and $V^P_{i,k_{j,max}}$ denote probabilistic values of A^P_{i,k_j}, respectively; (*ii*) $[V^P_{i,k_{j,min}}, V^P_{i,k_{j,max}}]$ denotes the confidence interval associated to A^P_{i,k_j}; (*iii*) p_{i,k_j} denotes the probability associated to $[V^P_{i,k_{j,min}}, V^P_{i,k_{j,max}}]$; (*iv*) $\mathbb{D}^P_{i,k_j}$ denotes the domain of A^P_{i,k_j}. The semantics of this confidence-interval-based model states that the possible value of A^P_{i,k_j} ranges between $V^P_{i,k_{j,min}}$ and $V^P_{i,k_{j,max}}$ with probability p_{i,k_j}. Also, a law describing the *probability distribution* according to which possible values of A^P_{i,k_j} vary over the interval $[V^P_{i,k_{j,min}}, V^P_{i,k_{j,max}}]$ is assumed. Without loss of generality, the *Uniform distribution* [9] is very often taken as reference. The Uniform distribution states that (possible) values in $[V^P_{i,k_{j,min}}, V^P_{i,k_{j,max}}]$ have *all* the same probability of being the exact value of A^P_{i,k_j}, denoted by V_{i,k_j}, actually, i.e.:

$$P\left(A^P_{i,k_j} = V_{i,k_j}\right) = P\left(A^P_{i,k_{j+1}} = V_{i,k_j}\right) = \frac{p_{i.k_j}}{\left|V^P_{i,k_{j,max}} - V^P_{i,k_{j,min}}\right|} \quad \mathbf{(2)}$$
$$\forall\, k_j, k_{j+1}\colon k_j \neq k_{j+1}, k_j \in \{0,1,\dots,|R^P_i| - 1\}, k_{j+1} \in \{0,1,\dots,|R^P_i| - 1\}$$

Despite the popularity of the Uniform distribution, the confidence-interval-based model above is prone to incorporate any other kind of state-of-the-art probability distribution [18].

Given a data cube C, the data cube schema of C, W, is defined as the tuple: $W = \langle D, H, M\rangle$, such that [15]: (*i*) $D = \{d_0, d_1, \dots, d_{|D|-1}\}$ denotes the set of dimensions of C; (*ii*) $H = \{h_0, h_1, \dots, h_{|H|-1}\}$ denotes the set of hierarchies of C, being h_i in H the hierarchy associated to the dimension d_i in D; (*iii*) M denotes the set of measures of C. As directly related to data cubes measures, a SQL aggregation operator (e.g., SUM, COUNT, AVG) is set as the baseline operation of the aggregation process generating data cube cells [15]. Without loss of generality, in this paper we consider as reference the class of SUM-based data cubes, which is general enough to capture a wide spectrum of real-life DW and BI applications [11]. On the other hand, other SQL aggregation operators can be easily derived from this baseline one (e.g., COUNT can be derived from SUM combined with AVG). In addition to this, as regards the specific in-memory data representation, we refer to *MOLAP data cubes* [16], i.e. data cubes represented in terms of *multidimensional arrays*, a general format to which any alternative in-memory data cube representation technique (e.g., ROLAP, HOLAP) may converge easily [16]. Also, for the sake of simplicity, we hereby assume of dealing with single-measure data cubes, i.e. $M = \{m_0\}$, and discard the case of *multiple-measure data cubes* [14], i.e. M such that $|M| > 1$, as the latter case can be straightforwardly obtained via extending actual models and algorithms provided for single-measure data cubes. Dimensions $d_0, d_1, \dots, d_{|D|-1}$ in D and the measure m_0 in M are defined from attributes in D_P, meaning that the DW administrator assigns the role of dimension to a partition of attributes in D_P whereas he/she assigns the role of measure to one attribute in D_P. This is what is commonly intended as a *multidimensional abstraction of relational data* [15]. For the sake of simplicity, in our research we assume that the measure m_0 is chosen among exact attributes in D_P, i.e. $m_0 \in R^E_i \subset R_i$, such that $R_i \in D_P$. Contrary to the constraint on the measure m_0, dimensions in D can instead be chosen among both exact and probabilistic attributes in D_P, i.e. $d_i \in R^E_i \subset R_i$ or $d_i \in R^P_i \subset R_i$, such that $R_i \in D_P$.

Given a probabilistic data cube C, each cell of C, denoted by $c(i_0, i_1, \dots, i_{|D|-1}) = C[i_0][i_1]\dots[i_{|D|-1}]$, such that $i_0 \in \{0, 1, \dots, |d_0| - 1\}$, $i_1 \in \{0, 1, \dots, |d_1| - 1\}$, …, $i_{|D|-1} \in \{0, 1, \dots, |d_{|D|-1}| - 1\}$, is probabilistic in nature, according to D_P. Formally, $c(i_0, i_1, \dots, i_{|D|-1}) =$

$\langle[B_{min}^{c(i_0..i_{|D|-1})}, B_{max}^{c(i_0..i_{|D|-1})}], p_{c(i_0..i_{|D|-1})}\rangle$, such that: (*i*) $[B_{min}^{c(i_0..i_{|D|-1})}, B_{max}^{c(i_0..i_{|D|-1})}]$ denotes the confidence interval (over *aggregate values*) associated to $c(i_0, i_1, \ldots, i_{|D|-1})$, with $B_{min}^{c(i_0..i_{|D|-1})} < B_{max}^{c(i_0..i_{|D|-1})}$; (*ii*) $p_{c(i_0..i_{|D|-1})}$ denotes the probability associated to $[B_{min}^{c(i_0..i_{|D|-1})}, B_{max}^{c(i_0..i_{|D|-1})}]$, with $0 \leq p_{c(i_0..i_{|D|-1})} \leq 1$. As a first result, it should be noted that our notion of probabilistic data cube is novel under the classical notion proposed in [5], which aims at outputting exact data cubes.

3 Computing Probabilistic OLAP Data Cubes

3.1 Preliminaries

In this Section, we provide models and algorithms for computing probabilistic OLAP data cubes from probabilistic databases, as one of the most prominent contribution of our research. Given a probabilistic database $D_P = \{R_0, R_1, \ldots, R_{|D_P|-1}\}$, and a data cube schema $W = \langle D, H, M\rangle$, the probabilistic data cube $C(D_P)$ over D_P is defined by means of a *multidimensional mapping scheme* $\mathbb{M}_{D_P \to C(D_P)} = \langle \mathbb{M}_D, \mathbb{M}_H, \mathbb{M}_M\rangle$, such that: (*i*) $\mathbb{M}_D\colon\{R_0, \ldots, R_{|D_P|-1}\} \to \{d_0, \ldots, d_{|D|-1}\}$ represents a mapping (sub-)scheme between attributes in D_P and dimensions in $D \in W$, such that $A_{i,k_j} \to d_{h_l}$ denotes a mapping between an attribute A_{i,k_j} of a relation R_i in D_P and a dimension d_{h_l} in D, with k_j in $\{0, 1, \ldots, |R_i| - 1\}$ and h_l in $\{0, 1, \ldots, |D| - 1\}$; (*ii*) $\mathbb{M}_H\colon\{R_0, \ldots, R_{|D_P|-1}\} \to \{h_0, \ldots, h_{|H|-1}\}$ represents a mapping (sub-)scheme between attributes in D_P and hierarchies in $H \in W$, such that $A_{i,k_j} \to h_{v_m}$ denotes a mapping between an attribute A_{i,k_j} of a relation R_i in D_P and a hierarchy h_{v_m} in H, with k_j in $\{0, 1, \ldots, |R_i| - 1\}$ and v_m in $\{0, 1, \ldots, |H| - 1\}$; (*iii*) $\mathbb{M}_M\colon\{R_0, \ldots, R_{|D_P|-1}\} \to \{m_0\}$ represents a mapping (sub-)scheme between an attribute A_{i,k_j} of a relation R_i in D_P and the measure m_0 in $M \in W$ (from Sect. 2, recall that $|M| = 1$), denoted by $A_{i,k_j} \to m_0$, with k_j in $\{0, 1, \ldots, |R_i| - 1\}$. For the sake of simplicity, in this research we investigate the case of computing probabilistic data cubes such that $\mathbb{M}_H = \emptyset$, which determines the special case of data cubes without hierarchies. It should be noted that such data cubes are general enough to capture a wide family of real-life applications, as clearly highlighted in [10]. On the other hand, models and algorithms presented in our paper can be easily extended as to deal with more significant hierarchy-equipped data cubes.

Before presenting our proposed approach for computing probabilistic data cubes, we need to recall a well-known concept in *Data Mining* (DM), i.e. *discretization of relational database attributes* (e.g., [16]). Given a database D and an attribute A_{i,k_j} of a relation R_i in D, such that k_j in $\{0, 1, \ldots, |R_i| - 1\}$, having $\mathbb{D}_{i,k_j}$ as domain, the *discretized domain* of A_{i,k_j}, denoted by $\mathcal{D}(A_{i,k_j}) = \{\rho_{i,k_j,0}, \rho_{i,k_j,1}, \ldots, \rho_{i,k_j,|\mathcal{D}(A_{i,k_j})|-1}\}$, is defined as a collection of *discretized members* $\rho_{i,k_j,h}$ derived from $\mathbb{D}_{i,k_j}$ on the basis of the nature of A_{i,k_j}. If A_{i,k_j} is numeric in nature, then $\mathcal{D}(A_{i,k_j})$ is composed by all the numeric members extracted from the domain that is in a *bijective relation* with $\mathbb{D}_{i,k_j}$ starting from the minimum value in A_{i,k_j}, denoted by V_{i,k_j}^{MIN}, to the maximum value in A_{i,k_j}, denoted by V_{i,k_j}^{MAX}. For instance, if A_{i,k_j} stores positive integer values, then $\mathbb{D}_{i,k_j}$ is in a bijective relation with the domain of natural numbers $\mathbb{N}$ and $\mathcal{D}\left(A_{i,k_j}\right) =$

$\left\{V_{i,k_j}^{MIN}, V_{i,k_j}^{MIN}+1, \ldots, V_{i,k_j}^{MAX}-1, V_{i,k_j}^{MAX}\right\}$. If A_{i,k_j} is categorical in nature, then a *topological ordering relation* $\prec$ over categorical members in $\mathbb{D}_{i,k_j}$ must be introduced. Note that, in the previous example, $\prec \equiv <$. On the basis of the ordering determined by $\prec$, $\mathcal{D}(A_{i,k_j})$ is composed by all the categorical members in $\mathbb{D}_{i,k_j}$ starting from the "minimum" member in A_{i,k_j}, V_{i,k_j}^{MIN}, to the "maximum" member in A_{i,k_j}, V_{i,k_j}^{MAX}. For instance, if A_{i,k_j} stores the week days, then $\mathcal{D}\left(A_{i,k_j}\right) = \{Sunday, Monday, \ldots, Saturday\}$ (e.g., $Sunday \prec Monday$).

On the basis of the formal DM framework above, given a database D and an attribute A_{i,k_j} of a relation R_i in D, such that k_j in {0, 1, …, $|R_i| - 1$}, having $\mathbb{D}_{i,k_j}$ as domain and $\prec$ as topological ordering relation, we introduce the function nextM that takes as input A_{i,k_j}, $\mathbb{D}_{i,k_j}$, $\prec$ and a member V_{i,k_j} in $\mathcal{D}(A_{i,k_j})$, and returns as output the member V_{i,k_j}^{SUC} that "follows" V_{i,k_j} in $\mathcal{D}(A_{i,k_j})$ based on the ordering determined by $\prec$. Formally, nextM$(A_{i,k_j}, \mathbb{D}_{i,k_j}, \prec, V_{i,k_j}) = V_{i,k_j}^{SUC}$, such that $V_{i,k_j} \prec V_{i,k_j}^{SUC}$.

Now, focus the attention on how the probabilistic data cube $C(D_P)$ is finally computed from the input probabilistic database D_P and data cube schema W. Let $D = \left\{d_0, \ldots, d_{|D|-1}\right\} = \left\{A_{i,k_0}^E, A_{i,k_1}^E, \ldots, A_{i,k_{|E|-1}}^E, A_{i,k_{|E|}}^P, A_{i,k_{|E|+1}}^P, \ldots, A_{i,k_{|P|-|E|-1}}^P\right\}$ be the set of $|D| = |P| - |E|$ dimensions in W, selected from $|E|$ exact attributes and $|P|$ probabilistic attributes in D_P (see Sect. 2), respectively, such that (*i*) A_{i,k_j}^E, with k_j in {0, 1, …, $|R_i| - 1$}, being R_i a relation in D_P, denotes an exact attribute in D_P and (*ii*) A_{i,k_j}^P, with k_j in {0, 1, …, $|R_i| - 1$}, being R_i a relation in D_P, denotes a probabilistic attribute in D_P. Let D^E denote the set of exact attributes/dimensions in D and D^P the set of probabilistic attributes/dimensions in D, respectively ($D^E \cap D^P = \varnothing$). Let $M = \{m_0\} = \{\hat{A}_{i,k_j}^E\}$ be the singleton measure in W, such that $\hat{A}_{i,k_j}^E$, with k_j in {0, 1, …, $|R_i| - 1$}, being R_i a relation in D_P, denotes an exact attribute in D_P (from Sect. 2, recall that, in our probabilistic data cube model, measures are always chosen among exact attributes in D_P).

3.2 A Decomposition Approach for Computing Probabilistic Data Cubes

The approach for computing $C(D_P)$ we propose is two-step in nature. In the first step, the so-called decomposed probabilistic database, denoted by $D_P^{\triangleright}$, is obtained from D_P directly by means of an innovative *decomposition technique* whose main aim consists in *breaking confidence intervals of probabilistic attribute values, while preserving the probabilistic nature of tuples in D_P*. In the second step, $C(D_P)$ is finally computed from $D_P^{\triangleright}$ based on conventional multidimensional aggregation methods [15].

First, we describe the proposed approach for extracting $D_P^{\triangleright}$ from D_P. Basically, for each *dimensional attribute* A_{i,k_j} of relations R_i in D_P, in the first step we decompose attribute values V_{i,k_j} of A_{i,k_j} into a set of *decomposed attribute values*, denoted by $\mathcal{P}(A_{i,k_j})$, depending on the nature of A_{i,k_j} (exact, or probabilistic) and by exploiting the *multidimensional relation* with the corresponding attribute value $\hat{V}_{i,k_j}$ of the *measure attribute* $\hat{A}_{i,k_j}$ of relations R_i in D_P. Then, decomposed attribute value sets $\mathcal{P}(A_{i,k_j})$ are used to populate $D_P^{\triangleright}$ and, finally, $C(D_P)$ is aggregated from $D_P^{\triangleright}$ directly, based on the input data cube schema W. It should be noted that, depending on W, since OLAP dimensions and measures are typically a *sub-set* of the whole attribute set of the input database [15], without loss of generality we observe that $|D_P^{\triangleright}| < |D_P|$. This nice amenity contributes to lower the *spatial complexity* of the approach we propose.

Let: (*i*) $d_i \equiv A^E_{i,k_j}$ be an exact dimension in W; (*ii*) $m_0 \equiv \hat{A}^E_{i,k_j}$ be the exact measure in W; (*iii*) $A^E_{i,k_j}[\ell] = V^E_{i,k_j}$ be an exact value of A^E_{i,k_j} at position ℓ, such that $0 \le \ell \le \left|\mathbb{D}^E_{i,k_j}\right| - 1$, being $\mathbb{D}^E_{i,k_j}$ the domain of A^E_{i,k_j}; (*iv*) $\hat{A}^E_{i,k_j}[\ell] = \hat{V}^E_{i,k_j}$ be the corresponding exact value of $\hat{A}^E_{i,k_j}$ at position ℓ, such that $0 \le \ell \le \left|\widehat{\mathbb{D}}^E_{i,k_j}\right| - 1$, being $\widehat{\mathbb{D}}^E_{i,k_j}$ the domain of $\hat{A}^E_{i,k_j}$. Since d_i is exact, decomposing the pair $\langle d_i, m_0\rangle \equiv \langle A^E_{i,k_j}, \hat{A}^E_{i,k_j}\rangle$ generates the following set of decomposed attribute values $\mathcal{P}\left(A^E_{i,k_j}\right) = \left\{\langle V^E_{i,k_j}, 1, \hat{V}^E_{i,k_j}\rangle\right\}$. Note that $\left|\mathcal{P}\left(A^E_{i,k_j}\right)\right| = 1$. Contrary to the latter case, let: (*i*) $d_i \equiv A^P_{i,k_j}$ be a probabilistic dimension in W; (*ii*) $m_0 \equiv \hat{A}^E_{i,k_j}$ be the exact measure in W; (*iii*) $A^P_{i,k_j}[\ell] = \langle \left[V^P_{i,k_{j,min}}, V^P_{i,k_{j,max}}\right], p_{i,k_j}\rangle$ be a probabilistic value of A^P_{i,k_j} at position ℓ, such that $0 \le \ell \le \left|\mathbb{D}^P_{i,k_j}\right| - 1$, being $\mathbb{D}^E_{i,k_j}$ the domain of A^P_{i,k_j}; (*iv*) $\hat{A}^E_{i,k_j}[\ell] = \hat{V}^E_{i,k_j}$ be the corresponding exact value of $\hat{A}^E_{i,k_j}$ at position ℓ, such that $0 \le \ell \le \left|\widehat{\mathbb{D}}^E_{i,k_j}\right| - 1$, being $\widehat{\mathbb{D}}^E_{i,k_j}$ the domain of $\hat{A}^E_{i,k_j}$. Since d_i is probabilistic, decomposing the pair $\langle d_i, m_0\rangle \equiv \langle A^P_{i,k_j}, \hat{A}^E_{i,k_j}\rangle$ generates the following set of decomposed attribute values:

$$\begin{aligned}\mathcal{P}\left(A^P_{i,k_j}\right) = \; & \left\{ \langle V^P_{i,k_{j,min}}, \frac{p_{i,k_j}}{\left|V^P_{i,k_{j,max}} - V^P_{i,k_{j,min}}\right|}, \frac{\hat{V}^E_{i,k_j}}{\left|V^P_{i,k_{j,max}} - V^P_{i,k_{j,min}}\right|}\rangle, \right. \\ & \langle \mathrm{nextM}\left(A^P_{i,k_j}, \mathbb{D}^E_{i,k_j}, \prec, V^P_{i,k_{j,min}}\right), \frac{p_{i,k_j}}{\left|V^P_{i,k_{j,max}} - V^P_{i,k_{j,min}}\right|}, \\ & \frac{\hat{V}^E_{i,k_j}}{\left|V^P_{i,k_{j,max}} - V^P_{i,k_{j,min}}\right|}\rangle, \\ & \ldots, \\ & \left. \langle V^P_{i,k_{j,max}}, \frac{p_{i,k_j}}{\left|V^P_{i,k_{j,max}} - V^P_{i,k_{j,min}}\right|}, \frac{\hat{V}^E_{i,k_j}}{\left|V^P_{i,k_{j,max}} - V^P_{i,k_{j,min}}\right|}\rangle \right\}\end{aligned} \qquad \textbf{(3)}$$

Note that $\left|\mathcal{P}\left(A^P_{i,k_j}\right)\right| = \left|\mathcal{D}\left(A^P_{i,k_j}\right)_{\left[V^P_{i,k_{j,min}}:V^P_{i,k_{j,max}}\right]}\right|$, such that $I_{[I_{min}:I_{max}]}$ denotes the *sub-set operator* that, given a set *I*, extracts from *I* the sub-set spanning from $I_{min} \in I$ to $I_{max} \in I$.

For each dimensional attribute value in W, the decomposition task implemented by the first step of the approach we propose generates in $D_P^{\rhd}$ a decomposed attribute value set storing tuples of kind: $\langle V^{\rhd}_{i,k_j}, p^{\rhd}_{i,k_j}, \hat{V}^{\rhd}_{i,k_j}\rangle$. By composing such tuples for *all* the $|D|$ dimensional attributes in W, we finally obtain the generic tuple stored in $D_P^{\rhd}$ as follows: $\langle\langle V^{\rhd}_{i,k_0}, p^{\rhd}_{i,k_0}, \hat{V}^{\rhd}_{i,k_0}\rangle, \langle V^{\rhd}_{i,k_1}, p^{\rhd}_{i,k_1}, \hat{V}^{\rhd}_{i,k_1}\rangle, \ldots, \langle V^{\rhd}_{i,k_{|D|-1}}, p^{\rhd}_{i,k_{|D|-1}}, \hat{V}^{\rhd}_{i,k_{|D|-1}}\rangle\rangle$. It should be noted that, with respect to the original database schema of D_P and the input data cube schema W, the (database) schema of $D_P^{\rhd}$ only maintains the dimensional attributes d_0, d_1, …, $d_{|D|-1}$ and discards the (singleton) measure attribute m_0. This because the contribution of measure attribute values in D_P is "distributed" across decomposed attribute values in $D_P^{\rhd}$. Again, similarly to the previous consideration on the cardinality of $D_P^{\rhd}$ with respect to the cardinality of D_P, it should be clear enough that this further nice amenity contributes to lower the final spatial complexity of $D_P^{\rhd}$.

In the second step of our proposed approach for computing probabilistic OLAP data cubes, the final data cube $C(D_P)$ is aggregated from $D_P^{\rhd}$ directly based on conventional multidimensional aggregation methods [15] plus the novelty represented by an *innovative technique for computing confidence intervals and probabilities of probabilistic data cube cells* (see Sect. 2). For the sake of simplicity, let us to denote as $D = \{d_0, \ldots, d_{|D|-1}\} = \{A_{i,k_0}, A_{i,k_1}, \ldots, A_{i,k_{|D|-1}}\}$ the set of $|D|$ dimensions in W, by removing the notation allowing us to distinguish between exact and probabilistic dimensional attributes, due the fact that, as shown above, the decomposition task implemented by the first step of our proposed approach treats both kinds of attributes in a unified manner. Equally, we could refer to the schema of $D_P^{\rhd}$ as: $D_P^{\rhd}\left(A_{i,k_0}, A_{i,k_1}, \ldots, A_{i,k_{|D|-1}}\right)$. Let $\mathcal{D}(A_{i,k_0}), \mathcal{D}(A_{i,k_1}), \ldots, \mathcal{D}(A_{i,k_{|D|-1}})$ be the discretized domains of $A_{i,k_0}, A_{i,k_1}, \ldots, A_{i,k_{|D|-1}}$, respectively. Based on conventional multidimensional aggregation methods [15], for each multidimensional entry $\boldsymbol{h_l} = \left(\rho_{i,k_0,h_0}, \rho_{i,k_1,h_1}, \ldots, \rho_{i,k_{|D|-1},h_{|D|-1}}\right)$ in the *multidimensional space* defined by the *Cartesian product* $\mathcal{D}(A_{i,k_0}) \times \mathcal{D}(A_{i,k_1}) \times \ldots \times \mathcal{D}(A_{i,k_{|D|-1}})$, such that $h_0 \in \{0, 1, \ldots, |\mathcal{D}(A_{i,k_0})| - 1\}$, $h_1 \in \{0, 1, \ldots, |\mathcal{D}(A_{i,k_1})| - 1\}$, …, $h_{|D|-1} \in \{0, 1, \ldots, |\mathcal{D}(A_{i,k_{|D|-1}})| - 1\}$, a set of attribute values $\Psi(\boldsymbol{h_l}) = \Psi\left(\rho_{i,k_0,h_0}, \rho_{i,k_1,h_1}, \ldots, \rho_{i,k_{|D|-1},h_{|D|-1}}\right)$ is selected from $D_P^{\rhd}$. $\Psi(\boldsymbol{h_l})$ is defined as follows:

$$\Psi\left(\rho_{i,k_0,h_0}, \rho_{i,k_1,h_1}, \ldots, \rho_{i,k_{|D|-1},h_{|D|-1}}\right) = \begin{array}{l} \{\langle V_{i,k_0}^{\rhd}, p_{i,k_0}^{\rhd}, \hat{V}_{i,k_0}^{\rhd}\rangle, \\ \ldots, \\ \langle V_{i,k_{|D|-1}}^{\rhd}, p_{i,k_{|D|-1}}^{\rhd}, \hat{V}_{i,k_{|D|-1}}^{\rhd}\rangle \\ |V_{i,k_0}^{\rhd} = \rho_{i,k_0,h_0} \wedge \\ \ldots \wedge \\ V_{i,k_{|D|-1}}^{\rhd} = \rho_{i,k_{|D|-1},h_{|D|-1}}\} \end{array} \tag{4}$$

Finally, $\Psi(\boldsymbol{h_l})$ is used to compute the value of the probabilistic data cube cell $C(D_P)[\boldsymbol{h_l}] = \langle [B_{min}^{\boldsymbol{h_l}}, B_{max}^{\boldsymbol{h_l}}], p_{\boldsymbol{h_l}}\rangle$, whose characteristic parameters $B_{min}^{\boldsymbol{h_l}}$, $B_{max}^{\boldsymbol{h_l}}$ and $p_{\boldsymbol{h_l}}$ are defined as follows (from Sect. 2, recall that in our research we focus on SUM-based data cubes):

$$B_{min}^{\boldsymbol{h_l}} = \left\lfloor \begin{array}{c} argmin\left\{\hat{V}_{i,k_j}^{\rhd}\right\} \\ \Psi(\boldsymbol{h_l}), 0 \leq j \leq |D|-1 \end{array} \right\rfloor \tag{5}$$

$$B_{max}^{\boldsymbol{h_l}} = \left\lfloor \sum_{\Psi(\boldsymbol{h_l}), j=0}^{|D|-1} \hat{V}_{i,k_j}^{\rhd} \right\rfloor \tag{6}$$

$$p_{\boldsymbol{h_l}} = \prod_{\Psi(\boldsymbol{h_l}), j=0}^{|D|-1} p_{i,k_j}^{\rhd} \tag{7}$$

4 Conclusions and Future Work

A complete framework for computing multidimensional OLAP data cubes over probabilistic data has been proposed in this paper. We also conducted a set of preliminary experiments over synthetic probabilistic datasets (not shown in this paper for space reasons) that have confirmed the feasibility of the proposed framework. Future work is mainly focused on developing and conducting a wide experimental campaign on both synthetic and real-life probabilistic datasets, and on extending the framework in order to make it able of dealing with more complex SQL aggregation operators beyond the simple ones considered in the actual research (e.g., SUM, COUNT) and domain constraints over dataset attributes like those considered in [7].

References

[1] Agarwal, S., Agrawal, R., Deshpande, P., Gupta, A., Naughton, J.F., Ramakrishnan, R., Sarawagi, S.: On the Computation of Multidimensional Aggregates. In: Proceedings of VLDB 1996 Int. Conf. (1996)

[2] Agrawal, P., Benjelloun, O., Sarma, A.D., Hayworth, C., Nabar, S.U., Sugihara, T., Widom, J.: Trio: A System for Data, Uncertainty, and Lineage. In: Proceedings of VLDB 2006 Int. Conf. (2006)

[3] Barbarà, D., Garcia-Molina, H., Porter, D.: The Management of Probabilistic Data. IEEE Transactions on Knowledge Data Engineering 4(5) (1992)

[4] Benjelloun, O., Sarma, A.D., Halevy, A.Y., Theobald, M., Widom, J.: Databases with Uncertainty and Lineage. VLDB Journal 17(2) (2008)

[5] Burdick, D., Deshpande, P.M., Jayram, T.S., Ramakrishnan, R., Vaithyanathan, S.: OLAP over Uncertain and Imprecise Data. In: Proceedings of VLDB 2005 Int. Conf. (2005)

[6] Burdick, D., Deshpande, P.M., Jayram, T.S., Ramakrishnan, R., Vaithyanathan, S.: Efficient Allocation Algorithms for OLAP over Imprecise Data. In: Proceedings of VLDB 2006 Int. Conf. (2006)

[7] Burdick, D., Doan, A., Ramakrishnan, R., Vaithyanathan, S.: OLAP over Imprecise Data with Domain Constraints. In: Proceedings of VLDB 2007 Int. Conf. (2007)

[8] Cheng, R., Kalashnikov, D., Prabhakar, S.: Evaluating Probabilistic Queries over Imprecise Data. In: Proceedings of ACM SIGMOD 2003 Int. Conf. (2003)

[9] Colliat, G.: OLAP, Relational, and Multidimensional Database Systems. SIGMOD Record 25(3) (1996)

[10] Cuzzocrea, A.: Improving Range-Sum Query Evaluation on Data Cubes via Polynomial Approximation. Data & Knowledge Engineering 56(2) (2006)

[11] Cuzzocrea, A., Wang, W.: Approximate Range-Sum Query Answering on Data Cubes with Probabilistic Guarantees. Journal of Intelligent Information Systems 28(2) (2007)

[12] Dalvi, N. Suciu, D.: Efficient Query Evaluation on Probabilistic Databases. In: Proceedings of VLDB 2004 Int. Conf. (2004)

[13] Dalvi, N., Suciu, D.: Management of Probabilistic Data: Foundations and Challenges. In: Proceedings of ACM PODS 2007 Int. Conf. (2007)

[14] Deligiannakis, A., Roussopoulos, N.: Extended Wavelets for Multiple Measures. In: Proceedings of ACM SIGMOD 2003 Int. Conf. (2003)

[15] Gray, J., Chaudhuri, S., Bosworth, A., Layman, A., Reichart, D., Venkatrao, M., Pellow, F., Pirahesh, H.: Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals. Data Mining and Knowledge Discovery 1(1) (1997)

[16] Han, J., Kamber, M.: Data Mining: Concepts and Techniques, second ed. Morgan Kauffmann Publishers, San Francisco, CA, USA (2006)

[17] Harinarayan, V., Rajaraman, A., Ullman, J.: Implementing Data Cubes Efficiently. In: Proceedings of ACM SIGMOD 1996 Int. Conf. (1996)

[18] Papoulis, A.: Probability, Random Variables, and Stochastic Processes, second ed. McGraw-Hill, New York City, NY, USA (1984)

[19] Ré, C. Suciu, D.: Approximate Lineage for Probabilistic Databases. PVLDB 1(1) (2008)

[20] Sarma, A.D., Theobald, M., Widom, J: Exploiting Lineage for Confidence Computation in Uncertain and Probabilistic Databases. In: Proceedings of IEEE ICDE Int. Conf. (2008)

[21] Vassiliadis, P., Sellis, T.: A Survey of Logical Models for OLAP Databases. SIGMOD Record 28(4) (1999)

On Provenance of Data Fusion Queries

Domenico Beneventano, Abdul Rahman Dannoui, Antonio Sala

Dipartimento di Ingegneria dell'Informazione
Università di Modena e Reggio Emilia
Via Vignolese 905, 41125 Modena, Italy
firstname.lastname@unimore.it

Abstract. *Data Lineage* is an open research problem. This is particularly true in data integration systems, where information coming from different sources, potentially uncertain or even inconsistent with each other, is integrated. In this context, having the possibility to trace the lineage of certain data can help unraveling possible unexpected or questionable results.

In this paper, we describe our preliminary work about this problem in the context of the MOMIS data fusion system. We discuss and compare the use of *lineage* and *why*-provenance for the data fusion operator used in the MOMIS system; in particular we evaluate how the computation of the *why*-provenance should be extended to deal with *Resolution Functions* used in our data fusion system.

1 Introduction

Lineage, or *provenance*, in its most general definition, describes where data came from, how it was derived and how it was modified over time. Lineage provides valuable information that can be exploited for many purposes, ranging form simple statistical resumes presented to the end-user, to more complex applications such as managing data uncertainty or identifying and correcting data errors. For these reasons, in the last few years the research activity in the Information Management System area has been increasingly focused on this topic. In particular, lineage has been studied extensively in data warehouse systems [9,8]. However, in *Data Integration* systems, lineage is still considered as an open research problem [14,13]. Data Integration systems deal with information coming from different sources, potentially uncertain or even inconsistent with each other. In this context, collecting lineage information becomes a necessity. Lineage information helps the integration process by improving the system capability to introspect about the sources reliability and the certainty of the data.

A fundamental task in data integration is *data fusion*, the process of fusing multiple records representing the same real-world object into a single, consistent, and clean representation. Data fusion involves the resolution of possible conflicts between data coming from different sources [5]. A recent tutorial [10] listed data lineage as one of the open problems and desiderata for data fusion systems. Merging data implies a partial loss of the original values of the local sources. For this reason database administrators and data owners are notoriously hesitant to merge data. Data lineage can help explaining merging decisions by tracking which original values were involved and how they have been fused together.

In this paper, we describe our preliminary work on the application of data lineage techniques in the Data Fusion framework of the MOMIS Integration System. As described in previous works [2,1], MOMIS (Mediator envirOnment for Multiple Information Sources) is a framework to perform integration of structured and semi-structured data sources. MOMIS is characterized by a classical wrapper/mediator architecture: the local data sources contain the real data, while a Global Schema (GS) provides a *reconciled, integrated, read-only view* of the underlying sources. The GS and the mapping between GS and the local sources have to be defined at design time by the Integration Designer, together with *Resolution Functions* to solve data conflicts. End-users can then pose queries over this GS.

The concept of data lineage for relational databases was introduced in [9]: the *lineage* of an output record is based on identifying a subset of input records relevant to the output record; intuitively, an input record is relevant to an output record if it contributed to the existence of that output record. The notion of *why*-provenance defined in [6] is based on identifying subinstances of the input that “witness” a part of the output, i.e., *why*-provenance encodes all the *possible different derivations* of a output tuple in the query result by storing a set of input tuples *for each derivation*.

In this paper, we discuss and compare the use of *lineage* and *why*-provenance for the data fusion operator used in the MOMIS system; in particular we evaluate how the computation of the *why*-provenance should be extended to deal with *Resolution Functions* used in our data fusion system.

The remainder of the paper is organized as follows. In section 2 we will introduce the basic definitions of the MOMIS framework that will be used along the paper. In section 3 we will informally discuss the use of *lineage* and *why*-provenance for data fusion queries, then in section 4 we formally define thes concepts. Finally, conclusions and future works are sketched in section 5.

2 The MOMIS Data Fusion System

In this section we will introduce the basic definition of the MOMIS framework [2,1] that will be used along this paper. MOMIS has been developed by the DBGROUP of the University of Modena and Reggio Emilia[1]. An open source version of the MOMIS system is delivered and maintained by the academic spin-off DataRiver[2].

A MOMIS Data Integration System is constituted by: a set of *local schemas* $\{LS_1, \ldots, LS_k\}$, a *global schema* GS and *Global-As-View* (*GAV*) mapping assertions [15] between GS and $\{LS_1, \ldots, LS_k\}$. A global schema GS is a set of *global classes*, denoted by G. A local schema LS is a set of *local classes*, denoted by L. Both the global and the local schemas are expressed in the ODL_{I^3} language [4]. However, for the scope of this paper, we consider both the GS and the LS_i as relational schemas, but we will refer to their elements respectively as global and local classes to comply with the MOMIS terminology.

For each global class G, a *Mapping Table* (*MT*) is defined, whose columns represent a set of local classes $\{L_1, \ldots, L_n\}$ (this set is called *local classes belonging* to G and

[1] `http://www.dbgroup.unimore.it`

[2] `http://www.datariver.it`

Fig. 1. Examples of Mapping Tables

Mapping Table of G_1

G_1	L_1	L_2
ID	ID	ID
A	A	
B		B
C	C	C

Mapping Table of G_2

G_2	L_1	L_2	L_3
ID	ID	ID	ID
A	A		
B		B	
C	C	C	C

is denoted by $\mathcal{L}(G)$) and whose rows represent the global attributes GA of G. An element $MT[GA][L]$ represents the local attribute of L which is mapped onto the global attribute GA, or $MT[GA][L]$ is empty (there is no local attribute of L mapped onto the global attribute GA)[3]. A global attribute GA such as there is only a not null element $MT[GA][L]$ is called *one-to-one*, otherwise is called *one-to-many*. As an example, in Figure 1, we consider three local classes with schema $L_1(ID, A, C)$, $L_2(ID, B, C)$ and $L_3(ID, C)$, respectively, and we show the Mapping Table of the global classes G_1, with schema $G_1(ID, A, B, C)$ and with local classes $\mathcal{L}(G_1) = \{L_1, L_2\}$, and G_2 with schema $G_2(ID, A, B, C)$ and with local classes $\mathcal{L}(G_2) = \{L_1, L_2, L_3\}$.

GAV mapping assertions are expressed by specifying for each global class G a query over $\mathcal{L}(G)$, called *mapping query* and denoted by $\mathcal{MQ}^G$, which defines the instance of G starting from the instances of its local classes. The mapping query $\mathcal{MQ}^G$ is defined to make a global class perform *Data Fusion* among its local class instances [5]: multiple *local tuples* coming from local classes and representing the same real-world object are fused into a single and consistent *global tuple* of the global class. To identify multiple local tuples coming from local classes and representing the same real-world object, we assume that error-free and shared object identifiers exist among different sources: two local tuples with the same object identifier indicate the same object in different sources. In our example, we assume ID as an object identifier.

Data Reconciliation, i.e. to solve conflicts among instantiations of the same object in different sources, is performed by *Resolution Functions* [16]: for each GA such that there are more than one non empty element $MT[GA][L]$, a *Resolution Function* (*RF*) is defined to obtain, starting from the transformed local classes, a *merged value* for GA. In our example an `AVG` resolution function is defined on `C`.

In order to carry on the *Data Fusion* operations described so far, the mapping query $\mathcal{MQ}^G$ is defined by means of the *full outerjoin-merge operator* proposed in [17] and adapted to the MOMIS framework in [1]. Here, we consider an informal formulation of $\mathcal{MQ}^G$ in SQL. Intuitively, defining $\mathcal{MQ}^G$ by means of a full outerjoin-merge corresponds to the following two operations: (1) Computation of the *Full Outer Join*, on the basis of the shared object identifiers, of the *local classes* of *G*; (2) Application of the

[3] In this paper, for the sake of simplicity, we consider a simplified version of the MOMIS framework proposed in [2,1], where $MT[A][L]$ is a set of local attributes and *Data Transformation Functions* specify how local attribute values have to be transformed into corresponding global attribute values. Moreover we assume $S(G) = \cup_i S(L_i)$, i.e. global and local attribute names are the same.

Fig. 2. Instances of the local classes L_1, L_2 and L_3 and of the global classes G_1 and G_2 computed with the full outer join merge operator

L_1

ID	A	C
1	3	24
2	NULL	20
3	9	NULL
4	8	25
5	NULL	20

L_2

ID	B	C
1	3	24
2	NULL	30
3	NULL	20
5	NULL	30

L_3

ID	C
5	25

$G_1 = G_2$

ID	A	B	C
1	3	3	24
2	NULL	NULL	25
3	9	NULL	20
4	8	NULL	25
5	NULL	NULL	25

resolution functions. Thus $\mathcal{MQ}^G$ can be formulated by standard SQL, with the exception of (some) resolution functions. As an example, for the global class G_1 of Figure 1, we have the following $\mathcal{MQ}^{G_1}$:

```
MQ^G_1 : SELECT COALESCE(L1.ID,L2.ID) AS ID,
                L1.A AS A,
                L2.B AS B,
                AVG(L1.C,L2.C) AS C
         FROM L1 FOJ L2 ON (L2.ID=L1.ID)
```

where FOJ is an abbreviation for the SQL full outer join operator[4] , COALESCE is the standard SQL function which returns its first non-null parameter value and AVG is a (non standard SQL) function to compute the average value.

For the global class G_2 (with more than two local classes) of Figure 1, we have the following $\mathcal{MQ}^{G_2}$:

```
MQ^G_2: SELECT COALESCE(L1.ID,L2.ID,L3.ID) AS ID,
              L1.A AS A, L2.B AS B,
              AVG(L1.C,L2.C,L3.C) AS C
        FROM L1 FOJ L2 ON (L2.ID=L1.ID)
                FOJ L3 ON (L3.ID=L1.ID OR L3.ID=L2.ID )
```

As an example, in Figure 2, we show the instances of local classes[5] and the corresponding instance of the global classe computed by means of $\mathcal{MQ}^G$. As it can be easily verified, for this example, the instances of G_1 concide with the instances of G_2.

3 Provenance for Data Fusion Queries in MOMIS

In this paper we refer to the definition of *lineage* and *why*-provenance as formulated in [7] for relational databases. The concept of *lineage* for relational databases was introduced in [9]: the *lineage* of an output record is based on identifying a subset of input

[4] As shown in [1] if we use a join condition based on the shared object identifier ID, the order of local classes in the full outer join evaluation is not relevant.

[5] Let C denote either a local class L or a global class G. An istance of C is a set of tuples conforming with the schema of C; we conflate the notation and use the same symbol C for both the class C and an instance of C.

records relevant to the output record; intuitively, an input record is relevant to an output record if it contributed to the existence of that output record. As in [7], we consider *lineage* as a subinstance of the input, whereas in [9] lineage is defined as a vector of subsets of the input relations; this is a minor difference in presentation.

Buneman et al. [6] defined the notion of *why*-provenance in terms of a deterministic semistructured data model and query language. *Why*-provenance is based on identifying subinstances of the input that "witness" a part of the output, i.e., *why*-provenance encodes all the *possible different derivations* of an output tuple in the query result by storing a set of input tuples *for each derivation*. In [7], *why*-provenance is reformulated in terms of the relational model and relational algebra query language.

Both these concepts of *lineage* and *why*-provenance are defined for unions of conjunctive queries, and are then insufficient to capture provenance for data fusion queries based on the full outer join merge operator, which involves a form of negation. For this reason, we first informally discuss how the concepts of *lineage* and *why*-provenance should be extended to this new data fusion operator (section 3.1). Then we introduce their formal definitions in section 4.

3.1 Lineage and Why-Provenance for Projection Queries

We use q with subscripts to denote queries and Q to denote the relation that results from evaluating q. Let G be a global class and let $\mathcal{L}(G) = \{L_1, \ldots, L_n\}$ be the set of its local classes. Given a query on G, we want to express the lineage of a global tuple $t \in Q$ in terms of the local classes of G, thus:

- the *lineage* of t is a set of local tuples, i.e. an element of $\mathcal{P}(L_1 \cup \ldots \cup L_n)$.
- the *why-provenance* (or *witness basis*) of t is a set of sets of local tuples, i.e., is an element of $\mathcal{P}(\mathcal{P}(L_1 \cup \ldots \cup L_n))$.

The *lineage* of a (global) tuple in the result of a query is the set of (local) tuples that were involved in some derivation of that result tuple. We use L^{id} to denote the tuple t of a local class L with object identifier ID equal to id, i.e. $t[ID] = id$. In the same way, for a query q, Q^{id} denotes the tuple $t \in Q$ with object identifier ID equal to id.

In this paper, to define the provenance for the full join merge operator, we focus on Projection Queries on a single global class, where only *one-to-one* attributes (besides the object identifier ID) are selected. A first example is shown in Figure 3; in the derivation of Q_1^1 either L_1^1 and L_2^1 are involved, thus the lineage of Q_1^1 is $\{L_1^1, L_2^1\}$. In other words, if a global tuple is derived from the fusion of two local tuples, one from a local class, one from another local class, the lineage shows both these two local tuples.

Why-provenance [6] encodes all the *possible different derivations* of a global tuple in the query result by storing a set of local tuples *for each derivation*. In our example, there is a unique derivation of Q_1^1, which involve both the tuples L_1^1 and L_2^1, thus its why-provenance is $\{\{L_1^1, L_2^1\}\}$. For the tuple Q_1^3 there are two possible derivations, $\{L_1^3, L_2^3\}$ and $\{L_1^3\}$, thus its why-provenance is $\{\{L_1^3, L_2^3\}, \{L_1^3\}\}$. For the tuple Q_1^2 there are three possible derivations, thus its why-provenance is $\{\{L_1^2, L_2^2\}, \{L_1^2\}, \{L_2^2\}\}$. As it can be seen in the example, when only *one-to-one* attributes (besides the ID) are selected, different derivations are due to the presence of the *NULL* value into the local tuples.

Fig. 3. Lineage for $q_1 =$ `select ID,A,B from G1`

Q_1

ID	A	B
1	3	3
2	NULL	NULL
3	9	NULL
4	8	NULL
5	NULL	NULL

Lineage	*Why-provenance*	*Minimal Why-provenance*
$\{L_1^1, L_2^1\}$	$\{\{L_1^1, L_2^1\}\}$	$\{\{L_1^1, L_2^1\}\}$
$\{L_1^2, L_2^2\}$	$\{\{L_1^2\}, \{L_2^2\}, \{L_1^2, L_2^2\}\}$	$\{\{L_1^2\}, \{L_2^2\}\}$
$\{L_1^3, L_2^3\}$	$\{\{L_1^3, L_2^3\}, \{L_1^3\}\}$	$\{\{L_1^3\}\}$
$\{L_1^4\}$	$\{\{L_1^4\}\}$	$\{\{L_1^4\}\}$
$\{L_1^5, L_2^5\}$	$\{\{L_1^5\}, \{L_2^5\}, \{L_1^5, L_2^5\}\}$	$\{\{L_1^5\}, \{L_2^5\}\}$

In all the previous examples, we discussed the lineage and the why-provenance of global tuples obtained fusing two local tuples, one from a local class, and the other from a different local class. We now consider the tuple Q_1^4 coming from a tuple of $L1$ that has no corresponding tuple in $L2$. How can we then define the lineage of such a kind of tuple? Recently, in TRAMP [11], the concept of *PI-CS*-provenance (*Perm Influence Contribution Semantics*) is introduced to produce more precise provenance information for outer joins. As observed in [11], the lineage of Q_1^4 would contain all tuples from relation L_2, but in fact none of them contributed to Q_1^4. We agree with [11] claiming that a better semantics for the provenance of the tuple Q_3^1 would indicate that L_1^4 paired with no tuples from L_2 influences Q_1^4 (rather than saying every tuple of L_2 is in the provenance of this tuple). On the other hand, though, the *PI-CS* provenance as defined in TRAMP [11] is not able to represent *each derivation*, in fact for tuple Q_1^3 should show only the witness basis $\{\langle L_1^3,L_2^3\ \rangle\}$.

These considerations lead us to define the lineage and *why*-provenance of Q_1^4 in a straightforward way as $\{L_1^4\}$ and $\{\{L_1^4\}\}$, respectively.

Besides the concept of *why*-provenancee, in [6] the concept of *minimal why-provenance* is introduced to represent the *minimal witness basis* which consists of all the minimal witnesses in the witness basis, where a witness is minimal if none of its proper subinstances is also a witness in the witness basis. For example, $\{\{L_1^3\}\}$ is a minimal witness for Q_1^3 whereas $\{\{L_1^3, L_2^3\}, \{L_1^3\}\}$ is not.

We conclude this initial discussion about lineage for the data fusion operator considering the same query posed on the global class G_2 obtained from all the three local classes: we denote this query as q_2. As it can be easily verified, the result of q_2 is the same as the result of q_1, and the lineage of tuples that are not present in L_3 does not change. On the other hand, lineage of tuple Q_2^5 is $\{L_1^5, L_2^5, L_3^5\}$, while the *why*-provenance is $\mathcal{P}(\{L_1^5, L_2^5, L_3^5\}) \setminus \emptyset$ (i.e., all the possible non-empty subsets of the set $\{L_1^5, L_2^5, L_3^5\}$), the minimal why-provenance is $\{\{L_1^5\}, \{L_2^5\}, \{L_3^5\}\}$.

We will now discuss and compare what *lineage* and *why*-provenance produce for query with resolution functions, that represent the most significant aspect in our data fusion framework. To this end, we will consider the query q_3 posed on the global class G_2 (see Figure 4)which selects the *one-to-many* global attribute `C` defined by the `AVG` resolution function.

The computation of the `AVG` resolution function corresponds to performing a grouping on the object identifier and to the computation of the `AVG` aggregation function. On the other hand, either *lineage* and *why*-provenance defined in [7] are limitated to

Fig. 4. Lineage for $q_3 =$ `select ID,C from G2`

Q_3

ID	C	*Lineage*	*Why-provenance*	*Minimal Why-provenance*
1	24	$\{L_1^1, L_2^1\}$	$\{\{L_1^1\}, \{L_2^1\}, \{L_1^1, L_2^1\}\}$	$\{\{L_1^1\}, \{L_2^1\}\}$
2	25	$\{L_1^2, L_2^2\}$	$\{\{L_1^2, L_2^2\}\}$	$\{\{L_1^2, L_2^2\}\}$
3	20	$\{L_1^3, L_2^3\}$	$\{\{L_1^3, L_2^3\}, \{L_2^3\}\}$	$\{\{L_2^3\}\}$
4	25	$\{L_1^4\}$	$\{\{L_1^4\}\}$	$\{\{L_1^4\}\}$
5	25	$\{L_1^5, L_2^5, L_3^5\}$	$\{\{L_1^5, L_2^5, L_3^5\}, \{L_1^5, L_2^5\}, \{L_3^5\}\}$	$\{\{L_1^5, L_2^5\}, \{L_3^5\}\}$

Selection–Projection–Join–Union operations and thus are not defined for grouping with aggregation.

Lineage for grouping with aggregation is defined in TRIO [3] and in TRAMP [11]. *Trio*-lineage is similar to *why*-provenance, but derivations involving the same set of tuples are represented separately, i.e., with the *Trio*-lineage, *bag* of sets of input tuples are represented, each of which corresponds to one derivation. For grouping with aggregation, the *Trio*-lineage of a tuple t in the set of all tuples in the group that corresponds to t, and the same holds for the *PI-CS*-provenance introduced in TRAMP [11].

This way to define the lineage for grouping with aggregation can be applied in the case of the lineage for our data fusion operator containing resolution functions, obtaining the straightforward result shown in Figure 4; as an example for tuple Q_3^1 the lineage is $\{L_1^1, L_2^1\}$, i.e. the two local tuples which are fused to obtain Q_3^1.

On the other hand, either *Trio*-lineage [3] and *PI-CS*-provenance [11] for grouping with aggregation are not appropriate to define *why*-provenance for our data fusion operator containing resolution functions, since with the *why*-provenance we want to encode all the *possible different derivations*. In other words, we believe a better semantics for the *why*-provenance of the tuple Q_3^1 would be $\{\{L_1^1\}, \{L_2^1\}, \{L_1^1, L_2^1\}\}$, where each set of local tuples which corresponds to one derivation is represented. Another significant example is Q_3^5. In this way, for a global tuple obtained by means of an `AVG` resolution function, the *why*-provenance produces *all* the possible derivations, with a behaviour that is homogeneous with the case of one-to-one attributes discussed before. This is true in the case of the average function, but the application of the *why*-provenance need to be further analyzed for different types of resolution functions.

In [16], the properties of the resolution functions are examined; in particular, resolution functions and subdivided into *mediating* and *deciding* functions. A function is *mediating* if $RF(v_1, \ldots, v_n) = y$, meaning that a new value is created by the resolution function. Intuitively, for some mediating functions, such as `AVG` and `MEDIAN`, the *why*-provenance provides several witnesses , while for other ones, such as `SUM` and `CONCAT`, a unique witness is produced.

Deciding functions choose among the already present values, e.g., `COALESCE` or `SHORTEST`, where $RF(v_1, \ldots, v_i, \ldots, v_n) = v_i, i \in \{1, \ldots, n\}$. As observed in [16], if we assume that ties (e.g., two shortest values) are broken by a secondary criterion, e.g., the order of the values, we always get a defined result. Intuitively, for deciding functions, the *why*-provenance provides a unique witness with only the local tuple whose value is chosen by the resolution function.

In [7], the relationships between lineage, why-provenance (i.e., the witness basis) and the minimal witness basis are discussed. In particular, the authors show that both lineage and the minimal witness basis can be computed from why-provenance; however, neither lineage, nor why-provenance can be obtained from the minimal witness basis, as it can be verified also in our examples.

Another consideration about the relationship between why-provenance and minimal why-provenance that is peculiar of the data fusion operator is the following: both the second and the third tuple of q_1 of Figure 3, i.e., Q_1^3 and Q_1^4, have a null value for the attribute B, but in the case of Q_1^3 the null value is coming from a data source (the local class L_1), while in the case of Q_1^4 the null value is obtained because the attribute B has no mapping on a local source. In other words, the results from query q_1 show no differences between a null value coming from the data sources (tuple Q_1^3) and a null value value obtained because the attribute has no mapping on a local source (tuple Q_1^4).

The difference between these two cases can be described in terms of *why*-provenance, while the *minimal why*-provenance remains the same. In other words, having the possibility to query the *why*-provenance, we can have different results in these two cases, while it is not possible with the *minimal why*-provenance (neither with the *lineage*).

4 *Lineage* and *Why*-Provenance: a formal definition

Let G be a global class with schema $S(G)$ and let $\mathcal{L}(G) = \{L_1, \ldots, L_n\}$ be the set of its local classes, with schema $S(L_i)$, for each i. As in our previous examples, we assume ID as a share object identifier among all local classes, i.e. $ID \in \cap_i S(L_i)$ and we assume $S(G) = \cup_i S(L_i)$, i.e. global and local attribute names are the same (we will use A_i to denote global and local attributes).

Given a set of global attributes $S = \{A_1, \ldots, A_k\}$ we consider the query

$$q_S = \texttt{SELECT DISTINCT} \quad A_1, \ldots, A_k \ \texttt{FROM} \ G$$

i.e., we consider the set semantics.

Given a query q_S, we add to the select list the object identifier ID, i.e. we consider the query with attributes $S \cup \{ID\}$, denoted by q_S^*; we first define *lineage* and *why*-provenance for this query q_S^* and then we will give the definitions for a generic query q_S, i.e. for a query where the select list S doesn't necessarily contain the object identifier ID. *Lineage* and *why*-provenance for q_S^* will be respectively denoted by IDLIN and IDWHY; *lineage* and *why*-provenance for q_S will be respectively denoted by LIN and WHY.

The query q_S^* is rewritten w.r.t. local classes as follows:

```
SELECT
  COALESCE(L1.ID,L2.ID, L3.ID) AS ID,
  Li.A AS A -- if  A is mapped only in Li
  RF(L1.A, ... Lk.A) AS A -- if A is  mapped in L1, ... Lk
FROM L1 FOJ L2 ON (L2.ID=L1.ID)
     FOJ L3 ON (L3.ID=L1.ID OR L3.ID=L2.ID )
     ... FOJ Ln ON (Ln.ID=L1.ID OR ... OR Ln.ID=Ln-1.ID)
```

Let $u(id) \in q^*_S$ be the *unique* tuple of q^*_S with ID equal to id, i.e. $u(id)[ID] = id$; given $u(id) \in q^*_S$, we define the set $\mathbf{LC}(id) = \{L_i^{id} \mid \exists L_i \in \mathcal{L}(G), \exists L_i^{id} \in L_i\}$ of local tuples with ID equal to id; $\mathcal{P}(\mathbf{LC}(id))$ denotes the set of all subset of $\mathbf{LC}(id)$.

Given $u(id) \in q^*_S$, its lineage is defined by:

$$\text{IDL}\textsc{in}(q^*_A, u(id)) = \mathbf{LC}(id)$$

Thus, lineage for $u(id) \in q^*_S$, is indipendent from the selected global attributes, as shown in the examples of figures 3 and 4.

To define *why*-provenance, we first consider a single global attribute A; given $u(id) \in q^*_A$, $\text{IDW}\textsc{hy}(q^*_A, u(id))$ is a subset of $\mathcal{P}(\mathbf{LC}(id))$ defined as follows:

- if $A = ID$ is the object identifier: $\text{IDW}\textsc{hy}(q^*_A, u(id)) = \mathcal{P}(\mathbf{LC}(id))$.
- if A is one-to-one, mapped only on $L.A$:
  - if $L^{id}[A]$ is $NULL$
    then $\text{IDW}\textsc{hy}(q^*_A, u(id)) = \mathcal{P}(\mathbf{LC}(id))$
    else $\text{IDW}\textsc{hy}(q^*_A, u(id)) = \{S \in \mathcal{P}(\mathbf{LC}(id)) \mid L^{id} \in S\}$.
- if A is a one-to-many, mapped into $k \leq n$ local classes and defined by the resolution function RF:
  $\text{IDW}\textsc{hy}(q^*_A, u(id))$ = { { L_1^{id},...,L_q^{id} } | $q \leq k$, $L_i^{id} \in L_i, 1 \leq i \leq q$ and
  $RF(L_1^{id}[A], \ldots, L_k^{id}[A]) = u(id)[A]$ }

The first rule is trivial. The second rule takes into account that, due to the full outer join operation, a $NULL$ value for $u(id)[A]$, can come either from a $NULL$ value from the local class where A is mapped (i.e. $L^{id}[A]$) or it can be obtained from any local class where A is not mapped. The last rule takes into account a one-to-many global attribute A; in this case the value for A is computed by means of a resolution function: $FJ^{id}[A] = RF(v_1, v_2, \ldots, v_k)$, with $v_i = L_i^{id}[A]$, $1 \leq i \leq k$. The rules generate a witness { L_1^{id},...,L_q^{id} } for each subset $v_1, v_2, \ldots, v_q$, $q \leq k$, such that $RF(v_1, v_2, \ldots, v_q) = RF(v_1, v_2, \ldots, v_k)$.

The function $\text{IDW}\textsc{hy}(q^*_S, u(id))$ is extended to a subset $S = \{A_1, \ldots, A_k\}$ with $k \geq 1$, as follows:

$$\text{IDW}\textsc{hy}(q^*_S, u(id)) = \bigcap \Big\{ \text{IDW}\textsc{hy}(q^*_S, u(id)) \mid A \in S \Big\}$$

As an example, for the query in Figure 3 we have $S = \{ID, A, B\}$; $\text{IDW}\textsc{hy}(q^*_S, u(3))$ is computed as:

1. $\text{IDW}\textsc{hy}(q^*_{ID}, u(3))$ = $\mathcal{P}(\{L_1^3, L_2^3\}) \setminus \emptyset$
2. $\text{IDW}\textsc{hy}(q^*_A, u(3))$ = $\{\{L_1^3\}, \{L_1^3, L_2^3\}\}$
3. $\text{IDW}\textsc{hy}(q^*_B, u(3))$ = $\{\{L_1^3\}, \{L_2^3\}, \{L_1^3, L_2^3\}\}$

Then $\text{IDW}\textsc{hy}(q^*_S, u(3)) = \{\{L_1^3\}, \{L_1^3, L_2^3\}\}$.

Finally, we give the definitions for a generic query q_S, i.e. for a query where the select list S doesn't necessarily contain the object identifier ID. The *Lineage* and the *why*-provenance of $t \in q_S$ are, respectively, defined as follows:

$$\textsc{Lin}(q_S, t) = \bigcup \Big\{ \text{IDL}\textsc{in}(q^*_S, u(id)) \mid u(id)[S] = t \Big\}$$

$$\textsc{Why}(q_S, t) = \bigcup \Big\{ \text{IDW}\textsc{hy}(q^*_S, u(id)) \mid u(id)[S] = t \Big\}$$

5 Conclusion and Future Work

In this paper we prensented our work in progress to apply data provenance techniques in the Data Fusion framework of the MOMIS Integration System. We focused our attention on the extension of the concept of *why*-provenance to deal with Resolution Functions, by considering in particular the case of the average function; for resolution functions different from the AVG, the properties of the resolution functions need to be examined, starting from the discussion proposed in [16].

Moreover, in this preliminar paper, to investigate the provenance for the full join merge operator, we focused on Projection Queries on a single global class. While the extension to queries with `WHERE` conditions is straightforward, for more complex queries a further analysis is needed.

Future work will be directed in the following directions:

- *Querying Data Lineage*.
  In a data fusion scenario, the data lineage can be useful to understand the relation between the results we obtain querying a global class G and the local classes G is mapped on. This is particularly important for example to evaluate how the data we obtain from a data integration system can be affected when one or more local sources become unavailable.
  It is thus necessary to allow querying data lineage, providing an appropriate method to express conditions in our queries to consider tuples with lineage from certain local classes.
  For example, the results from query q_1 (figure 3) show no differences between a `NULL` value coming from the data sources (first tuple) and a `NULL` value obtained because the attribute has no mapping on a local source (second tuple). Having the possibility to query the data lineage, we can have different results in these two cases.
- *Where*-Provenance.
  Our preliminary work on data lineage started with analyzing *lineage* and *why*-Provenance; the next step will be analyzing also the *Where*-Provenance, with particular regards to resolution functions. The starting point will be the observation in [16]: mediating resolution functions does not allow evaluating the *Where*-Provenance, while it is possible to assign it with deciding resolution functions.
- *Other Provenance Models*.
  Beyond *Lineage* and *Why*-provenance, several other concepts of provenance (or provenance models) were proposed in literature, and, among these, the most informative form of provenance is the semiring of provenance polynomials [12]. On the other hands, these provenance models are defined for unions of conjunctive queries, and then need to be extended to capture provenance for queries based on the full outer join merge operator, which involves a form of negation.
- *Complexity Analysis*.
  In the comparison between *lineage* and *why*-provenance, also the computational costs must be considered: *why*-provenance provides more information than *lineage*, but, intuitively, its computational cost is higher. It is thus important to discuss this precision/cost tradeoff.

- *Implementation.*
  In the MOMIS system, to answer a query over a global class G, the query must be rewritten as an equivalent set of queries expressed on the local schemas (local queries); this query translation performs some query optimization techniques, such as predicate push down (to push a constraint on local queries) and full join simplification (to reduce full join to left/right/inner join). Thus, our idea is to extend our query processing to include also the provenance computation in the query rewriting, in order to be able to provide the user with lineage information when obtaining the results.

References

1. Beneventano, D., Bergamaschi, S., Guerra, F., Orsini, M.: Data integration. In: Embley, D., Thalheim, B. (eds.) Handbook of conceptual modelling. Springer-Verlag (2010), to appear. Available at http://dbgroup.unimo.it/SSE/SSE.pdf
2. Beneventano, D., Bergamaschi, S., Guerra, F., Vincini, M.: Synthesizing an integrated ontology. IEEE Internet Computing 7(5), 42–51 (2003)
3. Benjelloun, O., Sarma, A.D., Halevy, A., Widom, J.: Uldbs: databases with uncertainty and lineage. In: VLDB '06: Proceedings of the 32nd international conference on Very large data bases. pp. 953–964. VLDB Endowment (2006)
4. Bergamaschi, S., Castano, S., Vincini, M., Beneventano, D.: Semantic integration of heterogeneous information sources. Data Knowl. Eng. 36(3), 215–249 (2001)
5. Bleiholder, J., Naumann, F.: Data fusion. ACM Comput. Surv. 41(1), 1–41 (2008)
6. Buneman, P., Khanna, S., Tan, W.C.: Why and where: A characterization of data provenance. In: ICDT. pp. 316–330 (2001)
7. Cheney, J., Chiticariu, L., Tan, W.C.: Provenance in databases: Why, how, and where. Foundations and Trends in Databases 1(4), 379–474 (2009)
8. Cui, Y., Widom, J.: Lineage tracing for general data warehouse transformations. The VLDB Journal 12(1), 41–58 (2003)
9. Cui, Y., Widom, J., Wiener, J.L.: Tracing the lineage of view data in a warehousing environment. ACM Trans. Database Syst. 25(2), 179–227 (2000)
10. Dong, X.L., Naumann, F.: Data fusion: resolving data conflicts for integration. Proc. VLDB Endow. 2(2), 1654–1655 (2009)
11. Glavic, B., Alonso, G., Miller, R.J., Haas, L.M.: Tramp: Understanding the behavior of schema mappings through provenance. PVLDB 3(1), 1314–1325 (2010)
12. Green, T.J., Karvounarakis, G., Tannen, V.: Provenance semirings. In: Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems. pp. 31–40. PODS '07, ACM, New York, NY, USA (2007)
13. Halevy, A., Li, C.: Information integration research: Summary of nsf idm workshop breakout session. NSF IDM Workshop (2003)
14. Halevy, A., Rajaraman, A., Ordille, J.: Data integration: the teenage years. In: VLDB '06: Proceedings of the 32nd international conference on Very large data bases. pp. 9–16. VLDB Endowment (2006)
15. Lenzerini, M.: Data integration: A theoretical perspective. In: PODS. pp. 233–246 (2002)
16. Naumann, F., Bleiholder, J.: Conflict handling strategies in an integrated information system. In: Proceedings of the WWW Workshop in Information integration on the Web (IIWEB) (2006)
17. Naumann, F., Freytag, J.C., Leser, U.: Completeness of integrated information sources. Inf. Syst. 29(7), 583–615 (2004)

On Equality-Generating Dependencies in Ontology Querying (Extended Abstract)

Andrea Calì[1,3] and Andreas Pieris[2]

[1]Dept. of Computer Science and Information Systems, Birkbeck University of London, UK
[2]Computing Laboratory, University of Oxford, UK
[3]Oxford-Man Institute of Quantitative Finance, University of Oxford, UK

andrea@dcs.bbk.ac.uk
andreas.pieris@comlab.ox.ac.uk

Abstract. In ontology-based data access, data are queried through an *ontology* that offers a representation of the domain of interest. In this context, correct answers are those entailed by the logical theory constituted by the data and the ontology. Traditional database constraints like tuple-generating dependencies (TGDs) and equality-generating dependencies (EGDs) are a useful tool for ontology specification. However, their interaction usually leads to intractability or undecidability of query answering. *Separability* is the notion that captures the lack of interaction between TGDs and EGDs. In this paper we exhibit a novel and general sufficient condition for separability, in the case where the ontology is expressed with inclusion dependencies (a subclass of TGDs) and EGDs.

1 Introduction

Answering queries over ontologies has become an important problem in knowledge representation and databases. In ontology-enhanced database systems, an extensional relational database D is combined with an *ontological theory* Σ describing rules and constraints which derive new intensional data from the extensional data. A query is not just answered against the database D, but against the logical theory $D \cup \Sigma$. This problem has been addressed in several settings. For instance, the constraints of [1, 4, 6] are tailored to express Entity-Relationship schemata, while [12] deals with expressive constraints based on Answer Set Programming. The work [8] introduces and studies first-order constraints derived from a "light" version of F-logic [15], called *F-logic Lite*. Another relevant formalisms for knowledge bases, especially in the Semantic Web, is the *DL-lite family*; in [10, 17] tractable query answering techniques under DL-lite knowledge bases are presented.

In ontology-based query answering, a prominent family of languages, recently proposed, is the Datalog$^{\pm}$ family. In Datalog$^{\pm}$, the ontological theory is expressed by means of rules of two kinds: *(i) tuple-generating dependencies (TGDs)*, i.e., (function-free) Horn rules enhanced with the possibility of having existentially quantified variables in the head; *(ii) equality-generating dependencies*, that is, (function-free) Horn rules with a single equality atom in the head. Several decidable and tractable Datalog$^{\pm}$ languages have been studied [2, 3, 5, 7]. Even the least expressive Datalog$^{\pm}$ languages, with some extra features which do not increase the complexity of query answering, are able to properly extend the DL-lite languages. This suggests that TGDs and EGDs, which are in fact "traditional" database constraints, are a powerful and flexible tool for ontology modeling.

In this paper we stick to the language of TGDs and EGDs, and we address the problem of the interaction between the two types of constraints. Notice that, when there is no limitation on how TGDs and EGDs interact, the conjunctive query answering problem is undecidable; in fact, it is undecidable already for *inclusion dependencies (IDs)* and *key dependencies (KDs)*, two subclasses of TGDs and EGDs, respectively. For this reason the notion of *separability* was first proposed in [9]. A set $\Sigma = \Sigma_T \cup \Sigma_E$, where Σ_T and Σ_E are TGDs and EGDs, respectively, is said to be separable if, assuming the theory $D \cup \Sigma$ to be consistent, for each database D, the answers to a conjunctive query Q under Σ and under Σ_T coincide. In other words, EGDs do not play any role in query answering, and queries can be answered by considering the TGDs only. Several conditions have been proposed to ensure separability (see Section 3); here we propose a sufficient condition in the case where the constraints are IDs together with general EGDs. We also discuss that this condition can be easily combined with the known condition for TGDs (and thus IDs) and FDs [5], hence identifying a more general sufficient condition. The result can be straightforwardly extended to *linear* TGDs and EGDs, where linear TGDs are a slight generalization of IDs consisting of TGDs with exactly one atom in the body; however, for clarity of exposition, we illustrate our results in the case of IDs.

We do believe that our preliminary results pave the way to the discovery of more general separability conditions between EGDs (or their restrictions) and classes of TGDs such as *guarded TGDs* (*guarded Datalog*$^{\pm}$) [2] or *sticky(-join) sets of TGDs* (*sticky(-join) Datalog*$^{\pm}$) [5, 7].

2 Preliminaries

General. We define the following pairwise disjoint (infinite) sets of symbols: *(i)* a set Γ of *constants* (constitute the "normal" domain of a database), *(ii)* a set Γ_N of *labeled nulls* (used as placeholders for unknown values, and thus can be also seen as variables), and *(iii)* a set Γ_V of *variables* (used in queries and dependencies). Different constants represent different values (*unique name assumption*), while different nulls may represent the same value. We denote by $\mathbf{X}$ sequences of variables $X_1, \ldots, X_k$, where $k \geqslant 0$. Also, let $[n] = \{1, \ldots, n\}$, for each $n \geqslant 1$. We shall consider relational databases having, in general, values in $\Gamma \cup \Gamma_N$. We assume the reader is familiar with the relational model, and with (Boolean) conjunctive queries (CQs); for more details we refer the reader to, e.g., [5].

We recall that a *homomorphism* from a set of atoms to another set of atoms is a substitution $h : \Gamma \cup \Gamma_N \cup \Gamma_V \rightarrow \Gamma \cup \Gamma_N \cup \Gamma_V$ that is the identity on Γ. If there are homomorphisms from A_1 to A_2 and vice-versa, then A_1 and A_2 are *homomorphically equivalent*. The notion of homomorphism naturally extends to conjunctions of atoms. Given a set of symbols S, two atoms $\underline{a}_1$ and $\underline{a}_2$ are S-*isomorphic* iff there exists a bijection h such that $h(\underline{a}_1) = \underline{a}_2$, $h^{-1}(\underline{a}_2) = \underline{a}_1$, and h is the identity on S.

Dependencies. Given a schema $\mathcal{R}$, a *tuple-generating dependency (TGD)* σ over $\mathcal{R}$ is a first-order formula $\forall\mathbf{X}\forall\mathbf{Y}\,\varphi(\mathbf{X},\mathbf{Y}) \rightarrow \exists\mathbf{Z}\,\psi(\mathbf{X},\mathbf{Z})$, where $\varphi(\mathbf{X},\mathbf{Y})$ and $\psi(\mathbf{X},\mathbf{Z})$ are conjunctions of atoms over $\mathcal{R}$, called the *body* and the *head* of σ, denoted as $body(\sigma)$ and $head(\sigma)$, respectively. Henceforth, we will omit the universal quantifiers in TGDs. Such σ is satisfied by a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h such that $h(\varphi(\mathbf{X},\mathbf{Y})) \subseteq D$, there exists an extension h' of h (i.e., $h' \supseteq h$) where $h'(\psi(\mathbf{X},\mathbf{Z})) \subseteq D$. A restricted class of TGDs which is of special interest in this paper is the class of *inclusion dependencies (IDs)*, i.e., TGDs with just one body-atom and one head-atom, without repetition of variables neither in the body nor in the head.

An *equality-generating dependency (EGD)* η over $\mathcal{R}$ is a first-order formula of the form $\forall\mathbf{X}\,\varphi(\mathbf{X}) \rightarrow X_i = X_j$, where $\varphi(\mathbf{X})$ is a conjunction of atoms over $\mathcal{R}$, called the *body* and denoted as $body(\eta)$, and $\{X_i, X_j\} \subseteq \mathbf{X}$. For brevity, we will omit the universal quantifiers in EGDs. Such η is satisfied by a database D for $\mathcal{R}$ iff, whenever there exists a homomorphism h such that $h(\varphi(\mathbf{X})) \subseteq D$, then $h(X_i) = h(X_j)$.

CQ Answering under Dependencies. We now define *query answering* under TGDs and EGDs. Given a database D for $\mathcal{R}$, and a set Σ of TGDs and EGDs over $\mathcal{R}$, the *models* of D w.r.t. Σ, denoted as $mods(D,\Sigma)$, is the set of all databases B such that $B \models D \cup \Sigma$, i.e., $B \supseteq D$ and B satisfies Σ. The *answer* to a CQ Q w.r.t. D and Σ, denoted as $ans(Q,D,\Sigma)$, is the set $\{\mathbf{t} \mid \mathbf{t} \in Q(B), \text{ for each } B \in mods(D,\Sigma)\}$. The *answer* to a Boolean CQ (BCQ) Q w.r.t. D and Σ is *positive*, denoted as $D \cup \Sigma \models Q$, iff $ans(Q,D,\Sigma) \neq \varnothing$. CQ and BCQ answering under TGDs and EGDs are LOGSPACE-equivalent [2]. We thus focus only on the BCQ answering problem.

The Chase Procedure. The *chase procedure* (or simply *chase*) is a fundamental algorithmic tool introduced for checking implication of dependencies [16], and later for checking query containment [14]. Informally, the chase is a process of repairing a database w.r.t. a set of dependencies so that the resulted database satisfies the dependencies. The chase works on an instance through the so-called TGD and EGD *chase rules*. A violation of a TGD is repaired by *adding* one atom in order to satisfy it, where the positions corresponding to the existentially-quantified variables are occupied by fresh values in Γ_N. A violation of an EGD is repaired by *unifying* symbols in order to satisfy it. If two constants of Γ are unified, then we have a *hard violation* and the chase *fails*. We shall use the term *chase* interchangeably for both the procedure and its result. The chase of an instance D w.r.t. a set of dependencies Σ is denoted as $chase(D,\Sigma)$. For space reasons, we have to refer the reader to, e.g., [5] for more details.

The (possibly infinite) chase of D w.r.t. Σ is a *universal model* of D w.r.t. Σ, i.e., for each database $B \in mods(D,\Sigma)$, there exists a homomorphism from $chase(D,\Sigma)$ to B [11, 13]. Using this fact it can be shown that the chase is a formal tool for query answering under TGDs and EGDs. In particular, given a BCQ Q, $D \cup \Sigma \models Q$ iff $chase(D,\Sigma) \models Q$, providing that the chase does not fail. If the chase fails, then the set of models of D w.r.t. Σ is empty, and $D \cup \Sigma \models Q$ holds trivially.

3 Overview of Decidable Classes

A semantic notion that ensures decidability of query answering under sets of TGDs and EGDs, providing that the set of TGDs falls in a decidable class, is *separability* [3, 9].

Roughly speaking, separability guarantees that queries can be answered by considering only the set of TGDs (apart from an initial check whether the chase fails); the formal definition is given in Section 4. Several sufficient syntactic conditions for separability have been proposed in the literature.

An early separable class of IDs and KDs, called *key-based*, was proposed in the seminal work of Johnson and Klug [14]. In short, given an ID σ of the form $r(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, s(\mathbf{X}, \mathbf{Z})$, *(i)* the set of $\mathbf{X}$-attributes of $head(\sigma)$ must be strictly contained in the set of key attributes of the relation s, and also *(ii)* the $\mathbf{X}$-attributes of $body(\sigma)$ must be disjoint from the set of key attributes of the relation r.

As observed by Cali et al. [9], the first condition of key-based sets of IDs and KDs, as explained above, can be relaxed so that the set of $\mathbf{X}$-attributes of $head(\sigma)$ can be also equal to the set of key attributes of s. Furthermore, the second condition, which imposes a restriction on the bodies of the IDs, is not needed. In particular, the class of *non-key-conflicting (NKC) IDs* was defined which is as follows: given an ID $\sigma = r(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, s(\mathbf{X}, \mathbf{Z})$, the set of $\mathbf{X}$-attributes of $head(\sigma)$ is not a proper set of the set of key attributes of s. Notice that NKC IDs capture the well-known class of *foreign key dependencies*, which corresponds to the case where the set of $\mathbf{X}$-attributes of $head(\sigma)$ is equal to the set of key attributes of s.

The class of NKC IDs was generalized in [3] to the context of arbitrary (single-head) TGDs by defining the class of *non-key-conflicting TGDs*. Actually, the underlying idea is the same: given a TGD $\sigma = \varphi(\mathbf{X}, \mathbf{Y}) \rightarrow \exists \mathbf{Z}\, r(\mathbf{X}, \mathbf{Z})$, the set of $\mathbf{X}$-attributes of $head(\sigma)$ is not a proper set of the set of key attributes of r; moreover, each existentially quantified variable in $head(\sigma)$ must occur only once. In [5], was observed that the class of non-key-conflicting TGDs can be effortless extended to treat, not just keys, but functional dependencies.

The main reason due to which the above classes are separable is because there is no real interaction among TGDs and EGDs. In other words, once the given database satisfies the set of EGDs, we know that it is not possible to apply an EGD during the construction of the chase. The separable classes of IDs and KDs introduced in [4, 6] in the context of Entity-Relationship schemata, instead, are such that KDs can be applied during the chase. The separable class of IDs and EGDs that we propose in this paper is actually a generalization of the classes introduced in [4, 6].

4 Separable IDs and EGDs

In this section we exhibit a sufficient syntactic condition for separability between a set of IDs and a set of EGDs. Before we proceed further, let us give the formal definition of separability [3, 9].

Definition 1. *Let Σ_T be a set of TGDs over a schema $\mathcal{R}$, and Σ_E a set of EGDs over $\mathcal{R}$. The set $\Sigma = \Sigma_T \cup \Sigma_E$ is* separable *if, for every database D for $\mathcal{R}$, either $chase(D, \Sigma)$ fails, or $chase(D, \Sigma) \models Q$ iff $chase(D, \Sigma_T) \models Q$, for every BCQ Q over $\mathcal{R}$.*

Non-Conflicting Sets of IDs and EGDs. We now define when a set of IDs and EGDs is *non-conflicting*, and then establish that this condition is indeed sufficient for separability. Before we proceed further, let us give some preliminary definitions.

First, we define the notion of *affected positions* of a relational schema w.r.t. a set of TGDs. Given a schema $\mathcal{R}$, and a set Σ of TGDs over $\mathcal{R}$, an *affected position* of $\mathcal{R}$ w.r.t. Σ is defined inductively as follows. Let π_h be a position in the head of a TGD $\sigma \in \Sigma$. If an existentially quantified variable occurs at π_h, then π_h is affected w.r.t. Σ. If the same universally quantified variable X appears both in position π_h, and in the body of σ at affected positions *only*, then π_h is affected w.r.t. Σ. Intuitively speaking, the affected positions of a schema w.r.t. a set Σ of TGDs, are those positions at which a labeled null may occur during the construction of the chase under Σ.

A useful notion is the well-known *query containment* under TGDs. In particular, given a set Σ of TGDs over a schema $\mathcal{R}$, and two CQs Q_1 and Q_2 over $\mathcal{R}$, we say that Q_1 is contained in Q_2 w.r.t. Σ, written $Q_1 \subseteq_\Sigma Q_2$, iff $Q_1(D) \subseteq Q_2(D)$, for every database D for $\mathcal{R}$ that satisfies Σ.

Consider now a set Σ_T of IDs over a schema $\mathcal{R}$, and an EGD η over $\mathcal{R}$ of the form $\varphi(\mathbf{X}) \rightarrow X_i = X_j$, where $\{X_i, X_j\} \subseteq \mathbf{X}$; we assume w.l.o.g. that Σ_T and η have no variables in common. Let λ be the substitution $\{X_j \rightarrow X_i\}$. The *derivation forest* for η under Σ_T, denoted as F_{η,Σ_T}, is constructed as follows. If at least one occurrence of the so-called *watched variable* X_i in $\lambda(\varphi(\mathbf{X}))$ occurs at a non-affected position, then F_{η,Σ_T} is empty; otherwise, the roots of the forest are the atoms of $\lambda(\varphi(\mathbf{X}))$. Now, we iteratively apply the following step to every atom $\underline{a}$ in the part of F_{η,Σ_T} constructed so far; let $\mathcal{V}_\eta$ be the set of all variables appearing in the atoms of $\varphi(\mathbf{X})$. For each $\sigma \in \Sigma_T$ (for which we assume w.l.o.g. that has no variables in common with any of the atoms in the part of F_{η,Σ_T} constructed so far), if there exists a homomorphism h such that $h(head(\sigma)) = \underline{a}$, and also

1. $h(body(\sigma))$ contains the watched variable X_i,
2. all the occurrences of X_i in $h(body(\sigma))$ occur at affected positions of $\mathcal{R}$ w.r.t. Σ_T, and
3. $h(body(\sigma))$ is not $\mathcal{V}_\eta$-isomorphic to some ancestor of $\underline{a}$ in the part of F_{η,Σ_T} so far constructed,

then add $h'(body(\sigma))$, where $h' \supseteq h$ maps the variables that occur in the body but not in the head of σ to their self, as a child of $\underline{a}$ in F_{η,Σ_T}.

We are now ready to define formally when a set of IDs and EGDs is non-conflicting. Henceforth, for notational convenience, given a set Σ of dependencies, we will denote by Σ_T and Σ_E the set of IDs and EGDs, respectively.

Definition 2. *Consider a set Σ of IDs and EGDs over a schema $\mathcal{R}$. We say that Σ is* non-conflicting *if, for each $\eta \in \Sigma_E$ of the form $\varphi(\mathbf{X}) \rightarrow X_i = X_j$, the following condition holds. For each atom $\underline{a}$ in F_{η,Σ_T}, $Q_1 \subseteq_{\Sigma_T} Q_2$, where Q_1 and Q_2 are the conjunctive queries $q(\mathbf{Y}) \leftarrow \varphi(\mathbf{X})$ and $q(\mathbf{Y}) \leftarrow \underline{a}$, respectively, where $\mathbf{Y}$ are the variables that appear both in $\varphi(\mathbf{X})$ and $\underline{a}$.*

Example 1. Consider the set Σ consisting by the TGDs

$$\begin{aligned}
\sigma_1 &: s(X,Y) \rightarrow r(Y,X)\\
\sigma_2 &: p(X) \rightarrow \exists Y\, s(Y,X)\\
\sigma_3 &: t(X,Y) \rightarrow r(X,Y)\\
\sigma_4 &: r(X,Y) \rightarrow s(Y,X)
\end{aligned}$$

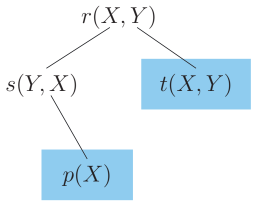


Fig. 1. The derivation forest F_{η,Σ_T} for Example 1.

and the EGD

$$\eta \;:\; r(X,Y), r(X,Z) \;\rightarrow\; Y = Z.$$

The derivation forest F_{η,Σ_T} is depicted in Figure 1. Note that the shaded nodes are not part of the forest. The atom $t(X,Y)$ is not added since the watched variable Y occurs at a non-affected position, while the atom $p(X)$ is not added since it does not contain the watched variable Y.

It is not difficult to see that $Q_1 \subseteq_{\Sigma_T} Q_2$ and $Q_1 \subseteq_{\Sigma_T} Q_3$, where

$$\begin{aligned} Q_1 &: q(Y) \leftarrow r(X,Y), r(X,Z) \\ Q_2 &: q(Y) \leftarrow r(X,Y) \\ Q_3 &: q(Y) \leftarrow s(Y,X). \end{aligned}$$

Consequently, Σ is non-conflicting. ■

Finiteness of the Derivation Forest. Let us now establish that the derivation forest of an EGD under a set of IDs is always finite.

Proposition 1. *Consider a set Σ_T of IDs over a schema $\mathcal{R}$, and an EGD η over $\mathcal{R}$. The derivation forest of η under Σ_T is finite.*

Proof. It suffices to show that on a certain path P of the derivation forest of η under Σ_T only finitely many non-$\mathcal{V}_\eta$-isomorphic atoms can appear. Let η be of the form $\varphi(\mathbf{X}) \rightarrow X_i = X_j$, and $\lambda = \{X_j \rightarrow X_i\}$. Observe that two atoms $\underline{a}$ and $\underline{b}$ of P are $\mathcal{V}_\eta$-isomorphic iff $\underline{a}^\star = \underline{b}^\star$, where $\underline{a}^\star$ and $\underline{b}^\star$ are the atoms obtained by replacing in $\underline{a}$ and $\underline{b}$, respectively, the variables that do not occur in $\lambda(\varphi(\mathbf{X}))$ with the "don't care" character "$\star$". Therefore, the maximum number of non-$\mathcal{V}_\eta$-isomorphic atoms that we can have on P is $|\mathcal{R}| \cdot (|S|+1)^w$, where w is the maximum arity over all predicates of $\mathcal{R}$, and S is the set of symbols that can appear on P, that is, the variables and constants that appear in the root node of P, and the constants that occur in Σ_T. Since both $\mathcal{R}$ and Σ_T are finite, the claim follows. □

Since the CQ containment problem under the class of IDs is decidable [14], we immediately get that the non-conflicting condition as defined above is decidable.

Soundness and Completeness. We now establish that non-conflicting sets of IDs and EGDs are indeed separable. Let us establish first an auxiliary technical lemma.

Lemma 1. *Consider a non-conflicting set Σ of IDs and EGDs over a schema $\mathcal{R}$. If $chase(D, \Sigma)$ does not fail, then, for every database D for $\mathcal{R}$, there exists a homomorphism h such that $h(chase(D, \Sigma)) \subseteq chase(D, \Sigma_T)$.*

Proof (sketch). The proof is by induction on the number of applications of the (TGD or EGD) chase rule. It is possible to show that, for each $k \geqslant 0$, there exists a homomorphism h_k such that $h_k(chase^{[k]}(D, \Sigma)) \subseteq chase(D, \Sigma_T)$, where $chase^{[k]}(D, \Sigma)$ is the initial finite part of the chase obtained by applying k times either the TGD or the EGD chase rule. Thus, the desired homomorphism is eventually $h = \bigcup_{i=0}^{\infty} h_i$. □

Theorem 1. *If a set Σ is non-conflicting, then it is also separable.*

Proof. Let D be a database for $\mathcal{R}$ such that $chase(D, \Sigma)$ does not fail. Clearly, by construction, $chase(D, \Sigma)$ satisfies all the dependencies in Σ. Therefore, $chase(D, \Sigma) \in mods(D, \Sigma) \subseteq mods(D, \Sigma_T)$. Since $chase(D, \Sigma_T)$ is a universal model of D w.r.t. Σ_T we immediately get that there exists a homomorphism h such that $h(chase(D, \Sigma_T)) \subseteq chase(D, \Sigma)$. On the other hand, Lemma 1 implies that there exists a homomorphism h' such that $h'(chase(D, \Sigma)) \subseteq chase(D, \Sigma_T)$. Due to h and h' we get that $chase(D, \Sigma)$ and $chase(D, \Sigma_T)$ are homomorphically equivalent. Thus, for every BCQ Q over $\mathcal{R}$, it holds that $chase(D, \Sigma) \models Q$ iff $chase(D, \Sigma_T) \models Q$, and the claim follows. □

Query Answering under Non-Conflicting Sets. We conclude by investigating the *data* and *combined complexity* of BCQ answering under non-conflicting sets. The data complexity is calculated by considering only the data as input, while the combined complexity by considering also the query and the dependencies as part of the input.

Theorem 2. *Consider a BCQ Q over a schema $\mathcal{R}$, a database D for $\mathcal{R}$, and a non-conflicting set Σ of IDs and EGDs over $\mathcal{R}$. The problem whether $D \cup \Sigma \models Q$ is in* AC^0 *in data complexity, and is* PSPACE*-complete in combined complexity.*

Proof (sketch). Suppose that $chase(D, \Sigma)$ does not fail. By Theorem 1, we get that $D \cup \Sigma \models Q$, or, equivalently, $chase(D, \Sigma) \models Q$ iff $chase(D, \Sigma_T) \models Q$. It is well-known that BCQ answering under IDs is in AC^0 in data complexity [3], and PSPACE-complete in combined complexity [14]. Since Σ_T is a set of IDs, providing that the chase does not fail, the desired complexity follows. Since the problem whether $chase(D, \Sigma)$ fails is tantamount to BCQ answering under IDs (see, e.g., [6]), the claim follows. □

5 Conclusions

We have addressed the problem of separability between TGDs and EGDs in the context of ontological query answering. We have exhibited a sufficient, syntactically checkable condition for separability for the case of IDs and general EGDs. Our non-conflicting condition can be combined with existing techniques in order to capture additional cases that involve functional dependencies which are not triggered during the construction of the chase. In particular, can be combined with the non-conflicting notion proposed in [5] for general TGDs and FDs. In this case, we say that a set is non-conflicting if the condition given in [5] is satisfied, or our non-conflicting condition is satisfied.

For simplicity reasons, in this work we considered only IDs. However, the non-conflicting condition can be extended to the slightly more general class of linear TGDs, i.e., TGDs with just one body-atom. This can be achieved by modifying the non-conflicting condition in such a way that, instead of a homomorphism that maps the head of a TGD σ to an atom $\underline{a}$ of the derivation forest, we need that $head(\sigma)$ and $\underline{a}$ unify. Then, we exploit the *most general unifier* of $head(\sigma)$ and $\underline{a}$.

Acknowledgements. The research leading to these results has received funding from the European Research Council under the European Community's Seventh Framework Programme (FP7/2007-2013)/ERC grant agreement no. 246858 – DIADEM.

References

1. A. Calì, D. Calvanese, G. D. Giacomo, and M. Lenzerini. Accessing data integration systems through conceptual schemas. In *Proc. of ER*, pages 270–284, 2001.
2. A. Calì, G. Gottlob, and M. Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *Proc. of KR*, pages 70–80, 2008.
3. A. Calì, G. Gottlob, and T. Lukasiewicz. A general datalog-based framework for tractable query answering over ontologies. In *Proc. of PODS*, pages 77–86, 2009.
4. A. Calì, G. Gottlob, and A. Pieris. Tractable query answering over conceptual schemata. In *Proc. of ER*, pages 175–190, 2009.
5. A. Calì, G. Gottlob, and A. Pieris. Advanced processing for ontological queries. *PVLDB*, 3(1):554–565, 2010.
6. A. Calì, G. Gottlob, and A. Pieris. Query answering under expressive entity-relationship schemata. In *Proc. of ER*, pages 347–361, 2010.
7. A. Calì, G. Gottlob, and A. Pieris. Query answering under non-guarded rules in Datalog+/-. In *Proc. of RR*, pages 175–190, 2010.
8. A. Calì and M. Kifer. Containment of conjunctive object meta-queries. In *Proc. of VLDB*, pages 942–952, 2006.
9. A. Calì, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of PODS*, pages 260–271, 2003.
10. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The DL-lite family. *J. Autom. Reasoning*, 39(3):385–429, 2007.
11. A. Deutsch, A. Nash, and J. B. Remmel. The chase revisisted. In *Proc. of PODS*, pages 149–158, 2008.
12. T. Eiter and M. Simkus. FDNC: Decidable nonmonotonic disjunctive logic programs with function symbols. *ACM Trans. Comput. Log.*, 11(2), 2010.
13. R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: Semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.
14. D. S. Johnson and A. C. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *J. Comput. Syst. Sci.*, 28(1):167–189, 1984.
15. M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *J. ACM*, 42(4):741–843, 1995.
16. D. Maier, A. O. Mendelzon, and Y. Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 4(4):455–469, 1979.
17. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. Data Semantics*, 10:133–173, 2008.

Keyword-based Search in Data Integration Systems*

Sonia Bergamaschi[1], Elton Domnori[1], Francesco Guerra[1], Raquel Trillo Lado[2], and Yannis Velegrakis[3]

[1] Università di Modena e Reggio Emilia, via Università 4, 41121 Modena, Italy
firstname.lastname@unimore.it

[2] SID - University of Zaragoza, María de Luna, 1, 50018 Zaragoza, Spain
raqueltl@unizar.es

[3] DISI - Università di Trento, Via Sommarive 14, 38123 Povo (TN), Italy
velgias@disi.unitn.eu

Abstract. In this paper we describe Keymantic, a framework for translating keyword queries into SQL queries by assuming that the only available information is the source metadata, i.e., schema and some external auxiliary information. Such a framework finds application when only intensional knowledge about the data source is available like in Data Integration Systems.

1 Introduction

One of the main motivations for supporting the research on data integration is to provide the user with a unique view synthesizing a set of distributed data sources. By querying a unique integrated view, the user obtains an answer that is the union of the results returned by the sources involved in the integration process. Since the benefits of this approach for end users are firm, the research community has been focused on techniques for building and querying the unified view [9] for over 20 years. Despite the results obtained in the field, data integration systems are not really permeating the real world, i.e., we register a low presence of data integration systems in real business environments. We think that one of the reasons is that querying in such environments is too complex for end users.

Data integration systems are typically queried by means of requests expressed in the native query languages (in general structured query languages such as SQL, OQL, SPARQL, ...). This is definitely a limit for a large exploitation of these systems: they require skilled users and also impose on data integration systems some of the limitations intrinsic to the query languages. A complete knowledge of the underlying data structures and their semantics is needed to formulate queries into a structured query language. Unfortunately, the former requires the user to deeply explore the structure of the source, that is an error prone and time consuming process when a source is composed of hundreds of unknown tables and attributes. The latter may be too large and too

* Extended abstract of the paper “S. Bergamaschi, E. Domnori, F. Guerra, R. Trillo Lado, Y. Velegrakis: Keyword Search over Relational Databases: a Metadata Approach, to appear at SIGMOD 2011” - This work was partially supported by project “Searching for a needle in mountains of data” (http://www.dbgroup.unimo.it/keymantic/).

complex to be communicated to the user. Understanding the semantics conveyed by the unified view means to know both the semantics conveyed by the data sources involved in the integration process and how the semantics of the local views are mapped to the integrated view. Therefore, it is clear that such a requirement may nullify the whole motivation for integrating sources: for users holding this knowledge the advantages of working with an integrated source are highly lowered.

Keyword-based searching has been introduced as a viable alternative to the highly structured query languages. A number of keyword searching systems over structured data have been developed, e.g., BANKS, DISCOVER, DBXplorer, Prècis and many others presented in various surveys [6, 11]. Their typical approach is to perform an off-line pre-processing step that scans the whole instance data and constructs an index, a symbol table or some structure which is later used during the run time to identify the parts of the database in which each keyword appears. Once they discover it, they perform a path discovery algorithm to find the different ways that these parts are connected (e.g., finding minimal joining networks, or Steiner trees).

Unfortunately, although keyword-based techniques can be very successful as local database services, they cannot be easily applied in large Data Integration Systems. One of the reasons is that the built index requires continuous maintenance since it is based on data values that may frequently change. If the index is locally stored in the source, it cannot be used to index the data of all the sources of a data integration system since it may be under the responsibility of different owners. Furthermore, in data integration systems, the data sources may not expose their whole data, but only portions of their schema, thus making impossible to build an index over the instances. On the other hand, integration systems rely on metadata, in the form of data types, lexical references, mappings extracted from the sources to be integrated, meaningful for addressing the solution of a keyword query, but that are not really exploited by the current keyword based search engines.

To overcome the above issues, we introduce Keymantic[1, 2] a new framework for keyword based searching on data integration systems that, in contrast to existing approaches, exploits intensional knowledge to transform keyword queries into semantically meaningful SQL queries that can be executed by the data integration system.

The structure of this paper is the following. Section 2 provides a motivating example, Section 3 introduces our approach and Section 4 sketches out some conclusion and future work.

2 Motivating Example

Let us assume that a virtual tourism district composed of a set of companies (travel agencies, hotels, local public administrations, tourism promotion agencies) wants to publish an integrated view of their tourism data about a location (see Figure 1). Keymantic allows users to query that data source with a two step process: firstly the keyword query is analyzed for discovering its intended meaning, then a ranked set of SQL queries, expressing the discovered meaning according to the database structure, is formulated.

Each keyword represents some piece of information that has been modeled in the database, but, depending on the design requirements of the data source, this piece might

Person

Name	Phone	City	Email
Saah	4631234	London	saah@aaa.bb
Sevi	6987654	Auckland	eevi@bbb.cc
Edihb	1937842	Santiago	edibh@ccc.dd

Reserved

Person	Hotel	Date
Saah	x123	6/10/2009
Sevi	cs34	4/3/2009
Edihb	cs34	7/6/2009

Hotel

id	Name	Address	Service	City
x123	Galaxy	25 Blicker	restaurant	Shanghai
cs34	Krystal	15 Tribeca	parking	Cancun
ee67	Hilton	5 West Ocean	air cond.	Long Beach

City

Name	Country	Description
Shanghai	China	...
Cancun	Mexico	...
Long Beach	USA	...
New York	USA	...

Booked

Person	Rest	Date
Saah	Rx1	5/4/2009
Sevi	Rx1	9/3/2009

Restaurant

id	Name	Address	Specialty	City
Rx1	Best Lobster	25, Margaritas	Seafood	Cancun
Rt1	Naples	200 Park Av.	Italian	New York

Fig. 1. A fraction of a database schema with its data.

have been modeled as data or metadata. Thus, the first task is to discover what each keyword models in the specific data source and to which metadata / data may be associated to. The association between keywords and database needs to be approximate: the synonymous and polysemous terms might allow the discovery of multiple intended meanings, each one with a rank expressing its relevance. Let us consider, for example, a query consisting of the keywords "Restaurant Naples". For instance, a possible meaning of the query might be "find information about the restaurant called Naples". In this case, the former keyword should be mapped into a metadata (the table Restaurant) and the other one into a value of the attribute Name of the same table Restaurant. A user might have in mind other meanings for the same keywords, for example, "find the restaurants that are located in the Naples Avenue", or "in the Naples city", or "that cook Naples specialties". All these intended meanings give rise to different associations of the keyword `Naples`; attributes Address, City or Specialty of the table Restaurant. This example shows also that keywords in a query are not independent: we expect that the keywords express different features of what the user is looking for. For this reason we expect that in our example "Naples" is a value referring to an element of the Restaurant table. If the user had formulated the keyword query "Restaurant name Naples", the number of possible intended meanings would have been reduced, since the keywords `name` forces the mappings of Naples into the attribute Name of the table Restaurant. Notice that different intended meanings may generate a mapping from the same keyword both into metadata and into data values. For example, in our database restaurant is the name of a table, but it is also one of the possible values of the attribute Service in the table Hotel. Finally, the order of the keywords in a query is also another element to be taken into account since related elements are usually close. If a user asks for "Person London restaurant New York" one possible meaning of the query is that the user is looking for the restaurant in New York visited by people from London. Other permutations of the keywords in the query may generate other possible interpretations.

The second step in answering a keyword query concerns the formulation of an SQL query expressing one of the discovered intended meanings. In a database, semantic relationships between values are modeled either through the inclusion of different attributes under the same table or through join paths across different tables. Different join paths can lead to different interpretations. Consider, for instance, the keyword query "Person

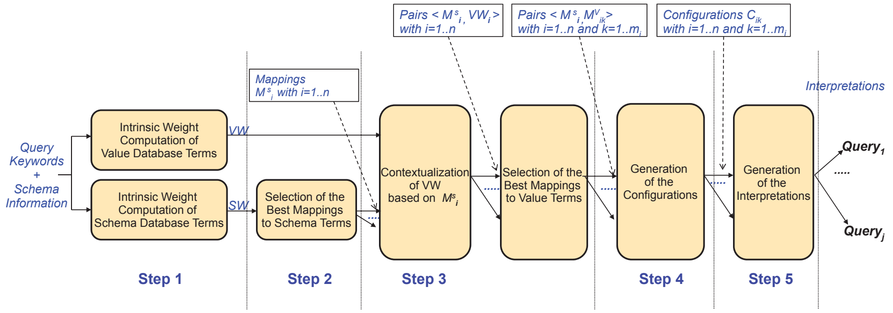


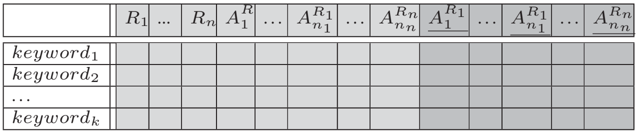
Fig. 2. Overview of the keyword query translation process

USA". One logical mapping is to have the word `Person` corresponding to the table Person and the word `USA` to a value of the attribute Country of the table City. Even when this mapping is decided, there are different interpretations of the keywords based on the different join paths that exist between the tables Person and City. For instance, one can notice that a person and a city are related through a join path that goes through the City attribute referring to the attribute Name in the table City (determining which people in the database are from USA), through another path that is based on the table Hotel (determining which people reserved rooms of Hotels in USA), and also through another path that is based on the table Restaurant (determining which people are reserved a table in an American restaurant).

Finding the different semantic interpretations of a keyword query is a combinatorial problem which can be solved by an exhaustive enumeration of the different mappings to database structures and values. The large number of different interpretations can be brought down by using internal and external knowledge that helps in eliminating mappings that are not likely to lead to meanings intended by the user. For instance, if one of the provided keywords in a query is `` `320-463-1463'' ``, it is very unlikely that this keyword refers to an attribute or table name. It most probably represents a value, and in particular, due to its format, a phone number. Similarly, the keyword `` `Bistro'' `` in a query does not correspond to a table or an attribute in the specific database. Some auxiliary information, such as a thesaurus, can provide the information that the word "bistro" is typically used to represent a restaurant, thus, the keyword can be associated to the Restaurant table.

3 From Keywords to Queries

The generation of *interpretations* (i.e. SQL queries) that most likely describe the intended semantics of a keyword query is based on semantically meaningful *configurations*, i.e. sets of mappings between each keyword and a database term. We introduce the notion of *weight* that offers a quantitative measure of the relativeness of a keyword to a database term, i.e., the likelihood that the semantics of the database term are the intended semantics of the keyword in the query. The sum of the weights of the keyword-database term pairs can form a score serving as a quantitative measure of the likelihood of the configuration to lead to an interpretation that accurately describes the intended



	R_1	...	R_n	A_1^R	$\ldots$	$A_{n_1}^{R_1}$	$\ldots$	$A_{n_n}^{R_n}$	$\underline{A_1^{R_1}}$	$\ldots$	$\underline{A_{n_1}^{R_1}}$	$\ldots$	$\underline{A_{n_n}^{R_n}}$
$keyword_1$													
$keyword_2$													
$\ldots$													
$keyword_k$													

Fig. 3. Weight table with its SW (light) and VW (dark) parts

keyword query semantics. The range and full semantics of the score cannot be fully specified in advance. They depend on the method used to compute the similarity. This is not a problem as long as the same method is used to compute the scores for all the keywords.

The naive approach for selecting the best configurations is the computation of the score of each possible configuration and then selecting those that have the highest scores. Of course, we would like to avoid an exhaustive enumeration of all the possible configurations, and compute only those that give high scores. The problem of computing the mapping with the maximum score without an exhaustive computation of the scores of all the possible mappings is known in the literature as the problem of *Bipartite Weighted Assignments* [5]. Unfortunately, solutions to this problem suffer from two main limitations. First, apart from the mutual exclusiveness, they do not consider any other interdependencies that may exist between the mappings. Second, they typically provide only the best mapping, instead of a ranked list based on the scores.

To cope with the first limitation, we introduce two different kinds of weights: the *intrinsic*, and the *contextual* weights. Given a mapping of a keyword to a database term, its intrinsic weight measures the likelihood that the semantics of the keyword is that of the database term if considered in isolation from the mappings of all the other keywords in the query. The computation of an intrinsic weight is based on syntactic, semantic and structural factors such as attribute and relation names, or other auxiliary external sources, such as vocabularies, ontologies, domains, common syntactic patterns, etc. On the other hand, a contextual weight is used to measure the same likelihood but considering the mappings of the remaining query keywords. This is motivated by the fact that the assignment of a keyword to a database term may increase or decrease the likelihood that another keyword corresponds to a certain database term. As an example, for the keyword query ``Restaurant Name Naples'' expressed on the database in Figure 1, since the keyword ``Naples'' is right next to keyword Name, mapping the keyword Name to the attribute Name of the table Restaurant makes more likely the fact that the keyword Naples is a name value, i.e., should be mapped to the domain of the attribute Name. At the same time, it decreases its relativeness to the other database terms. To cope with the second limitation, we have developed a novel algorithm for computing the best mappings. The algorithm is based on and extends the Hungarian (a.k.a., Munkres) algorithm [4].

A visual illustration of the individual steps in the keyword query translation task is depicted in Figure 2. A special data structure, called *weight matrix* (see Figure 3), plays a central role in these steps. The *weight matrix* is a two-dimensional array with a row for each keyword in the keyword query, and a column for each database term. The value of a cell $[i, j]$ represents the weight associated to the mapping between the keyword i and the database term j. An R_i and $A_j^{R_i}$ columns correspond to the relation R_i and the attribute A_j of R_i, respectively, while a column with an underlined attribute name

$A_j^{R_i}$ represents the data values that may be contained in the column A_j of table R_i, i.e., its domain. Two parts (i.e., sub-matrices) can be distinguished in the weight matrix. One corresponds to the database terms related to schema elements, i.e., relational tables and attributes, and the other one corresponds to attribute values, i.e., the domains of the attributes. We refer to database terms related to schema elements as *schema database terms* (SW), and to those related to domains of the attributes as *value database terms* (VW).

Intrinsic Weight Computation. The first step of the process is the intrinsic weight computation. The output is the populated SW and VW sub-matrices. The computation is achieved by the exploitation and combination of a number of similarity techniques based on structural and lexical knowledge extracted from the data source, and on external knowledge, such as ontologies, vocabularies, domain terminologies, etc. Note that the knowledge extracted from the data source is basically the meta-information that the source makes public, typically, the schema structure and constraints. In the absence of any other external information, a simple string comparison based on tree-edit distance can be used for populating the SW sub-matrix. For the VW sub-matrix the notion of *Semantic Distance* [7] can always be used in the absence of anything else. As it happens in similar situations [10], measuring the success of such a task is not easy since there is no single correct answer. In general, the more meta-information has been used, the better. However, even in the case that the current step is skipped, the process can continue with the weight matrix where all the intrinsic values have the same default value.

Selection of the Best Mappings to Schema Terms. The intrinsic weights provide a first indication of the similarities of the keywords to database terms. To generate the prominent mappings, we need on top of that to take into consideration the inter-dependencies among the mappings of the different keywords. We consider first the prominent mappings of keywords to schema terms. For that we work on the SW sub-matrix. Based on the intrinsic weights, a series of mappings M_1^S,M_2^S, …, M_n^S, of keywords to schema terms are generated. The mappings are those that achieve the highest overall score, i.e., the sum of the weights of the individual keyword mappings. The mappings are partial, i.e., not all the keywords are mapped to some schema term. Those that remain unmapped will play the role of an actual data value and they will be considered in a subsequent step for mapping to value database terms. The selection of the keywords to remain unmapped is based on the weight matrix and some cut-off threshold. Those with a similarity below the threshold remain unmapped. For each mapping M_i^S, the weights of its SW matrix are adjusted to take into consideration the context generated by the mapping of the neighboring keywords. It is based on the observation that users form queries in which keywords referring to the same or related concepts are adjacent. The generation of the mappings and the adjustment of the weights in SW are performed by an adaptation of the Hungarian algorithm. In particular, the algorithm does not stop after the generation of the best mapping to continues to the generation of the second best, the third, etc. Furthermore, some of its internal steps have been modified so that the weight matrix is dynamically updated every time that a mapping of a keyword to a database term is decided during the computation. The output of such a step is an updated weight matrix SW_i and, naturally, an updated score for each mapping M_i^S. Given the updated scores, some mappings may be rejected. The selection is based on a threshold.

There is no golden value to set the threshold value. It depends on the expectations from the keyword query answering systems. The higher its value, the less the interpretations generated at the end, but with higher confidence. In contrast, the lower the threshold value, the more the mappings with lower confidence.

Contextualization of VW and selection of the Best Mappings to Value Terms. For each partial mapping M_i^S of keyword to schema terms generated in the previous step, the mappings of the remaining unmapped keywords to value terms needs to be decided. This is done in two phases. First, the intrinsic weights of the VW sub-matrix that were generated in Step 1 are updated to reflect the added value provided by the mappings in M_i^S of some of the keywords to schema database terms. This is called the process of contextualization of the VW sub-matrix, and it increases the weights of the respective values terms to reflect exactly that. For example, in the keyword query ``Name Alexandria'' assume that the keyword Alexandria was found during the first step to be equally likely the name of a person or of a city. If in Step 2 the keyword Name has been mapped to the attribute Name of the table Person, the confidence that Alexandria is actually the name of a person is increased, thus, the weight between that keyword and the value database term representing the domain of attribute Name should be increased, accordingly. In the second phase, given an updated VW_i sub-matrix, the most prominent mappings of the remaining unmapped keywords to value database terms are generated. The mappings are generated by using again the adapted technique of the Hungarian algorithm. The result is a series of partial mappings M_{ik}^V, with k=$1..m_i$, where i identifies the mapping M_i^S on which the computation of the updated matrix VW_i was based. Given one such mapping M_{ik}^V the value weights are further updated to reflect the mappings of the adjacent keywords to value database terms, in a way similar to the one done in Step 2 for the SW sub-matrix. The outcome modifies the total score of each mapping M_{ik}^V, and based on that score the mappings are ranked.

Generation of the Configurations. As a fourth step, each pair of a mapping M_{ik}^V together with its associated mapping M_i^S is a total mapping of the keywords to database terms, forming a configuration C_{ik}. The score of the configuration is the sum of the scores of the two mappings, or alternatively the sum of the weights in the weight matrix of the elements $[i, j]$ where i is a keyword and j is the database term to which it is mapped through M_{ik}^V or M_i^S.

Generation of the Interpretations. Having computed the best configurations, the interpretations of the keyword query, i.e., the SQL queries, can be generated. The score of each such query is the score of the respective configuration. Recall, however, that a configuration is simply a mapping of the keywords to database terms. The presence of different join paths among these terms results in multiple interpretations. Different strategies can be used to further rank the selections. One popular option is the length of the join path [8] but other heuristics found in the literature [11] can also be used. It is also possible that a same interpretation can be obtained with different configurations. A post-processing analysis and the application of data-fusion techniques [3] can be used to deal with this issue. We adopt a greedy approach that computes a query for every alternative join path. In particular, we construct a graph in which each node corresponds to a database term. An edge connects two terms if they are structurally related, i.e., through a table-attribute-domain value relationship, or semantically, i.e., through a referential integrity constraint. Given a configuration we mark all terms that are part of the range of the configuration as "marked". Then we run a breath-first traversal (that favors shorter paths) to find paths that connect the disconnected components of the graph (if it is possible). Then, the final SQL query is constructed by using the "marked" database terms, and in particular, the tables for its from clause, the conditions modeled by the edges for its where clause and the remaining attributes for its select clause. After that, the process is repeated to find a different interpretation, that will be based on a different join path. The final order of the generated interpretations is determined by the way the different paths are discovered and the cost of the configuration on which each interpretation is based.

4 Conclusion and future work

We described a novel framework for keyword searching in relational databases. In contrast to traditional keyword searching techniques that have access to the actual data stored in the database, our technique uses intensional knowledge such as schema information, semantic knowledge, rules specified by users, and techniques that exploit common values and formats. The work opens many new challenging opportunities and research directions, such as the exploitation of standard modeling practices to enhance the configuration generation process and produce more meaningful configurations.

References

[1] S. Bergamaschi, E. Domnori, and Francesco. Keyword search over relational databases: a metadata approach. In *to appear in SIGMOD*. ACM, 2011.

[2] S. Bergamaschi, E. Domnori, F. Guerra, M. Orsini, R. T. Lado, and Y. Velegrakis. Keymantic: Semantic keyword-based searching in data integration systems. *PVLDB*, 3(2):1637–1640, 2010.

[3] J. Bleiholder and F. Naumann. Data fusion. *ACM Comput. Surv.*, 41(1), 2008.

[4] F. Bourgeois and J.-C. Lassalle. An extension of the Munkres algorithm for the assignment problem to rectangular matrices. *Communications of ACM*, 14(12):802–804, 1971.

[5] R. Burkard, M. Dell'Amico, and S. Martello. *Assignment Problems*. SIAM Society for Industrial and Applied Mathematics, Philadelphia, 2009.

[6] S. Chakrabarti, S. Sarawagi, and S. Sudarshan. Enhancing search with structure. *IEEE Data Eng. Bull.*, 33(1):3–24, 2010.

[7] R. Cilibrasi and P. M. B. Vitányi. The google similarity distance. *IEEE Transactions on Knowledge & Data Engineering*, 19(3):370–383, 2007.

[8] Y. Kotidis, A. Marian, and D. Srivastava. Circumventing Data Quality Problems Using Multiple Join Paths. In *CleanDB*, 2006.

[9] M. Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246. ACM, 2002.

[10] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB Journal*, 10(4):334–350, 2001.

[11] J. X. Yu, L. Qin, and L. Chang. *Keyword Search in Databases*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers, 2010.

Managing and using context information within the PerLa language
(extended abstract*)

F. A. Schreiber, L. Tanca, R. Camplani and D. Viganò

Politecnico di Milano
Dipartimento di Elettronica e Informazione
{schreibe,tanca,camplani}@elet.polimi.it
diego.vigano@mail.polimi.it

Abstract. Self-adaptability in pervasive real-world applications can be achieved by adopting a context-aware middleware. In this paper, we propose a context-management extension to the PerLa language and middleware, which allows for: (i) gathering of data from the environment, (ii) feeding this data to the internal context model and, (iii) once a context is active, acting on the relevant resources of the pervasive system, according to the chosen contextual policy.

Keywords: Pervasive system, context-awareness, hybrid intelligence, context management.

1 Introduction

Pervasive Systems deploy devices which are spatially distributed and possibly mobile, in order to monitor different kinds of physical phenomena for application support. Context-awareness is a property inherent to an autonomic Pervasive System and requires a clear definition of what context is and how the context parameter values can be extracted from the real world. Context can in fact be thought of as "any information that can be used to characterise the situation of an entity" [5,3].

Differently from most of existing approaches, we separate the Operational Pervasive System and its Context Management System in two different layers, while embedding in the same language the functionalities of both: a *context model* [2] allows the representation of context in terms of *observable* entities, which, in their turn, have some *symbolic* representation within the system and some of which correspond to *numerical* values gathered from the environment sensors. Gathering context data from the environment requires a simple interface, possibly based on a declarative approach [11,16], which, on the one side, interacts with the network of highly heterogeneous physical devices and, on the other, is correctly interfaced with the internal, symbolic representation of context. In literature, the context management systems can be classified considering the context

* This work has been funded by the European Commission Programme IDEAS-ERC Project 227077-SMScom.

model employed [3], the software architecture adopted [9] and the method used to query the system [8], also called *Context Query Language (CQL)* [15]. From the point of view of the architecture the choice lies between a centralised and a distributed model. Most of the recent projects adopt a distributed architecture (for example [1,7]), whose advantages lie essentially in a greater scalability of the whole system. In [6] a centralised architecture is instead adopted. By contrast, PerLa offers a distributed architecture which can be directly deployed on capable nodes (or the nearest capable node in case of nodes with reduced computational capability). Another point of analysis is represented by the CQL adopted to query the system. A first approach is represented by the use of classical method calls directly on the deployed middleware. This is partially the case of project [14], which also offers the publish/subscribe paradigm for querying the system. Other projects introduce a relational (e.g.: SQL) intermediate layer [6,10] (along with appropriate mapping mechanisms [12]) or XML-based query languages [13,15]. A final approach is based on RDF languages [1,7]. The PerLa language, instead, approaches the problem in a novel way introducing a unique CQL that allows to gather data at the preferred granularity and to perform actions on the pervasive system simultaneously. Haghighi et al. [8] list a series of characteristics that a CQL should implement: in the same paper it is shown that both RDF and SQL-based languages behave efficiently for context management, while the XML-based languages are more recommended to describe simple data structures rather than complex knowledge. Project [15] has recently improved the state of the art of this type of languages.
In this paper we present an extension of the PerLa language and middleware that overcomes this limitation by introducing a query-based context management system relying on the *Context Dimension Tree (CDT)* context model, introduced in Section 2. First of all the language is extended in order to introduce, create and maintain the CDT formalism into PerLa. On one hand this extension will enable the designer to declare the envisaged contexts, while on the other hand it will provide the possibility to define the actions that have to be performed upon each context activation. In parallel, the middleware architecture is also extended in order to manage the CDT structure, to verify context statuses and to physically perform the corresponding actions. To explain the concepts introduced in this paper we shall use a running example where a wine production process is to be monitored. The attention will be focused on the transport phase of the produced wine, as well as on some of the actors (farmers, oenologists and truck drivers) that are involved in the process. The paper is then structured as follows: in Section 2 the CDT context model is presented. Then, after a short introduction to PerLa, in Section 3 we present the extension to the language syntax used to create and manage contexts. Conclusions and future work are discussed in Section 4.

2 The *CDT* model

In this section we quickly introduce a simplified version of the formalism (i.e.: the *context model* [2]) we adopt to model the environment the pervasive system is operating into. The basic idea is to represent the context in terms of a set of *dimensions*, used to capture different characteristics of the environment, and of the admissible *values* (or *concepts*) that can be assumed by the dimensions. In Figure 1 we present the *context schema* in the form of a *Context Dimension Tree (CDT)* in its graphical notation for the wine monitoring example introduced in Section 1. In particular we use black nodes for the dimensions and white nodes for the values.

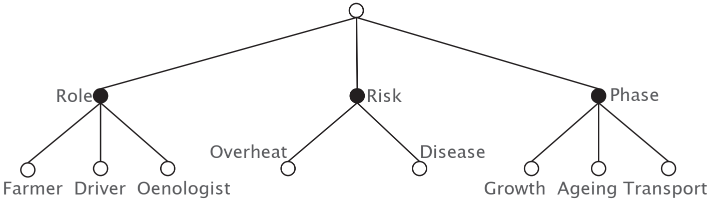


Fig. 1. The CDT schema (for the wine production process)

The `Role` dimension describes the "actors" involved: `Farmer`, `Oenologist` and `Driver`. The `Phase` dimension describes the phases of the wine production process and can assume the values `Growth`, `Ageing` and `Transport`. The third dimension is related to the risks to be kept under control, with the two values (concepts) of `Overheating`, owing to exposure of wine bottles to sunlight, and `Disease` which can affect the grapes. A *context instance* is then a conjunction of propositions, i.e. $Context \equiv \bigwedge_{i,j}(Dimension_i = Value_{i,j})$. As an example we consider the wine production process CDT and we define the *Transport_Monitoring* context. The bottled wine, in fact, must not be kept under direct sunlight for more than a certain amount of time to avoid overheat and a consequent alteration of the wine flavour: $Transport\_Monitoring \equiv (Role = Driver) \wedge (Phase = Transport) \wedge (Risk = Overheat)$. It is worth noticing that not all possible sets of concepts are valid contexts: for instance the dimension `Role` cannot assume simultaneously the `Driver` and `Farmer` values (the children concepts of a dimension are always to be instantiated in mutual exclusion). Invalid contexts are thus ruled out by appropriate constraints [2]. Once the context schema has been defined in terms of *symbolic* observables (e.g. the `Overheat` risk), it is possible to analyse how these can be mapped to *numeric* observables (e.g.: temperature ranges), which are instantiated by retrieving them from the pervasive system. The PerLa[1] system, presented in the next section, allows to perform this important task effectively and efficiently.

3 Managing context through PerLa

3.1 The language

As extensively presented in [16], PerLa is a framework to configure and manage modern pervasive systems. Adopting a data-centric approach [11,4], it relies on a query language using an SQL-like metaphor. PerLa queries allow to retrieve data from the pervasive system, to prescribe how the gathered data have to be processed and stored and to specify the behaviours of the devices. PerLa supports three types of queries: the *Low Level Queries (LLQs)* define the behaviour of a

[1] PerLa website: `http://perlawsn.sourceforge.net/`

(homogeneous group of) node(s), specifying the data selection criteria, the sampling frequency and the elaboration to be performed on sampled data. The *High Level Queries (HLQs)* define the high-level elaboration involving data streams coming from multiple nodes. Such queries specify operations on data streams. Finally the *Actuation Queries (AQs)* provide the mechanisms to change parameters of the devices or to send commands to actuators. A typical [16] PerLa *LLQ*, deployed on a group of sensors, and used to gather data from the field is shown below:

```
...
SELECT ID,temperature,humidity,location_x,location_y
SAMPLING EVERY 1 m
EXECUTE IF EXISTS(temperature) AND is_in_vineyard(location_x,location_y)
REFRESH EVERY 10m
```

PerLa is based on a middleware whose architecture exposes two main interfaces: a high-level interface which allows query injection and a low-level interface that provides plug&play mechanisms to handle devices. However, the PerLa language, in its initial version, does not support the definition and the management of context. To achieve our goal we have extended the original language with a *Context Language (CL)*, whose syntax has been divided into two parts, called *CDT Declaration* and *Context creation*:

CDT Declaration This part allows the user to build the CDT for the specific application:

```
CREATE DIMENSION <Dimension_Name>
  [CHILD OF <Parent_Node>]
  {CREATE CONCEPT <Concept_Name> WHEN <Condition>
  [EVALUATED ON <Low_Level_Query>]}*
```

The *CREATE DIMENSION* clause is used to declare that a new dimension must be added to a CDT, possibly as a child of a concept node (*CHILD OF* clause). Once a dimension has been declared, it is possible to specify the values it can assume, using the *CREATE CONCEPT/WHEN* pair. For each pair the designer must specify the name and the condition for assuming the specified values by means of *numeric* observables that can be measured from the environment. We postpone the explanation of the *EVALUATED ON* clause to the next Subsection, where it plays a fundamental role. The CDT of Section 2 for the wine production process is specified by the following set of statements:

```
CREATE DIMENSION Role
 CREATE CONCEPT Farmer WHEN get_user_role()='farmer'
 CREATE CONCEPT Oenologist WHEN get_user_role()='oenologist'
 CREATE CONCEPT Driver WHEN get_user_role()='driver'
CREATE DIMENSION Risk
 CREATE CONCEPT Disease WHEN get_interest_topic()='disease'
 CREATE CONCEPT Overheat WHEN temperature > 30 AND brightness>0.75;
CREATE DIMENSION Phase
 CREATE CONCEPT Growth WHEN get_phase()='growth'
 CREATE CONCEPT Ageing WHEN get_phase()='ageing'
 CREATE CONCEPT Transport WHEN get_phase()='transport'
```

This CDT is declared using an important feature of the PerLa language: the `get_user_role()`, `get_phase()` and `get_interest_topic()` functions are employed to retrieve context information that cannot be deduced from sensors readings, but have to do with other aspects of the application. This information is typically gathered from some external XML source or database. This clearly highlights how PerLa supports the passage from *symbolic* to *numeric* observable: the `Overheat` *symbolic* value is in fact defined in terms of the `Temperature` and `Brightness` physical quantities (thus *numeric* observables) that can be sampled from the environment using very simple queries.

Context creation This section allows the user to declare a context from a defined CDT and control its activation:

```
CREATE CONTEXT <Context_Name>
ACTIVE IF <Dimension>=<Value> [AND <Dimension>=<Value>]*
ON ENABLE <PerLa_Query>
ON DISABLE <PerLa_Query>     /*one-shot only*/
REFRESH EVERY <Period>
```

The *CREATE CONTEXT* statement is used to create a context instance in PerLa and allows to associate a unique name to it. The *ACTIVE IF* statement translates the $Context \equiv \bigwedge_{i,j}(Dimension_i = Value_{i,j})$ statement of Section 2 into PerLa. This statement is fundamental for the middleware in order to decide if a context is active or not. The actions that must be performed in both these situations must be specified using the *ON ENABLE* clause and are expressed using any type of query exposed in Subsection 3.1. The *ON DISABLE* clause can be coupled only with “one-shot” queries, that is, queries that are executed only once upon deactivation of a context, and thus do not create conflicts with the queries enabled by the next active contexts. The middleware will also perform the necessary controls according to the condition specified in the *REFRESH* clause that completes the syntax. In the next example we show how context management statements and queries/actuation commands on the target system are uniformly mixed in order to achieve a context-aware behaviour. For the *Transport_Monitoring* context we can use the following statements:

```
CREATE CONTEXT Transport_Monitoring
ACTIVE IF Phase = 'transport' AND Role='driver' AND Risk='overheat'
ON ENABLE:
  SELECT temperature, gps_latitude, gps_longitude
  WHERE temperature > 30
  SAMPLING EVERY 120 s
  EXECUTE IF location = 'truck_departing_zone'
SET PARAMETER 'alarm' = TRUE;
ON DISABLE:
DROP Transport_Monitoring;
SET PARAMETER 'alarm' = FALSE;
REFRESH EVERY 24 h;
```

In this example, after creating the context, a very short LLQ is issued: the *SELECT* clause specifies that both temperature and GPS location must be retrieved every two minutes (*SAMPLING EVERY* clause), while the *WHERE* clause allows to filter the sampled values. The *EXECUTE IF* finally deploys the query only on those devices located into the vineyard truck departing zone. This query features also an actuation query (*AQ*) introduced by the *SET PARAMETER* clause and is used to activate an alarm if the risk of overheat becomes real.

3.2 The middleware

The internal structure of the PerLa middleware as described in [16] has been modified to support the Context Language (CL). The new architecture is characterised by the introduction of a *Context Manager (CM)* in charge of: 1) creating and maintaining the CDT; 2) detecting which contexts are active or not in a precise moment; 3) performing the correct actions expressed by the user according to context statuses. In the following we analyse these steps.

Creation of the CDT During this phase all the necessary *numeric* observables (declared using the *CREATE CONCEPT/WHEN* clauses) are retrieved, and the *EVALUATED ON* clause becomes important. In fact, as long as this clause is unemployed, the CM executes a series of independent LLQs in order to retrieve the necessary information from the pervasive system. The designer could be interested in modifying this default behaviour, especially when the environment changes rapidly and the same observable is employed in different concepts (leading thus to some inconsistencies using different LLQs). This clause is useful also to introduce some optimisations (e.g.: discarding some unwanted devices). For example, on the `Overheat` dimension:

```
CREATE CONCEPT Overheat WHEN temperature > 30 AND brightness > 0.75;
 EVALUATED ON:
 SELECT temperature, brightness
 EXECUTE IF location='truck_departing_zone' AND battery>0.7
```

In this example the observables `temperature` and `brightness` are sampled simultaneously using one single query (instead of two independent LLQs). Moreover the query is executed only on those devices that are located in the truck departing zone and whose battery power is enough to operate efficiently (*EXECUTE IF* clause); notice that functional and non-functional data are collected in the same way. Once all the results are available (independently of the presence of the *EVALUATED ON* clause) the system can create a series of tables (one for each dimension with concepts nodes) that contain a column for every attribute expressed in the *CREATE CONCEPT/WHEN* clauses. The table reports also the IDs of the devices that were taken into account during the retrieval phase. Every table entry then represents the actual value (sampled from the environment) and the device that physically produced it. If we consider again the `Overheat` dimension and supposing that the computation of the relative *EVALUATED ON* returned only the 1,3,4 IDs, a table for this dimension could be the following:

ID	temperature	brightness
1	28	0.60
3	31	0.71
4	33	0.80

Table 1. Table for the *Overheat* dimension

Once all the necessary information has been gathered it is possible to evaluate every condition expressed by the *WHEN* clauses used during the CDT declaration. In particular, simply looking up every table, the CM assigns to a CDT concept node the ID(s) of those devices whose sampled values satisfy the condition expressed by the *WHEN* clause of the concept definition. When this phase is concluded the system knows which devices are in the situation described by the concepts of every dimension of the CDT. For example, referring to the `Overheat` in Table 1, the CM can deduce that only sensor number 4 is detecting the risk of overheat since both *temperature>30* and *brightness>0.75* conditions are true simultaneously, while this is not the case of sensors number 3 and 1. However it might happen that a *WHEN* clause be not satisfied by any of the sampled attributes: in this case no ID can be associated to the corresponding CDT concept. Analogously an ID can appear in more than one concept (as long as they are not children of the same dimension): this is the case of modern "intelligent" sensors that can sample more than one attribute simultaneously. With similar computations the CM also selects the concepts that correspond to the results calculated by the static functions, such as `get_user_role()`.

Context detection The purpose of this phase is to discover if one of the declared contexts has become active or has been deactivated. Remember that a context is active if the dimensions that define it assume the values specified by the $Context \equiv \bigwedge_{i,j}(Dimension_i = Value_{i,j})$ statement. Considering also the results of the static functions, the system recognises as active all the contexts whose CDT concepts contain a not-empty device list. In fact, from the CM point of view, if one ID has been associated with a concept it means that, for at least one device, a CDT dimension is currently assuming that value. If this situation is true for every $< Dimension >=< Value >$ used to define a context C then the environment is exactly in the situation expressed by C, and C is considered as active.

Performing context actions Once a context has been recognised as active, the CM simply injects the query specified by the *ON ENABLE* clause into the middleware dedicated components [16]. At this point the execution flow equals the one of any other query that is manually injected into the system, and is thus completely controlled and managed by the middleware dedicated components.

4 Conclusions and future works

In this paper we have presented an extension to the PerLa system in order to support the definition and management of contexts, using the CDT [2] as a formalism to model the environment. Next steps of our work are focused on further system optimizations and, since more than one context could be simultaneously active, on conflict detection and resolution.

References

1. J. Euzenat, J. Pierson and F. Ramparany. *Dynamic context management for pervasive applications*, volume 23. Cambridge Journals, 2008.
2. C. Bolchini, C.A. Curino, E. Quintarelli, F.A. Schreiber, and L. Tanca. Context information for knowledge reshaping. *Intl. Journal of Web Engineering and Technology*, 5(1):88–103, February 2009.

3. C.Bolchini, C.A. Curino, E. Quintarelli, F.A. Schreiber, and L.Tanca. A data-oriented survey of context models. *ACM SIGMOD Record*, 36(4):19–26, Dec 2007.
4. D. Chu, A. Tavakoli, L. Popa, and J. Hellerstein. Enterely declarative sensor network systems. In *Proc. VLDB '06*, pages 1203–1206, 2006.
5. A.K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5(1):4–7, 2001.
6. P. Fahy and S. Clarke. CASS - a middleware for mobile context-aware applications. In *Workshop on Context Awareness, MobiSys*, 2004.
7. T. Gu, H.K. Pung, and D.Q. Zhang. A service-oriented middleware for building context-aware services. *J. Netw. Comput. Appl.*, 28(1):1–18, 2005.
8. P.D. Haghighi, A. Zaslavsky, and S. Krishnaswamy. An evaluation of query languages for context-aware computing. In *Database and Expert Systems Applications, 2006. DEXA '06. 17th International Conference on*, pages 455 –462, 2006.
9. J. Hong, S. Eui-ho, and K. Sung-Jin. Context-aware systems: A literature review and classification. *Expert Syst. Appl.*, 36(4):8509–8522, 2009.
10. G. Judd and P. Steenkiste. Providing contextual information to pervasive computing applications. In *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on*, pages 133 – 142, 23-26 2003.
11. S.R. Madden, M.J. Franklin, J.M. Hellerstein, and W. Hong. TinyDB: an acquisitional query processing system for sensor networks. *ACM Trans. Database Syst.*, 30(1):122–173, 2005.
12. T. McFadden, K. Henricksen, and J. Indulska. Automating context-aware application development. In *UbiComp 1st International Workshop on Advanced Context Modelling, Reasoning and Management*, pages 90–95, 2004.
13. D. Nicklas, M. Grossmann, J. Minguez, and M. Wieland. Adding high-level reasoning to efficient low-level context management: A hybrid approach. In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pages 447–452.
14. H. Peizhao, J. Indulska, and R. Robinson. An autonomic context management system for pervasive computing. In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pages 213 –223, mar. 2008.
15. R. Reichle, M. Wagner, M.U. Khan, K. Geihs, M. Valla, C. Fra, N. Paspallis, and G.A. Papadopoulos. A context query language for pervasive computing environments. In *Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on*, pages 434 –440, 2008.
16. F.A. Schreiber, R. Camplani, M. Fortunato, M. Marelli, and G. Rota. PerLa: A language and middleware architecture for data management and integration in pervasive information systems. *IEEE Transactions on Software Engineering*, (PrePrints), 2011.

Querying Compressed Knowledge Bases in Pervasive Computing

Eugenio Di Sciascio[1], Michele Ruta[1], Floriano Scioscia[1], and Eufemia Tinelli[2]

[1] Politecnico di Bari, Via Re David 200, I-70125 Bari, Italy,
[disciascio, m.ruta, f.scioscia]@poliba.it,
[2] Universitá degli Studi di Bari, Via Orabona 4, I-70125 Bari, Italy,
tinelli@di.uniba.it

Abstract. In the so-called Semantic Web of Things (SWoT), annotated information is tied/derived to/from micro-devices, such as RFID tags and wireless sensors, deployed in an environment. Compression techniques are so needed, due to the verbosity of semantic XML-based languages. Beyond compression ratio, query efficiency is a key aspect for knowledge discovery in mobile ad-hoc scenarios where resources are constrained and topology is unpredictable. This paper proposes a querying schema for OWL knowledge bases, serialized in RDF/XML syntax and homomorphically compressed. The final goal is to allow query evaluation without requiring decompression. Algorithms are presented to prove the feasibility of the proposed approach, while practical examples highlight its usefulness.

1 Introduction and Motivation

In pervasive computing, several factors make information availability as unpredictable: network topology evolution due to node mobility, range limitations and inherent unreliability of wireless communications, node failure due to energy depletion. Thus, approaches based on centralized information storage and management are impractical. Several proposals for collaborative, dynamic resource discovery in ad-hoc networks can be found in literature. In some of them the exploitation of semantics allows to enhance retrieval effectiveness also coping with volatility and unpredictability. The so-called *Semantic Web of Things* (SWoT) aims at the integration of Semantic Web and pervasive computing technologies, in order to associate semantically rich information to real-world objects, locations and events. Data is derived and/or carried by inexpensive, disposable and unobtrusive micro-devices, such as *Radio Frequency IDentification* (RFID) tags and wireless sensors, attached to everyday items or deployed in given environments. Due to power, size and cost constraints, they are usually equipped with little or no processing capabilities, very small storage and short-range, low-throughput wireless links. Data should be extracted and processed by agents on mobile computing devices, through wireless ad-hoc networks.

According to Linked Data best practices, information resources in the Semantic Web should be denoted by dereferenceable URIs (Uniform Resource Identifiers) and annotated in RDF (Resource Description Framework, http://www.w3.

org/TR/rdf-primer/) w.r.t. an RDF Schema (http://www.w3.org/TR/rdf-schema/) or OWL (Web Ontology Language, http://www.w3.org/TR/owl2-overview/) ontology [1]. Language specifications include a standard XML serialization syntax. Adopted knowledge representation models are grounded on formal, logic-based semantics. Query languages, *e.g.*, SPARQL (SPARQL Protocol And RDF Query Language, http://www.w3.org/TR/rdf-sparql-query/), are defined to extract and combine asserted information, while reasoning engines can perform automated inference of knowledge entailed by a given *Knowledge Base* (KB).

One of the most important issues restraining a coherent development of the SWoT vision is the verbosity of the adopted XML-based languages: it is a significant hindrance to efficient storage and transmission of semantic annotations, so that compression techniques become essential. Furthermore, when evaluating encoding algorithms from the SWoT perspective, traditional metrics such as compression ratio and speed are not enough: efficiency and effectiveness of queries on compressed data are critical aspects. Particularly, compression schemes allowing to directly evaluate queries on encoded annotations, without full decompression [2, 3], can be very useful. Unfortunately, so far research has been focused on data extraction from generic structured documents referred to an XML Schema. Hence, main motivation for the present work is that efficient execution of semantic-based queries on compressed KB fragments (ontology segments or resource annotations) can significantly enhance resource discovery capabilities in mobile contexts. So this paper introduces a formal framework for querying OWL Knowledge Bases, serialized in RDF/XML and encoded with *COX (Compressor for Ontological XML-based languages)* [4], which exploits the *homomorphism* property to preserve document structure during compression. Algorithms are defined for the execution of most common elementary queries in the Semantic Web literature. Main contribution is in demonstrating both feasibility and soundness of such general-purpose semantic-based interrogations for on-the-fly knowledge extraction from compressed KBs. Several possibilities are opened for further applied research, devoted to support the integration in high-level query languages (*e.g.*, SPARQL) and inference services by simply and properly combining the proposed query building blocks.

The remaining of the paper is structured as follows. Technical background about KB compression is briefly recalled, before discussing the proposed approach in Section 2, while a case study in Section 3 provides a toy example to highlight usefulness of the proposal. After reporting on relevant related work in Section 4, Section 5 closes the paper.

2 Framework

2.1 Background

Lossless compression is based on substitution of symbols in the input message with code words, according to a statistical model for the input source. *Huffman coding* and *arithmetic encoding* [5] are two fundamental techniques. The former exploits a code table (the *huffman tree*), derived from symbol frequencies.

Widespread universal compression tools *gzip* (http://www.gzip.org/) and *bzip2* (http://bzip.org/) combine Huffman coding with pre-processing input transformations (Lempel-Ziv LZ77 algorithm and Burrows-Wheeler transform, respectively); the latter has better compression rate but it is also slower. In arithmetic encoding, instead, the whole message is represented by a real number in the $[0, 1)$ interval. Disjoint sub-intervals are assigned to all possible symbols, whose length is proportional to symbol frequency. An input message is then mapped to an interval I as follows: at the beginning, $I = [0, 1)$; for each symbol S, I is reduced proportionally to the sub-interval of S. At the end, any value in I will unambiguously encode the sequence of symbols in the message.

Since Semantic Web languages are based on XML syntax, XML-specific algorithms can achieve higher compression rates than universal ones, by exploiting inherent syntactic constraints. An important property is *homomorphism* [2]: homomorphic compression preserves the structure of the original XML data. That may allow to evaluate queries directly on compressed formats, by detecting document pieces which satisfy given query conditions without preliminary decompression of the whole document. *XGrind* [2] and *XPress* [3] are relevant homomorphic compression approaches, adopting Huffman coding and *Reverse Arithmetic Encoding* (RAE), respectively. They achieve high query performance for compressed XML data by virtue of homomorphism; compression rates, though, are lower than the best non-homomorphic XML compression algorithms.

COX (Compressor for Ontological XML-based languages) [4] is adopted here as reference format for querying compressed XML-based semantic annotations. It exploits different solutions to encode data structures (XML tags, attributes) and data (attribute values), in a two-step compression process. For data structures, a RAE variant is used. For attribute values, a dictionary is used to map the most frequent strings to 1-byte codes. COX deals with tag and attribute names in the same way. Attributes are distinguished by a "`@`" prefix. Therefore, in the rest of the section the word "tag" will refer equivalently to a tag or to an attribute.

In the first step, the XML document is parsed and statistics are gathered. After parsing, frequency of each tag name is computed as ratio between the number of occurrences of the tag itself and the total document tags. The $[d, D) = [1.0 + 2^{-7}, 2.0 - 2^{-15})$ interval is then split in disjoint sub-intervals, assigning slightly longer sub-intervals to very rare tags while preserving proportionality with respect to frequency. That avoids encoding errors for tags with a very low frequency. All values representing opening tags fall in the interval $[d, D)$. The interval $[1.0, d)$ is reserved to encode closing tags. Since every possible value is strictly between 1.0 and 2.0, the first byte will always be $\texttt{01111111}_2$ in 32-bit floating point representation, so it can be truncated without loss of information [3]. After the first step a *tag header* is written at the beginning of the output file. It contains a sequence of records composed by: 1 byte for the length of the tag name; the tag name; 3 bytes (after truncation) for encoding the minimum value of the sub-interval related to the tag. The statistic collection for text string frequencies is performed concurrently with the analysis of document structure: strings with both length and frequency higher than heuristic thresholds are encoded. At the end of the first step, a *value header* is written after the tag header. It consists of a sequence of strings, separated by the `ff`$_h$ character. The corresponding codes are single-byte values from `00`$_h$ to `fd`$_h$ and they are assigned to strings in progressive order, hence they can be omitted in the header.

In the second step, the body of the output file is produced. Opening and closing tags, attributes and attribute values are encoded in the same order as they appear in the input document to preserve homomorphism. Each tag T is encoded by applying RAE: as input message, the sequence of tag names is considered, starting with T and going toward its ancestors (hence the adjective "reverse") up to the root XML tag. The sub-interval corresponding to this tag path (named *simple-path*) is computed; then its minimum limit is represented as a 32-bit floating point value and the two central bytes are taken to encode T (the loss of precision due to discarding the least significant byte of the mantissa does not prevent a correct tag identification). Finally, an attribute value is processed as follows: if it belongs to the dictionary produced in the first step, it is replaced by its 1-byte code followed by the delimiter `fe`$_h$, otherwise the string is copied to output, followed by the delimiter `ff`$_h$.

2.2 Querying Approach

In the proposed framework, the classical KB definition $K = \langle \mathcal{T}, \mathcal{A} \rangle$ is adopted, where the TBox $\mathcal{T}$ refers to the ontological knowledge, and the ABox $\mathcal{A}$ specifies the assertional one. The framework deals with KBs in an OWL-DL subset whose characteristics are:

- $\mathcal{T}$ is a *simple TBox*, *i.e.*, a set of *Primitive Concept Specifications* ($A \sqsubseteq B$);
- object properties, data properties and disjoint concepts sets can be defined;
- $\mathcal{A}$ is a role-free ABox, *i.e.*, it is a finite set of individuals defined as instances of a general concept expression C without binary relations between individuals. C can be a conjunction of atomic concepts, unqualified existential quantifications, number restrictions and universal quantifications.

The adoption of a role-free ABox allows to reduce reasoning on assertional knowledge to reasoning on ontological one. Moreover, the selected OWL-DL subset ensures a good trade-off between expressiveness and computational complexity in real applications, as discussed in [6]. Both keyword-based search and a set of path-based queries will be proposed, in order to obtain useful classical inferences on $\mathcal{T}$ and query answering on $\mathcal{A}$. In what follows, it will be shown how, starting from a minimum query set, several other queries can be incrementally built. In fact, the proposed querying approach can be used to cope with non-standard inference services in [7, 6]; deeper discussion is beyond the scope of this work. With reference to TBox reasoning, a set of path-based queries is presented, most of which are exploited in [8]:

- *parents*(A) - it retrieves all the concepts B such that A is direct sub-class of B. Obviously, it is possible to retrieve all the *ancestors* B of A recursively applying the *parent* primitive to B until B is different from Top concept.

```
<rdf:Description rdf:about="http://sisinflab.poliba.it/food.owl#Cavendish">
  <rdf:type>
    <owl:Class>
      <owl:intersectionOf rdf:parseType="Collection">
        <rdf:Description rdf:about="http://sisinflab.poliba.it/food.owl#Banana"/>
        <rdf:Description rdf:about="http://sisinflab.poliba.it/food.owl#Ripe_Product"/>
        <owl:Restriction>
          <owl:onProperty rdf:resource="http://sisinflab.poliba.it/food.owl#Has_Colour"/>
          <owl:someValuesFrom rdf:resource="&owl;Thing"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="http://sisinflab.poliba.it/food.owl#Has_Colour"/>
          <owl:allValuesFrom rdf:resource="http://sisinflab.poliba.it/food.owl#Yellow"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="http://sisinflab.poliba.it/food.owl#Storage_Humidity"/>
          <owl:someValuesFrom rdf:resource="&owl;Thing"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="http://sisinflab.poliba.it/food.owl#Storage_Humidity"/>
          <owl:allValuesFrom rdf:resource="http://sisinflab.poliba.it/food.owl#Medium_Humidity"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="http://sisinflab.poliba.it/food.owl#Storage_Temperature"/>
          <owl:someValuesFrom rdf:resource="&owl;Thing"/>
        </owl:Restriction>
        <owl:Restriction>
          <owl:onProperty rdf:resource="http://sisinflab.poliba.it/food.owl#Storage_Temperature"/>
          <owl:allValuesFrom rdf:resource="http://sisinflab.poliba.it/food.owl#Room_Temperature"/>
        </owl:Restriction>
      </owl:intersectionOf>
    </owl:Class>
  </rdf:type>
</rdf:Description>
```

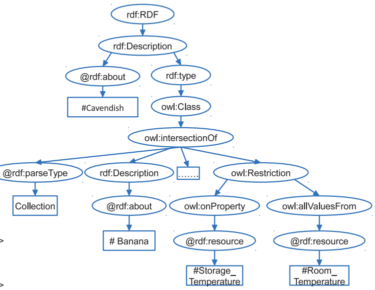


Fig. 1. Instance description in OWL and graph-based representation

- *children*(A) - it retrieves all the concepts B such that B is direct sub-class of A. Also in this case it is possible to retrieve all the *descendants* B of A applying recursively the *children* primitive to B until B is different from $Bottom$ concept.
- *properties*(A) - it retrieves all the properties P such that A is domain of P.
- *leaves*(A) - it retrieves all the concepts B as most specific of A. More formally, $leaves(A) = \{B | B = subClassOf(A) \wedge \neg \exists B' : (B' = subClassOf(A) \wedge B' = subClassOf(B))\}$.

With reference to ABox reasoning, two query models are presented: (i) *entity-based search*, implemented by means of string matching on the required concepts and their descendants, and (ii) *path-based queries*. The former is useful when the domain knowledge organization is unknown. The latter can be considered as a solution to the classical *query answering* problem. Note that a KB instance can be seen as graph partition reflecting the representational model used in COX algorithm (see the example in Figure 1). Some nodes can be distinguished as having the same depth w.r.t. document root (*i.e.*, `rdf:RDF` tag), and some other ones are in "partial order" among them. Hence, with reference to the graph in Figure 1, `owl:Restriction` tag directly precedes `owl:allValuesFrom` tag (*i.e.*, one hop of distance), while `owl:allValuesFrom` and `owl:onProperty` tags are at the same depth level.

The proposed query engine refers to the RDF/XML serialization recommended by OWL 2 language specifications. The support for all syntactic variants of RDF/XML is not explicitly dealt with.

Primitives. The following simple-paths will be referenced in query execution algorithms to find elements in the RDF model. For reader's convenience, they are not reported in reverse order.

P1 $rdf : RDF \rightarrow owl : Class \rightarrow @rdf : about$
P2 $rdf : RDF \rightarrow owl : ObjectProperty \rightarrow @rdf : about$
P3 $rdf : RDF \rightarrow owl : Class \rightarrow rdfs : subClassOf \rightarrow @rdf : resource$
P4 $rdf : RDF \rightarrow owl : ObjectProperty \rightarrow rdfs : domain \rightarrow @rdf : resource$
P5 $rdf : RDF \rightarrow rdf : Description \rightarrow @rdf : about$
P6 $rdf : Description \rightarrow rdf : Type \rightarrow owl : Class \rightarrow owl : IntersectionOf \rightarrow owl : Restriction \rightarrow owl : onProperty \rightarrow @resource$

P7 $rdf : Description \rightarrow rdf : Type \rightarrow owl : Class \rightarrow owl : IntersectionOf \rightarrow$
$owl : Restriction \rightarrow owl : onProperty \rightarrow owl : allValuesFrom \rightarrow$
$owl : Class \rightarrow @rdf : about$

P8 $rdf : Description \rightarrow rdf : type \rightarrow owl : Class \rightarrow owl : intersectionOf$
$\rightarrow rdf : Description \rightarrow @rdf : about$

Query execution is based on a set of primitives for accessing COX compressed documents, whose structure, as said, consists of a tag header $\mathcal{H}_T$, a value header $\mathcal{H}_V$ and a body $\mathcal{B}$. The primitives are listed in Table 1 and explained hereafter. $\mathcal{H}_T$, $\mathcal{H}_V$, $\mathcal{B}$ are supposed to be always accessible. Data complexity characterization is provided, along with required (read-only) accesses w.r.t. input size.

- *lookupTag* searches for a tag name in $\mathcal{H}_T$; if found, it returns its associated interval, else it returns *null*.
- *lookupValue* searches for a string value in $\mathcal{H}_V$; if found, it returns its associated 1-byte code, else it returns the value of the input argument.
- *lookupValueLike* searches for $\mathcal{H}_V$ within strings containing the input argument; it returns the (possibly empty) set of 1-byte codes associated to matching strings.
- *lookupCode* searches for a code in $\mathcal{H}_V$; if found, it returns its associated string, else it returns *null*.
- *computeSimplePath* computes the simple path interval; it uses the arithmetic encoding algorithm described in Section 2.1 and requires a *lookupTag* call for each element in the simple path.
- *findPathsWithValue* takes in input an interval i and a string value v; it gets $c := lookupValue(v)$, then it scans $\mathcal{B}$ to find all occurrences of the simple path encoded by i followed by c; they are returned as positions (in bytes) from the $\mathcal{B}$ beginning.
- *findPathsWithValueLike* is similar to the previous primitive. It takes in input an interval i and a string value v, and it scans $\mathcal{B}$ to find all occurrences of the simple path encoded by i followed by a string containing v; they are returned as positions (in bytes) from the $\mathcal{B}$ beginning.
- *getValuesAfter* takes in input an interval i and a position n; it scans the document from position n, up to the end of the related XML element. It returns a (possibly empty) set of string values following attributes encoded with a value in i.
- *getValuesBefore* scans the document backwards from position n up to the beginning of the nth XML element.

Queries. Algorithm 1 and Algorithm 2 exploit simple-paths $P1$ and $P3$ and COX access primitives to find parents and children of a given class, respectively. Algorithm 3 (resp. 4) calls Algorithm 1 (resp. 2) to find the class ancestors (resp. descendants). In order to find leaves of a given class the previous algorithms have to be exploited, as reported in Algorithm 5. Algorithm 6 uses simple-path $P4$ to look up for a domain relationship between the input class and a property name, then $P2$ to find the property name by scanning the compressed document backwards. Algorithm 7 performs keyword-based search using partial string matching both in the document header and body. Finally, Algorithm 8 finds the ABox individuals that are instances of a class intersection.

Name	Input	Output	Complexity
$lookupTag(t)$	tag name t	interval or $null$	$O(\|\mathcal{H}_T\|)$
$lookupValue(v)$	string value v	code of v or v itself	$O(\|\mathcal{H}_V\|)$
$lookupValueLike(v)$	string value v	(possibly empty) set of codes of strings containing v	$O(\|\mathcal{H}_V\|)$
$lookupCode(c)$	code c between 0 and 253	string at position c in $\mathcal{H}_V$ or $null$	$O(\|\mathcal{H}_V\|)$
$computeSimplePath(P)$	simple path P	interval or $null$	$O(\|P\| \times \|\mathcal{H}_T\|)$
$findPathsWithValue(i, v)$	interval i of simple path, string value v	(possibly empty) set of occurrences, as positions from start of document	$O(\|\mathcal{B}\| + \|\mathcal{H}_V\|)$
$findPathsWithValueLike(i, v)$	interval i of simple path, string value v	(possibly empty) set of occurrences, as positions from start of document	$O(\|\mathcal{B}\|)$
$getValuesAfter(i, n)$	interval i, position n	(possibly empty) set of strings	$O(\|\mathcal{B}\| + \|\mathcal{H}_V\|)$
$getValuesBefore(i, n)$	interval i, position n	(possibly empty) set of strings	$O(\|\mathcal{B}\| + \|\mathcal{H}_V\|)$

Table 1. Access primitives for a COX compressed document

Algorithm 1 $parents(a)$

```
Require: $a$ class name, $P1$ and $P3$ simple-paths
Ensure: $P$ set of parents of $a$
$P := \emptyset$
$i_1 := computeSimplePath(P1)$
$i_2 := computeSimplePath(P3)$
$N := findPathsWithValue(i_1, a)$
for all $n \in N$ do
    $P := P \cup getValuesAfter(i_2, n)$
end for
```

Algorithm 2 $children(a)$

```
Require: $a$ class name, $P1$ and $P3$ simple-paths
Ensure: $C$ set of children of $a$
$C := \emptyset$
$i_1 := computeSimplePath(P3)$
$i_2 := computeSimplePath(P1)$
$N := findPathsWithValue(i_1, a)$
for all $n \in N$ do
    $C := C \cup getValuesBefore(i_2, n)$
end for
```

Algorithm 3 $ancestors(a)$

```
Require: $a$ class name
Ensure: $A$ set of ancestors of $a$
$A := \emptyset$
$P := parents(a)$
$A := P$
for all $p \in P$ do
    $A := A \cup ancestors(p)$
end for
```

Algorithm 4 $descendants(a)$

```
Require: $a$ class name
Ensure: $D$ set of descendants of $a$
$D := \emptyset$
$C := children(a)$
$D := C$
for all $c \in C$ do
    $D := D \cup descendants(c)$
end for
```

Algorithm 5 $leaves(a)$

```
Require: $a$ class name
Ensure: $L$ set of leaves of $a$
$L := \emptyset$
$C := children(a)$
if $C == \emptyset$ then
    $L := L \cup \{a\}$
else
    for all $c \in C$ do
        $L := L \cup leaves(c)$
    end for
end if
```

Algorithm 6 $properties(a)$

```
Require: $a$ class name, $P2$ and $P4$ simple-paths
Ensure: $P$ list of properties having $a$ as domain
$P := \emptyset$
$i_1 := computeSimplePath(P4)$
$i_2 := computeSimplePath(P2)$
$C := ancestors(a) \cup a$
for all $c \in C$ do
    $N := findPathsWithValue(i_1, c)$
    for all $n \in N$ do
        $P := P \cup getValuesBefore(i_2, n)$
    end for
end for
```

3 Case Study

In order to clarify how the proposed framework works and how the different query models can be used, a simple case study in RFID supply chain management is

Algorithm 7 *keyword − based_search*$(A_1, \ldots, A_n)$

```
Require: $a_1, \ldots, a_n$ names to search, $P1$ simple path
Ensure: $C$ set of classes syntactically similar to $a_1, \ldots, a_n$
$C := \emptyset$
$i := computeSimplePath(P1)$
for $k = 1$ to $n$ do
    $V := lookupValueLike(a_k)$
    for all $v \in V$ do
        if $v! = null$ then
            $n := findPathsWithValue(i, v)$
            if $n! = \emptyset$ then
                $C := C \cup \{lookupCode(v)\}$
            end if
        else
            $n := findPathsWithValueLike(i, v)$
            if $n! = \emptyset$ then
                $C := C \cup \{v\}$
            end if
        end if
    end for
end for
```

Algorithm 8 *entity − based_search*$(a_1, \ldots, a_n)$

```
Require: $a_1, \ldots, a_n$ classes names, $P5$ and $P8$ simple paths
Ensure: $Ins$ list of individuals instance of the intersection of $a_1, \ldots, a_n$
$Ins := \emptyset$
$i_1 := computeSimplePath(P8)$
$i_2 := computeSimplePath(P5)$
for $1 = 1$ to $n$ do
    $C_i := descendants(a_i) \cup \{a_i\}$
    for all $c \in C_i$ do
        $N := findPathsWithValue(i_1, c)$
        for all $n \in N$ do
            $Ins_i := Ins_i \cup getValuesBefore(i_2, n)$
        end for
    end for
end for
$Ins := Ins_1 \cap \ldots \cap Ins_n$
```

considered, where each RFID tag stores a compressed semantic annotation of the product/stock it is attached to. In [6], backward-compatible extensions of EPCglobal RFID tag data structure were devised to accommodate an RDF product description, that could be read by means of standard RFID reader-tag air interface protocol. However, the case study concerns semantic-based queries upon KB in compressed COX format at a logical level, thus it can be applied to any physical data storage model.

Let us consider a large distribution warehouse, receiving several kinds of products manufacturers. A truck fleet is used to deliver products toward sale points. Each truck is endowed with a mobile computing device with embedded semantic-enabled RFID reader and on-board query processing capabilities. When a product is loaded into the truck storage compartment, the reader extracts the compressed annotation from the RFID tag of the product, and queries it in order to check: (i) if the product type is compatible with those the truck is allowed to transport and (ii) if characteristics of the storage compartment are adequate for the product (*e.g.,* lighting, humidity, temperature and any special storage or security equipments). Any incompatibility would likely indicate an error in truck assignment or product routing within the warehouse, hence it must be discovered immediately, through on-the-fly semantic query processing.

A stock S of Cavendish bananas is loaded on a truck T, which is allowed to transport fruit. Storage compartment of T provides no thermostat, hence it can only keep products at room temperature. S has an RFID tag with the compressed semantically annotated product description, expressed w.r.t. a suitable product ontology. Let us suppose that the product annotation coincides with Figure 1 and that the following assertions are in the reference ontology; Notation3 (http://www.w3.org/DesignIssues/Notation3.html) is adopted here for reader's convenience:

Algorithm 9 $checkUniversalRestriction(p, f)$

```
Require: $p$ property, $f$ atomic filler, $P6$, $P7$ simple-paths
Ensure: return $true$ if the individual contains a $\forall p.f$ restriction, $false$ otherwise
 1: $i_1 := computeSimplePath(P6)$
 2: $i_2 := computeSimplePath(P7)$
 3: $D := descendants(f) \cup \{f\}$
 4: $N := findPathsWithValue(i_1, p)$
 5: for all $n \in N$ do
 6:     $V := getValuesAfter(i_2, n)$
 7:     for all $v \in V$ do
 8:         if $v \in D$ then
 9:             return $true$
10:         end if
11:     end for
12: end for
13: return $false$
```

```
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix o: <http://sisinflab.poliba.it/food.owl#> .
o:Banana rdfs:subClassOf o:Fruit .
o:Uncontrolled_Temperature rdfs:subClassOf o:Temperature .
o:Room_Temperature rdfs:subClassOf o:Uncontrolled_Temperature .
o:Storage_Temperature rdfs:range o:Temperature .
```

Entity-based and keyword-based search - According to Algorithm 8, the computing device embedded in the truck runs $entity - based\_search(Fruit)$ on the description of $S.\ Cavendish$ is described as an instance of $Banana$, which is subclass of $Fruit$, hence it is returned. It is useful to point out that a keyword-based search can support the selection of suitable ontology classes before an entity-based search.

Path-based search - When known the KB structure and organization, it is possible to compose more complex queries for instance retrieval. In the above example, the device embedded in the truck has to check if the product description is compliant with the $\forall Storage\_Temperature.Uncontrolled\_Temperature$ restriction. To do so, Algorithm 9 is executed with $p = Storage\_Temperature$ and $f = Uncontrolled\_Temperature$. Intervals for simple-paths are computed first, and the set D of descendants of f is extracted from the TBox. Then the universal restriction is searched in line 4: if it is found and its filler belongs to D, the constraint is satisfied and function returns $true$. Notice that expanding the search for descendants of the filler, makes the check semantic-based rather than purely syntactic. In the example, checking the individual in Figure 1 returns a positive answer, because $Room\_Temperature$ is a subclass of $Uncontrolled\_Temperature$. Other constraints can be verified in a similar fashion; as an optimization, computed intervals for simple-paths can be cached for reuse during query processing on the same compressed RDF annotation.

A partial implementation of the query engine has been performed, including the primitives in Table 1. Experiments were executed on a notebook with Intel Core 2 Duo CPU (2.0 GHz clock frequency), 3 GB RAM and Ubuntu 10.04 OS (Linux 2.6.32 kernel version). Algorithm 9 was tested on a KB with 175 concepts, 11 roles and 15 instances (original OWL size is 114 kB, COX compressed size is 31 kB). Average execution time was 460 ms and main memory usage peak was 1.51 MB. In a further test, 260 *findPathsWithValue* primitives were executed with pseudo-random arguments: Figure 2 reports the distribution of times. These preliminary results suggest that the approach is viable. Since the implementation is not complete, comparative tests w.r.t. other query engines cannot be reported at this time.

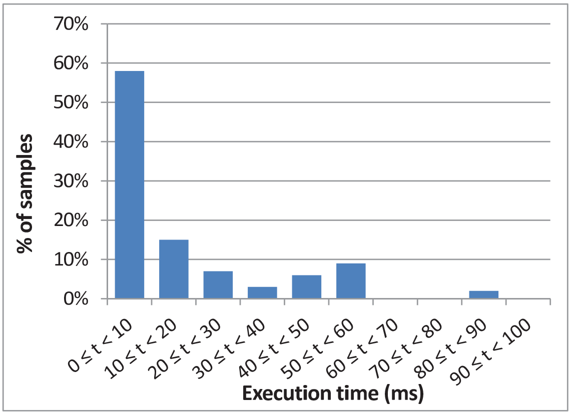


Fig. 2. Execution time of *findPathsWithValue* primitive

4 Related Work

With reference to XML-based languages, several tools supporting efficient querying over compression schemes exist. *XGrind* [2] can perform (i) exact-match and prefix-match queries directly on compressed values and (ii) range and partial-match queries on values decompressed on-the-fly. *XPress* [3] exploits RAE to improve the path-based queries. XPress query engine converts a label path expression into a sequence of intervals. Then, by using this sequence, the query executor checks whether the encoded values of XML tags are in a given interval of the sequence or not. *XQueC* [9] exploits indexing and XML storage strategies since it is focused on search speed rather than compression efficiency. The above tools execute path-based queries expressed in XPath (http://www.w3.org/TR/xpath/), which allow syntactic match of document elements and are strictly tied to the XML Schema for the particular document. Therefore, it is not possible to reuse an existing approach for semantic-based queries.

In known strategies for storing and querying RDF annotations, data structures and optimizations are focused on a database perspective [8]. The Semantic Web community has generally used traditional database systems [10]. As a consequence, most of the RDF query processing techniques are based on database query processing and optimization techniques, mainly focused on compression [11] and indexing approaches [12, 13]. Nevertheless, all these technologies do not cope with mobile computing issues. An interesting exception is the MQuery [14] framework for ubiquitous computing. It creates a compressed index of RDF graphs for improving context-aware retrieval, according to the idea that a mobile user wants to access specific data depending on given situations. The main drawback w.r.t. the approach proposed here is the limited flexibility and expandability, as MQuery provides a pre-defined query interface for selecting only from four possible interrogations.

Studies on the above works have suggested main query models expected by semantic-based applications: (i) full-text search, *i.e.*, keyword or string matching; (ii) queries based on data structure, *i.e.*, path-based and structure-based queries; (iii) a combination of them. As a consequence, the presented framework included both keyword-based search and a set of path-based queries.

5 Conclusion

A framework has been presented for querying knowledge bases expressed in OWL, serialized in RDF/XML and processed with a homomorphic compression. It is particularly suitable for scenarios dipped in the Semantic Web of Things vision. Primitives for querying compressed semantic annotations have been devised and used to perform interrogations on both the TBox and KB instances. The implementation of the framework in a software tool is ongoing. It will provide the needed insight into performance benefits and costs, allowing to evaluate possible optimizations. Moreover, it will support the development of a semantic-based query and reasoning engine for compressed KBs, by exploiting basic queries to implement high-level query languages and inference services commonly used in the Semantic Web.

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. Int. J. Semantic Web Inf. Syst. **5**(3) (2009) 1–22
2. Tolani, P., Haritsa, J.: XGRIND: A Query-friendly XML Compressor. In: Proc. of the 18th Int. Conf. on Data Engineering (ICDE.02), IEEE (2002) 225234
3. Min, J., Park, M., Chung, C.: A compressor for effective archiving, retrieval, and updating of XML documents. ACM Transactions on Internet Technology **6**(3) (2006) 223–258
4. Scioscia, F., Ruta, M.: Building a Semantic Web of Things: issues and perspectives in information compression. In: Semantic Web Information Management (SWIM'09). In Proc. of the 3rd IEEE Int. Conf. on Semantic Computing (ICSC 2009), IEEE Computer Society (2009) 589–594

5. Witten, I., Neal, R., Cleary, J.: Arithmetic coding for data compression. Communications of the ACM **30**(6) (1987) 520–540
6. Di Noia, T., Di Sciascio, E., Donini, F.M., Ruta, M., Scioscia, F., Tinelli, E.: Semantic-based Bluetooth-RFID interaction for advanced resource discovery in pervasive contexts. Int. Jour. on Semantic Web and Information Systems **4**(1) (2008) 50–74
7. Ruta, M., Zacheo, G., Grieco, L.A., Di Noia, T., Boggia, G., Tinelli, E., Camarda, P., Di Sciascio, E.: Semantic-based Resource Discovery, Composition and Substitution in IEEE 802.11 Mobile Ad Hoc Networks. Wireless Networks **16**(5) (2010) 1223–1251
8. Christophides, V., Plexousakis, D., Scholl, M., Tourtounis, S.: On labeling schemes for the Semantic Web. In: The 12th Int. Conf. on World Wide Web, ACM (2003) 544–555
9. Skibiski, P., Swacha, J.: Combining Efficient XML Compression with Query Processing. In: Advances in Databases and Information Systems. Volume 4690. Springer Berlin / Heidelberg (2007) 330–342
10. Sakr, S., Al-Naymat, G.: Relational processing of RDF queries: a survey. SIGMOD Rec. **38** (June 2010) 23–28
11. Atre, M., Chaoji, V., Zaki, M.J., Hendler, J.A.: Matrix Bit loaded: a scalable lightweight join query processor for RDF data. In: The 19th Int. Conf. on World Wide Web, ACM (2010) 41–50
12. Delbru, R., Toupikov, N., Catasta, M., Tummarello, G.: A node indexing scheme for web entity retrieval. In: The Semantic Web: Research and Applications. Volume 6089. (2010) 240–256
13. Zhang, S., Yang, J., Jin, W.: SAPPER: Subgraph Indexing and Approximate Matching in Large Graphs. Proc. of the VLDB Endowment **3**(1) (2010) 1185–1194
14. Zhang, Y., Zhang, N., Tang, J., Rao, J., Tang, W.: Mquery: Fast graph query via semantic indexing for mobile context. In: The 2010 IEEE/WIC/ACM Int. Conf. on Web Intelligence and Intelligent Agent Technology - Volume 01, IEEE Computer Society (2010) 508–515

Context-aware data tailoring through Answer Set Programming [*]

Extended Abstract

Angelo Rauseo, Davide Martinenghi, and Letizia Tanca

Politecnico di Milano - Dipartimento di Elettronica e Informazione, Milan, Italy
rauseo@elet.polimi.it, martinen@elet.polimi.it, tanca@elet.polimi.it

Abstract. In this paper we describe a technique for *context-aware data tailoring* by means of Answer Set Programming (ASP). We use ASP techniques to describe and generate the feasible contexts compatible with a context specification structure called Context Dimension Tree. We define suitable context-dependent views that, as soon as a specific feasible context is selected, retain only those data that are meaningful with respect to the context.

1 Introduction

In a world of great complexity and abundance of varied data and datasets, it is important to remove, from the answers we seek, all data that are not really interesting to the *context* in which they are used and thus do not constitute real information. The process of removing this "out-of-context noise" is commonly referred to as *context-aware data tailoring* [1].

In the literature, a context-aware perspective has been applied to manage: (i) the capability to adapt content presentation to different channels or to different devices, like in CSCP [2] or MAIS (http://www.mais-project.it), which configure the software on board of a mobile device based on presentation, device characteristics and available channel; (ii) agreement and shared reasoning among peers with the aim of building smart environments, like in CoBra [3], where an agent-based architecture supports context-aware computing in physical spaces (e.g., living rooms, vehicles, corporate offices and meeting rooms); (iii) modeling what the user is currently doing [4] or his/her physical situation [7]; (iv) making user interaction implicit by adapting information to user's needs, like in QUALEG [10],with context-aware service, or, in general, discovery in pervasive environments; (v) managing knowledge chunks: with an information management perspective, [9, 8] extend the relational model to deal with context introducing *facets* of the data under different contexts.

Our work falls into this last category, since in this paper we use Answer Set Programming (ASP) techniques with the aim of tailoring the data to present to

[*] This research is partially supported by the European Commission, Programme IDEAS-ERC, Project 227977-SMScom. D. Martinenghi acknowledges support from the Search Computing (SeCo) project, Programme IDEAS-ERC.

users only the interesting fragments. We adopt a context representation model, called *Context Dimension Tree (CDT)* [1], that allows designers to characterize the contexts which are significant – and allowed – in a given scenario. Using ASP we are able to retain the orthogonality of context modeling with respect to data management, while adopting the same framework as for data representation. Building on preliminary work presented in [6], we represent contextual information via disjunctive logic programs using the approach of ASP: given a program representing a CDT, each feasible context is represented by a model of the program. When extensional data are added, data tailoring is then obtained by extending the program with suitable views, defined by the designer, that indicate which parts of the data should be retained depending on the available contextual information.

Overall, the main contribution of this work is the design of an ASP framework for supporting data tailoring in context-aware systems. The advantage of this framework is that it serves as a natural way of defining situational, dynamic user views over a global database schema.

The rest of the paper is organized as follows. In Section 2, after introducing the notion of context, we define CDTs as the main tool for modeling context. In Section 3, we describe our representation of feasible contexts with respect to a CDT by disjunctive logic programs. In Section 4, we show how to retain only those data that are relevant for a given context by adding suitable views to the program. Finally, in Section 5 we conclude.

2 Context modeling

The *Context Dimension Tree*– formally defined in [1] –, is an orthogonal and extensible context representation tool that can be used not only in context-aware data tailoring, but also for different applications like service adaptivity or sensor-query configuration.

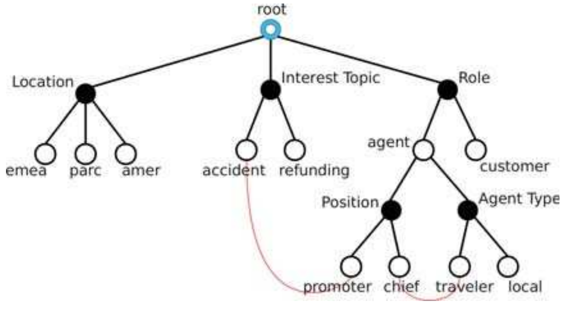


Fig. 1. A Context Dimension Tree (CDT) designed to support a car insurance company

The CDT, an example of which is reported in Figure 1, is a tree-shaped structure composed of two kinds of nodes: *black nodes*, which represent *context* *dimensions*, and *white nodes*, which represent *context values*. The hierarchical nature of the CDT grants different "levels of abstraction" to represent contexts. Context dimensions (black nodes) model different perspectives from which the domain of interest can be seen with respect to the user, the system and their interactions. The values these dimensions can take are represented as white nodes. A context is obtained as a set of dimension values, thus of white nodes. In order for the context to be projected over the data, a view is assigned to each context, thus contextual views should be designed which only select the portion of the data that is interesting with respect to the context of use.

The CDT reported in Figure 1 represents the contexts designed for tailoring the data of a car insurance company. The first-level dimensions (`Location`, `Interest Topic`, `Role`) determine, by means of their directly attached values, the main classification of data and users associated with them, while second-order dimensions attached to the `agent` value (`Position`, `Agent Type`) provide a more detailed specification to the agent figure when needed.

For instance, in Figure 1, the context {`promoter, traveler, emea`} represents the perspectives of a promoter, travel agent in EMEA region, while the context {`chief, amer`} represents a chief agent in the AMERICAS region. Both interact with the system and require access to meaningful data with respect to their situations. Contexts can thus be built from the CDT by appropriately assembling sets of context values. In order to be *valid*, a context built from the CDT has to satisfy the following two **Validity Properties**:

1. A context must neither contain two values that are children of the same parent dimension, nor two that descend from a common ancestor dimension but not from a common ancestor value other than `root`.
2. A context must not contain both a value and its ancestor value(s).

For example {`customer, promoter`} and {`customer, agent`} violate validity property 1. Validity property 2 is violated by the context {`agent, promoter`}.

The basic tree structure of the CDT is also enriched with the support for the definition of application constraints. The designer can annotate the CDT, specifying forbidden associations of value nodes within the same context by means of lines connecting pairs of white nodes. In the example CDT in Figure 1 we find two such restrictions: the values `accident` and `promoter` cannot be together in the same context (*promoter agents do not have access to information on accidents*); the same holds for `chief` and `traveler` (*a chief agent never travels*). The use of constraints allows us to define a *forbidden context* as a context whose set of values contains at least one of the forbidden value pairs.[1]

A *feasible context* is a context that is a valid context and is not a forbidden context. The association of a feasible context with the corresponding data is established by the following notion: the *relevant area* associated with a context C is the data view considered by the designer as interesting for a user or an

[1] Note that forbidden contexts are strictly application-dependent, while the validity properties defined before are independent of the single application at hand, and rather refer to structural properties of the CDT.

application which is in context *C*. The notion of relevant area is the actual implementation of data tailoring, and will be defined by means of a disjunctive logic program. The designer is responsible for the definition of the appropriate CDT with respect to the modeled domain and application needs, along with the relevant areas associated with the contexts generated from that CDT.

3 Context representation and generation through ASP

We now provide ASP-compliant description guidelines for the CDT model. Once the problem of representing contexts has been solved with the CDT, we address the main objectives of this paper: *i)* how to generate the *feasible contexts* of a CDT so that they will be available for use, and *ii)* how to define *relevant areas* and associate them with different contexts.

We use the DLV system [5] as our main development tool and report below the DLV representation of part of the example CDT in Figure 1 as a set of facts. Both values and dimensions are implemented by the definition of predicates: `value(Value_node)` the former, `dimension(Dimension_node)` the latter.

```
dimension(location).         value(root).   value(amer).
dimension(interest_topic).   value(parc).   ...
```

The following facts specify examples of the direct dimension-to-value connections using the `dim2val(D, V)` predicate

```
dim2val(location, parc).
dim2val(interest_topic, accident). ...
```

Once the direct connections have been provided to the system, the tree structure also requires to define the order relationship among both dimension and value nodes. Below we describe the connections between each value and its subdimensions, and their inheritance via `val2dim(V, D)` and `val2dim_path(V, D)`:

```
val2dim(agent, position). val2dim(agent, agent_type).
...
val2dim_path(V, D) :- val2dim(V, D).
val2dim_path(V, D) :- val2dim(V, D1), dim2val(D1, V1), val2dim_path(V1, D).
```

The `dim2val_path(D, V)` predicate determines which value `V` is a descendant of a dimension `D`:

```
dim2val_path(D, V) :- dim2val(D, V).
dim2val_path(D, V) :- dim2val(D, V1), val2dim(V1, D1), dim2val_path(D1, V).
```

The `common_ancestor_val(V1, V2)` and `common_ancestor_dim(V1, V2)` hold if values `V1` and `V2` have a common ancestor value or dimension, respectively:

```
sub_value(Vu, Vd) :- val2dim(Vu, D), dim2val_path(D, Vd).
common_ancestor_val(V1, V2) :- sub_value(V, V1), sub_value(V, V2), V! = root.
common_ancestor_dim(V1, V2) :- dim2val_path(D, V1), dim2val_path(D, V2).
```

Each feasible context corresponds to a model of the disjunctive logic program that encodes the definition, properties and constraints of the CDT. This is the natural way of approaching any problem in ASP, in which the multiple solutions of a problem (here, the feasible contexts) correspond to the multiple models of the program. The values (white nodes) composing a context are here represented as atoms with predicate `context_elem` in the model. As usual in ASP, the choice of whether a model includes or excludes an element is made via the "guessing phase": $\texttt{context\_elem(X)} \vee -\texttt{context\_elem(X)}$:- $\texttt{value(X)}, \texttt{not root\_node(X)}$. In particular, the rule indicates that any value that is not the root is either a context element or not a context element (but not both). Mutual exclusion is granted by the use of the so-called *true negation*, indicated in the DLV syntax with a minus symbol ("−") preceding a predicate name. (Standard *negation-as-failure*, instead, is indicated with the "`not`" keyword.) True negation is basically interpreted as part of the predicate name, with the implicit addition of a constraint in denial form stating mutual exclusion with the non-negated predicate, like the following one: :- $\texttt{context\_elem(X)}, -\texttt{context\_elem(X)}$.

Another standard technique is used to enforce non-emptiness of a feasible context: the "`non_empty_context`" flag is on whenever at least one context element is present, and the constraint below imposes, via a sort of double negation, that the flag be on.

```
non_empty_context :- context_elem(X).
:- not non_empty_context.
```

The validity properties defined in Section 2 must now be enforced by means of constraints over context elements:

Validity property 1

$$\text{:- } \texttt{context\_elem}(\texttt{V}_1), \texttt{context\_elem}(\texttt{V}_2), \texttt{V}_1 != \texttt{V}_2, \texttt{dim2val}(\texttt{D}, \texttt{V}_1), \texttt{dim2val}(\texttt{D}, \texttt{V}_2).$$
$$\text{:- } \texttt{context\_elem}(\texttt{V}_1), \texttt{context\_elem}(\texttt{V}_2), \texttt{V}_1 != \texttt{V}_2, \texttt{common\_ancestor\_dim}(\texttt{V}_1, \texttt{V}_2), \texttt{not common\_ancestor\_val}(\texttt{V}_1, \texttt{V}_2).$$

Validity property 2

$$\text{:- } \texttt{context\_elem}(\texttt{V}), \texttt{context\_elem}(\texttt{V}_\texttt{u}), \texttt{sub\_value}(\texttt{V}_\texttt{u}, \texttt{V}).$$

Similarly, we declare the application-dependent constraints to enforce the exclusion of forbidden contexts

```
:- context_elem(chief), context_elem(traveler).
:- context_elem(accident), context_elem(promoter).
```

4 Associating context with data

At runtime, the feasible context(s) obtained with the techniques shown in the previous section is selected as the *active context(s)*. The selection can be made either manually – by a designer or the user him/herself – or automatically, by a *context manager* that uses extra knowledge to infer the proper active context.

The program described in Section 3 is enriched with *context_elem* facts specifying the active context and with facts for all the extensional relations of the dataset to which the contextualization is applied. When the user, or the application, requires access to one of these extensional relations, the system will instead provide the contextual view over that relation, corresponding to the active context.

The first step we describe to obtain data tailoring is the design-time creation of specialized views over the dataset, which we will call *partial views* and which associate each context element with a portion of the dataset. Starting from partial views, complex *contextual views* are automatically built, which tailor data associating contexts with relevant areas.

4.1 Partial views

Partial views have to be defined at design time in tight association with the development of the CDT they refer to. Each partial view will be linked to a particular value node from the CDT and it will return a (possibly little) fragment of the dataset which has been recognized as interesting by the designer.

In the CDT model, value nodes placed at different levels provide different abstraction granularities ; accordingly, in general relevant areas related to values placed high in the tree will include, often strictly, relevant areas corresponding to values at deeper levels. The partial view for a relation r has the same schema as the original relation it is built upon, plus an additional argument indicating the context element.

Consider a database for the insurance domain, an excerpt of whose schema follows:

```
customer(Id, FName, Country, RiskLevel, Gender)   refunding(InsId, Date, Amount, Cause)
insurance(Id, ContractCode, AgentId, CustId)      geo_area(Area, Country)
agent(Id, FullName, Country, Grade, TravelStatus) ...
```

We show an example of partial view for the table `customer`:

%%% filtering on the `Interest Topic` *refunding*
p_view$_{\text{customer}}$(CId, FName, Country, Risk, Gen, refunding) :-
customer(CId, FName, Country, Risk, Gen), insurance(InsId, _, _, CId),
refunding(InsId, _, _, _), context_elem(refunding).

%%% filtering on the `Location` through a variable
p_view$_{\text{customer}}$(Id, FName, Country, Risk, Gender, Location) :-
customer(Id, FName, Country, Risk, Gender), geo_area(Location, Country),
context_elem(Location).

Here, we have conditions used to (i) use insurance ID, with the `insurance` and `refunding` relations, to retain those customers with an insurance with refunding procedures, and (ii) specify the right location using `geo_area` from the dataset, which associates areas with countries, in the partial view tailoring the location. Note that the first partial view refers specifically to the `refunding` value of the `InterestTopic`, while for the second partial view we produce a different partial view for each location by using a variable (`Location`).

4.2 Contextual views

We show now the pattern for defining a contextual view of the relation *customer*, but the technique is general and is used to build all the contextual views, each as intersection of the partial views associated with the actual context elements.

$\texttt{c\_view}_{\texttt{customer}}$(Id, FName, Country, Risk, Gender) :-
  customer(Id, FName, Country, Risk, Gender),
  not $-\texttt{c\_view}_{\texttt{customer}}$(Id, FName, Country, Risk, Gender).
$-\texttt{c\_view}_{\texttt{customer}}$(Id, FName, Country, Risk, Gender) :-
  customer(Id, FName, Country, Risk, Gender), context_elem(CE),
  not $\texttt{p\_view}_{\texttt{customer}}$(Id, FName, Country, Risk, Gender, CE).

The first rule indicates that a tuple is transferred from *customer* to the view unless the tuple is known not to be part of the view. The second rule states that a tuple from *customer* is known not to be part of the view if there is some context element such that the tuple does not belong to the corresponding partial view. Therefore, a tuple is retained in the contextual view if and only if it is in the partial view corresponding to each context element of the context currently active, which gives the required semantics of a contextual view as intersection of the partial views of the context components.

Query:
  q(FName) :- $\texttt{c\_view}_{\texttt{customer}}$(_, FName, _, _, _).

Relations:
  % customer(Id, FName, Country, RiskLevel, Gender)
  customer(0, gordon_brown, uk, high, m).
  customer(1, barack_obama, us, moderate, m).
  customer(8, albert_a_gore, us, low, m).
  % geo_area(Area, Country)
  geo_area(emea, uk).   geo_area(amer, us).
  % insurance(InsId, ContractCode, AgentId, CId)
  insurance(in1, c4, uk1, 0).   insurance(in2, c3, us2, 1).
  insurance(in9, c4, us1, 8).
  % refunding(InsId, Date, Amount, Cause)
  refunding(in1, 20100606, 5000, bonus).   refunding(in2, 20100809, 5000, bonus).

Context elements:
  context_elem(amer).   context_elem(refunding).

Answer:
  {q(barack_obama)}

Table 1. Query selecting customer names relevant to the active context

Once a context is active, any query on the relations in the dataset will be redirected over the contextual views, which have the same schemata as the original relations, so that only the interesting or permitted data will be available in the answer. Let us assume that the active context consists of the elements {amer,refunding}, which are therefore part of the program as context_elem facts. In the example reported in Table 1, the customer relation is replaced in the query by the $\texttt{c\_view}_{\texttt{customer}}$ predicate. In this particular context, only the data related to customers who requested a refunding procedure in the AMERICAS region will be presented to the user. The query answer consists of the intersection of those sets of names corresponding to the persons selected by the partial view for the customer relation related to the context element amer (i.e., barack_obama and albert_a_gore), and refunding (i.e., barack_obama and gordon_brown).

5 Conclusions

Thanks to the native multi-model nature of Answer Set Programming, it has been possible to realize a straightforward and uniform definition of contexts and context-aware views. Contexts have been defined keeping in mind a data-oriented attitude, and they have been generated, orthogonally with respect to the data.

Future developments will deal with: (i) address contextualization also beyond the task of data tailoring, and (ii) support run-time modifications and changing of the CDT, needed to deal with a fully dynamic ubiquitous system.

References

1. C. Bolchini, C. Curino, E. Quintarelli, F. A. Schreiber, and L. Tanca. Context information for knowledge reshaping. *Int. Journal on Web Engineering and Technology*, 5(1):88–103, 2009.
2. S. Buchholz, T. Hamann, and G. Hübsch. Comprehensive structured context profiles (CSCP): Design and experiences. In *Proc. of 1st Intl Work. on Context Modelling and Reasoning*, pages 43–47, 2004.
3. H. Chen, T. Finin, and A. Joshi. An intelligent broker for context-aware systems. In *Proc. of Intl Conf on Ubiquitous Computing - Poster Session*, pages 183–184, 2003.
4. M. Kaenampornpan and E. O'Neill. An intergrated context model: Bringing activity to context. In *Proc. of Work. on Advanced Context Modelling, Reasoning and Management*, 2004.
5. N. Leone, G. Pfeifer, W. Faber, T. Eiter, G. Gottlob, S. Perri, and F. Scarcello. The dlv system for knowledge representation and reasoning. *ACM Trans. Comput. Logic*, 7(3):499–562, 2006.
6. G. Orsi and L. Tanca. Context modelling and context-aware querying: can datalog be of help? In *Datalog 2.0 Workshop, March 2010, Proceedings*, 2010.
7. D. Preuveneers, J. van den Bergh, D. Wagelaar, A. Georges, P. Rigole, T. Clerckx, E. Berbers, K. Coninx, and K. de Bosschere. Towards an extensible context ontology for ambient intelligence. In *Proc. of the 2nd European Symp. on Ambient Intelligence*, pages 148–159, 2004.
8. Y. Roussos and T. Sellis. A model for context aware relational databases. Technical Report TR-2008-6, National Technical University of Athens, 2008.
9. Y. Roussos, Y. Stavrakas, and V. Pavlaki. Towards a context-aware relational model. In *Proc. of 1st intl Context Representation and Reasoning Work.*, pages 7.1–7.12, 2005.
10. A. Segev and A. Gal. Putting things in context: a topological approach to mapping contexts to ontologies. *Journal on data semantics IX*, pages 113–140, 2007.

On the Power of Enforcing Local Consistency*

Gianluigi Greco[1] and Francesco Scarcello[2]

Dept. of Mathematics[1] and DEIS[2], University of Calabria, 87036, Rende, Italy
ggreco@mat.unical.it, scarcello@deis.unical.it

Abstract. Enforcing local consistency is a well-known technique to simplify the evaluation of conjunctive queries. A fundamental result in database theory states that the class of queries for which—on every database—local consistency entails global consistency is precisely the class of acyclic queries. In the last few years, several efforts have been made to define structural decomposition methods isolating larger classes of nearly-acyclic queries, yet retaining the same nice properties as acyclic ones. In particular, it is known that queries having (generalized) hypertree-width at most k can be evaluated in polynomial time, and that this structural property is also sufficient to guarantee that k-local consistency solves the problem, as for acyclic queries. However, the precise power of such an approach was an open problem: Is it the case that bounded generalized hypertree-width is also a necessary condition to guarantee that (k)local consistency entails global consistency? The paper provides a positive answer to the question. In fact, it precisely characterizes the power of enforcing local consistency in the more general framework of tree projections, of which any known structural decomposition method is just a special instance.

1 Properties of Acyclic Conjunctive Queries

Acyclic conjunctive queries, i.e. queries with an acyclic query hypergraph, were the subject of many seminal research works since the early ages of database theory. Indeed, these queries enjoy three properties that are crucial for the design of efficient query optimizers.

Firstly, *acyclic instances can be efficiently solved.* From an acyclic query, we can build (in linear time) a *join tree* [3], which is a tree whose vertices correspond to the various atoms and where the subgraph induced over all the atoms containing any given variable is a tree. According to Yannakakis's algorithm [17], Boolean acyclic queries can be evaluated by processing any of their join trees bottom-up, by performing upward semijoins, thus keeping the size of the intermediate relations small. At the end, if the relation associated with the root atom of the join tree is not empty, then the answer of the query is not empty. For non-Boolean queries, after the bottom-up step described above, one can perform the reverse top-down step by filtering each child vertex from those tuples that do not match with its parent tuples. The filtered database, called *full reducer*, then

* A longer version appeared in the proceedings of ACM PODS 2010 [9].

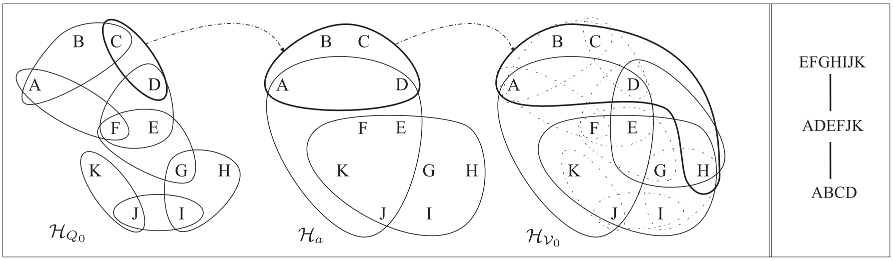


Fig. 1. A tree projection $\mathcal{H}_a$ of $\mathcal{H}_{Q_0}$ w.r.t. $\mathcal{H}_{\mathcal{V}_0}$; On the right: A join tree $\mathcal{JT}_a$ for $\mathcal{H}_a$.

enjoys the *global consistency* property, i.e., no tuple can be missed if we compute the join of all (relations associated with) query atoms (with respect to the new database). Moreover, by exploiting this property, all solutions can be computed with a backtrack-free procedure (i.e., with backtracks occurring only looking for further solutions, never for wrong choices).

Secondly, *acyclicity is efficiently recognizable.* Indeed, deciding whether a hypergraph is acyclic is feasible in linear time [16] and belongs to the class L (deterministic logspace). In fact, this follows from the fact that hypergraph acyclicity belongs to SL [6], and that SL is equal to L [15].

And, finally, *the class of acyclic instances coincides with the class of queries where local consistency entails global consistency.* We say that local (also, pairwise) consistency holds if, for every pair of query atoms p, q, $p \ltimes q = p \neq \emptyset$ and $q \ltimes p = q \neq \emptyset$ hold. The acyclic instances that fulfil this property also fulfil the global consistency property [2]. Thus, by enforcing local consistency, i.e., by taking the semijoins between all pairs of atoms until a fixpoint is reached, we may immediately answer the query (either the database becomes empty and hence the query has no answer, or the resulting database is a full reducer). In addition, and more surprisingly, if a class of instances can be answered by means of this approach, then it only contains acyclic instances [2].

Generalizations of Acyclicity. Structures arising from real applications, however, are hardly precisely acyclic. Yet, they are often not intricate and, in fact, tend to exhibit some limited degree of cyclicity, which suffices to retain most of the nice properties of acyclic ones.

Several efforts have been recently spent to investigate invariants that are best suited to identify nearly-acyclic hypergraphs, leading to the definition of a number of so-called *structural decomposition methods* (such as the *(generalized) hypertree* [7], the *fractional hypertree* [12], and the *component hypertree* [8] decompositions). These methods aim at transforming a given cyclic hypergraph into an acyclic one, by organizing its edges (or its nodes) into a polynomial number of clusters, and by suitably arranging these clusters as a tree, called decomposition tree. The original problem instance can then be evaluated over the decomposition tree, with a cost that is exponential in the cardinality of the largest cluster, also called *width* of the decomposition, and polynomial if this width is bounded by some constant. Despite their different technical definitions, there is a simple mathematical framework that encompasses all the above decomposition methods, which is the framework of the *tree projections* [10]. In this setting, a query Q is given together with a set $\mathcal{V}$ of views, whose schemas are defined over the variables in Q. The question is whether (parts of) the views can be arranged as to form a tree projection, i.e., a novel acyclic query that still "covers" Q. By representing Q and $\mathcal{V}$ via the hypergraphs $\mathcal{H}_Q$ and $\mathcal{H}_\mathcal{V}$, where hyperedges one-to-one correspond with the sets of variables in the query and view atoms, respectively, the tree projection problem reveals its graph-theoretic nature: A tree projection of $\mathcal{H}_Q$ w.r.t. $\mathcal{H}_\mathcal{V}$ is an acyclic hypergraph $\mathcal{H}_a$ such that each hyperedge of $\mathcal{H}_Q$ is contained in some hyperedge of $\mathcal{H}_a$, which is in its turn contained in a hyperedge of $\mathcal{H}_\mathcal{V}$. Whenever such a hypergraph $\mathcal{H}_a$ exists, we say that the pair of hypergraphs $(\mathcal{H}_Q, \mathcal{H}_\mathcal{V})$ has a tree projection.

As an example, it is well-known (see, e.g., [11]) that a query Q has generalized hypertree width bounded by k if, and only if, there is a tree projection of $\mathcal{H}_Q$ w.r.t. $\mathcal{H}_Q^k$, the latter being the hypergraph associated with the set of all the views that can be built by joining all the possible sets of k atoms in Q. In fact, the various decomposition methods just differ in how they define the set of views to be built for evaluating Q. In the tree projection framework, views are instead arbitrary and might be even materialized beforehand.

Example 1. Consider the Boolean conjunctive query

$$\begin{aligned} Q_0 : r_1(A,B,C) \wedge r_2(A,F) \wedge r_3(C,D) \wedge r_4(D,E,F) \wedge \\ r_5(E,F,G) \wedge r_6(G,H,I) \wedge r_7(I,J) \wedge r_8(J,K), \end{aligned}$$

whose associated hypergraph $\mathcal{H}_{Q_0}$ is depicted in Figure 1, together with the other structures discussed next.

To answer Q_0, assume that a set $\mathcal{V}_0$ of views is available comprising the views associated with the query atoms, plus four additional views. The set of attributes of each view (variables in $\{A, ..., K\}$) is a hyperedge in the hypergraph $\mathcal{H}_{\mathcal{V}_0}$ (query views are depicted as dashed hyperedges). In the middle between $\mathcal{H}_{Q_0}$ and $\mathcal{H}_{\mathcal{V}_0}$, Figure 1 reports the hypergraph $\mathcal{H}_a$ which covers $\mathcal{H}_{Q_0}$, and which is in its turn covered by $\mathcal{H}_{\mathcal{V}_0}$—e.g., $\{C, D\} \subseteq \{A, B, C, D\} \subseteq \{A, B, C, D, H\}$.

Since $\mathcal{H}_a$ is in addition acyclic (just check the join tree JT_a), $\mathcal{H}_a$ is a tree projection of $\mathcal{H}_{Q_0}$ w.r.t. $\mathcal{H}_{\mathcal{V}_0}$. $\triangleleft$

2 Tree Projection Properties and Open Questions

The interest on the tree projection framework goes back to the eighties, when it was noticed that queries that admit a tree projection can be evaluated in polynomial time [10] (see, also, [14]). Thus, tree projections smoothly preserve the first crucial property of acyclic queries. On the other hand, tree projections are not efficiently recognizable [8], i.e., the second property is not preserved.

Our knowledge on the third property is less clear instead. Indeed, the question about the precise relationship between tree projections and local and global consistencies was firstly raised in [2], and remained open so far, despite it was attacked via different approaches and proof techniques, which gave some partial results, reported below.

Let $\mathcal{V}$ be an arbitrary set of views that also contains views associated with query atoms. Let $\mathsf{lc}(\mathcal{V}^{\text{DB}})$ denote that the views (evaluated over DB) enjoy the local consistency property, let $\mathsf{gc}(Q^{\text{DB}})$ denote that the query (evaluated over DB) enjoys the global consistency property, and let $Q^{\text{DB}} \neq \emptyset$ denote that the answer of Q on DB is not empty. The picture in the literature is as follows:

- The existence of a tree projection of $\mathcal{H}_Q$ w.r.t. $\mathcal{H}_\mathcal{V}$ entails that, $\forall$DB, $\mathsf{lc}(\mathcal{V}^{\text{DB}}) \rightarrow \mathsf{gc}(Q^{\text{DB}})$ [14].
- For the case of tree projections corresponding to generalized hypertree decompositions, the existence of a tree projection of $\mathcal{H}_{Q'}$ w.r.t. $\mathcal{H}_\mathcal{V}$, where Q' is *the core* of Q and $\mathcal{H}_\mathcal{V} = \mathcal{H}_Q^k$, entails that, $\forall$DB, $\mathsf{lc}(\mathcal{V}^{\text{DB}}) \rightarrow Q^{\text{DB}} \neq \emptyset$ [4]. The result holds for any query Q' that is homomorphically equivalent to Q, denoted by $Q' \approx_{\texttt{hom}} Q$ (instead of just for its core, which is any smallest one).
- For queries with bounded arity, and considering tree projections that correspond to *tree decompositions* [13] (the most general method for structures having bounded arity), a tree projection of $\mathcal{H}_{Q'}$ w.r.t. $\mathcal{H}_\mathcal{V}$, where Q' is the core of Q and $\mathcal{H}_\mathcal{V} = \mathcal{H}_Q^{tk}$ is the hypergraph whose hyperedges are all possible sets of at most $k+1$ variables, exists if, only if, $\forall$DB, $\mathsf{lc}(\mathcal{V}^{\text{DB}}) \rightarrow Q^{\text{DB}} \neq \emptyset$ [1].

Note that the first result states that the existence of a tree projection of the query is a sufficient condition for the global consistency property to hold, whenever the database is local consistent. It was conjectured that the existence of a tree projection is also a necessary condition [10, 14] for having this property. In fact, the second result states a more liberal sufficient condition for the decision problem [4] (for the special case of generalized hypertree decompositions), but we shall see that it is not sufficient for actually computing query answers. The third result provides instead a necessary and sufficient condition for query answering via local consistency, based on tree decompositions [1]. Yet, this condition holds only for structures of fixed arity and, again, only for the decision problem.

From this picture, it emerges that the original question about global consistency, which is related to the computation problem, did not obtain even partial answers. In fact, from the above recent results one may suspect that the mentioned conjecture may not hold, in general. This is because in the general setting, where we may have queries with multiple occurrences of the same relation symbol, the concept of the *core* of a query Q plays a crucial role [5], as it should be clear from the next example.

Example 2. Consider the queries: $Q_1: \ r(A,B) \wedge r(B,C) \wedge r(C,D) \wedge r(D,A)$ and $Q_2: \ r(A,B) \wedge r(B,C) \wedge r(D,C) \wedge r(A,D)$. These queries are completely equivalent as far as their hypergraphs are concerned, since $\mathcal{H}_{Q_1} = \mathcal{H}_{Q_2}$. However, Q_1 is already a core, while a core of Q_2 is the acyclic sub-query $r(A,B) \wedge r(B,C)$. By focusing on Q_2 rather than on its core, we could overestimate its intricacy. $\triangleleft$

However, the above conjecture (about the necessity of the existence of tree projections for having global consistency entailed by local consistency) might still hold in the original setting considered in [10], where all relational symbols in a query are distinct.

3 Results on Tree Projections

In this paper, we provide a clear picture of the relationship between tree projections and local and global consistencies. Concerning the decision problem of checking whether the query has a solution, we show that the sufficient conditions identified for some special cases are also necessary, even in the most general framework. However, the technical machinery needed for obtaining our results is quite different from the one used in [1] for tree decompositions, which does not work when we have arbitrary signatures or arbitrary views. The main ingredients of our proofs are a nice connection between a suitable chase and possible tree projections, and the finite controllability of union of conjunctive queries.

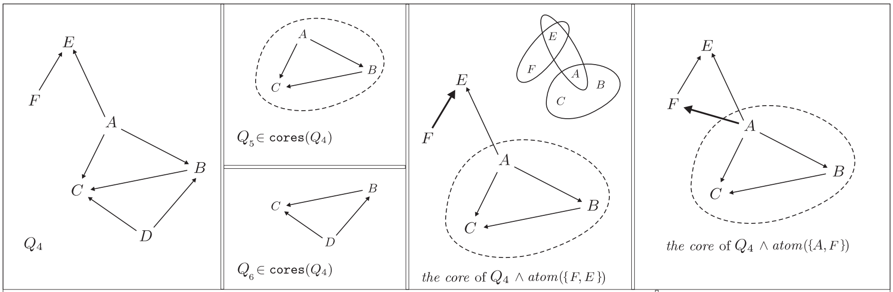
The following are equivalent:
(1) For every database DB*,* $\mathsf{lc}(\mathcal{V}^{\mathrm{DB}})$ *entails* $Q^{\mathrm{DB}} \neq \emptyset$*.*
(2) There is a subquery $Q' \approx_{\mathtt{hom}} Q$ *for which* $(\mathcal{H}_{Q'}, \mathcal{H}_{\mathcal{V}})$ *has a tree projection.*

Unfortunately, we prove that unless P = NP there is no polynomial-time algorithm that may compute a solution, even if some homomorphically equivalent subquery has a tree projection (and thus the decision problem is actually solved by local consistency). In particular, this negative result holds even for the special case of tree decompositions on structures with fixed arities.

Our second contribution is to single out the (stronger) conditions under which local consistency entails global consistency. Firstly, we prove that the conjecture of [10] was correct, if there are no multiple occurrences of the same relation symbol: the existence of a tree projection of the whole query is also a necessary condition. Moreover, we show that in general things are much more intricate, and to find a necessary and sufficient condition requires to carefully exploit possible endomorphisms of the query. It emerged that to characterize when, at local consistency, an atom q contains *all, and only,* the correct tuples of the query Q projected over the variables $var(q) = \{X_1, ..., X_n\}$ of q, we must look for tree projections of some "output-aware" substructures of Q.

We say that $\{X_1, ..., X_n\}$ is *tp-covered* in Q if there is a tree projection of $(\mathcal{H}_{Q'}, \mathcal{H}_{\mathcal{V}})$, where Q' is some core of the novel query $Q \wedge r(X_1, ..., X_n)$, in which r is a fresh relation symbol. Note that we cannot talk about "the" core of this query, since different isomorphic cores may behave differently with respect to the available views, in the setting of tree projections. In fact, Q' is now some minimal subquery forced to contain the desired output variables.

Example 3. Consider the query $Q_4 : r(A,B) \wedge r(B,C) \wedge r(A,C) \wedge r(D,C) \wedge r(D,B) \wedge r(A,E) \wedge r(F,E)$, which is graphically reported in Figure 2, where edge orientation just reflects the position of the variables in the various (binary) query atoms. Consider the set of views $\mathcal{V}_4 = \mathit{views}(Q_4) \cup \{v_1(A,B,C), v_2(A,F)\}$, i.e., assume that there are two additional views covering the variables $\{A,B,C\}$ and $\{A,F\}$, respectively. Then, note that $Q_5 : r(A,B) \wedge r(B,C) \wedge r(A,C)$ and $Q_6 : r(D,B) \wedge r(B,C) \wedge r(D,C)$ are two (isomorphic) cores of Q_4, but they have


E
F
A
B
C
D
Q_4
A
B
C
$Q_5 \in \texttt{cores}(Q_4)$
B
C
D
$Q_6 \in \texttt{cores}(Q_4)$
E
F
A
B
C
E
F
A
B
C
the core of $Q_4 \wedge atom(\{F, E\})$
E
F
A
B
C
the core of $Q_4 \wedge atom(\{A, F\})$

A	B
a_1	b_1
a_2	b_2

B	C
b_1	c_1
b_2	c_2

A	C
a_1	c_1
a_2	c_2

A	E
a_1	c_1
a_2	c_2

F	E
b_1	c_1
b_2	c_2

D	B
d_1	b_1
d_2	b_2
a_1	b_1
a_2	b_2

D	C
d_1	c_2
d_2	c_1
a_1	c_1
a_2	c_2

A	F
a_1	b_1
a_2	b_2
a_1	b_2
a_2	b_1

A	B	C
a_1	b_1	c_1
a_2	b_2	c_2

Fig. 2. The query Q_4, its cores Q_5 and Q_6, the cores of the queries $Q_4 \wedge atom(\{F, E\})$ (with its tree projection), and $Q_4 \wedge atom(\{A, F\})$, and the tuples in the database DB_4.

different structural properties. Indeed, $(\mathcal{H}_{Q_5}, \mathcal{H}_{V_4})$ admits a tree projection (note in the figure that the view over $\{A, B, C\}$ "absorbs" the cycle), while $(\mathcal{H}_{Q_6}, \mathcal{H}_{V_4})$ does not. Consider now the set of variables $\{F, E\}$, which does not occur in any core of the query, and the novel query $Q_4 \wedge atom(\{F, E\})$. This query has a unique core, which is again depicted in Figure 2. Notice that this core does not coincide with any of the two cores of the original query. Nonetheless, it admits a tree projection, consisting of the hyperedges $\{F, E\}$, $\{A, E\}$, and $\{A, B, C\}$, as illustrated in the figure. Thus, $\{F, E\}$ is *tp-covered* in Q_4.

On the other hand, the hypergraphs associated with the cores of both $Q_4 \wedge atom(\{D, C\})$, and $Q_4 \wedge atom(\{D, B\})$ are precisely the same as the hypergraph $\mathcal{H}_{Q_6}$ associated with the core Q_6, that is, the triangle with vertices D, B, C having no tree-projection w.r.t. to $\mathcal{H}_{V_4}$. It follows that both $\{D, C\}$ and $\{D, B\}$ are not *tp-covered* in Q_4. Finally, for an example application with arbitrary set of variables (i.e., not just contained in query atoms), consider the set $\{A, F\}$. Consider then the query $Q_4 \wedge atom(\{A, F\})$ and note that its core does not have a tree projection. Thus, $\{A, F\}$ is not *tp-covered* in Q_4. $\lhd$

The following are equivalent:
(1) For every database DB*,* $\mathsf{lc}(\mathcal{V}^{\mathrm{DB}})$ *entails* $\mathsf{gc}(Q^{\mathrm{DB}})$*.*
(2) For each query atom q, $var(q)$ is tp-covered in Q.

Note that, whenever *(2)* holds and one is interested in output variables included in some query atom, then all solutions are immediately available.

Our third contribution in the general framework of the tree projections is then to deal with those cases where one is interested in computing answers over an arbitrary subset of variables covered by some available view.

The following are equivalent:
(1) For every DB*,* $\mathsf{lc}(\mathcal{V}^{\mathrm{DB}})$ *entails "views containing O are correct".*
(2) The set of variables O is tp-covered in Q.

Example 4. Consider the database DB_4 shown in Figure 2 for the views $v_1(A,B,C)$, $v_2(A,F)$, $v_3(A,B)$, $v_4(B,C)$, $v_5(A,C)$, $v_6(D,B)$, $v_7(D,C)$, $v_8(A,E)$, and $v_9(F,E)$. The database is local consistent. Observe that, e.g., $v_5(A,C)$ contains exactly the tuples that we get by evaluating Q_4 on DB_4 over the output variables $\{A,C\}$. However, $v_6(D,B)$, $v_7(D,C)$, and $v_2(A,F)$ contain two tuples that do not belong to any answer of the query. $\triangleleft$

4 Applications to *k*-local Consistency

The results we have stated above in the general framework of tree projections can be specialized to the case known as k-local consistency, that is, the case where local consistency is enforced among views obtained by considering all possible groups of k variables or k atoms. It is well known and also observed above that such groups of views may be used to characterize tree-decompositions and hypertree decompositions, respectively. However, the precise power of enforcing local consistency in the latter case was an open question, while for the case of groups of k-variables, this question was recently closed, at least for the decision problem [1]. We next provide a full answer to these questions, by focusing for the sake of simplicity on the general (and more open case) of groups of k atoms.

Let $k > 0$ be any natural number, and let Q be a conjunctive query and DB a database. Let $\mathcal{V}_k$ be the set of views that, besides views associated with query atoms, contains all views obtained as the conjunction of any group $\{q_1, \ldots, q_{k'}\}$ of atoms from Q, with $k' \leq k$. Let DB_k be its corresponding database, that is, the database that contains, for any such a view in $\mathcal{V}_k$, the join of atoms $\{q_1, \ldots, q_{k'}\}$ w.r.t. DB (i.e., the subquery of Q^{DB} over these atoms). Also, denote by *tp*$_k$*-covered* the *tp-covered* property where $\mathcal{V}_k$ is used in place of $\mathcal{V}$. Then, we next provide the precise characterization of the power of k-local consistency:

The following are equivalent:
(1) For every database DB, $\mathsf{lc}(\mathcal{V}_k^{\mathrm{DB}_k})$ *entails "views containing O are correct".*
(2) The set of variables O is tp$_k$*-covered in Q.*

Note that the result is not a trivial consequence of the previous one. Indeed, the database DB over which we enforce local consistency is not arbitrary anymore, because the relation associated with each view is computed by means of a join of at most k relations from DB. Hence, some further effort is needed to show that (1) entails (2). In fact, for any set O that is not *tp-covered* in Q, we are able to show a counterexample database DB such that its derived database DB_k has the desired property: some view containing the variables O is not correct after enforcing local consistency on $\mathcal{V}_k$. For completeness, note that this property, at local consistency, entails that every view containing O is not correct.

Finally, we provide an even stronger and more readable result precisely characterizing the power of k-local consistency in terms of generalized hypertree decompositions. For simplicity, we next report its specialization to the decision problem ($O = \emptyset$), which is also the affirmative answer to the open question in [4].

The following are equivalent:
(1) For every database DB, $\mathsf{lc}(\mathcal{V}_k^{\mathrm{DB}_k})$ *entails* $Q^{\mathrm{DB}} \neq \emptyset$*.*
(2) Q *has a core having generalized hypertree width at most* k*.*

Even in this case, observe that the result is an easy but not trivial corollary of the previous one, because of a subtle issue. Let Q' be any core of Q, and recall that Q' may be much smaller than Q. Thus, the set of views that can be used to form a k-width hypertree decomposition of Q' only come from groups of k atoms occurring in Q'. It follows that this set may be much smaller than the full set $\mathcal{V}_k$ which is built from the full query Q, and which is considered in the *tp*$_k$*-covered* notion (and hence in the previous result).

References

1. A. Atserias, A. Bulatov, and V. Dalmau. On the Power of k-Consistency, In *Proc. of ICALP'07*, pp. 279–290, 2007.
2. C. Beeri, R. Fagin, D. Maier, and M. Yannakakis. On the Desirability of Acyclic Database Schemes. *Journal of the ACM*, 30(3), pp. 479–513, 1983.
3. P.A. Bernstein and N. Goodman. The power of natural semijoins. *SIAM Journal on Computing*, 10(4), pp. 751–771, 1981.
4. H. Chen and V. Dalmau. Beyond Hypertree Width: Decomposition Methods Without Decompositions. In *Proc. of CP'05*, pp. 167–181, 2005.
5. V. Dalmau, Ph.G. Kolaitis, and M.Y. Vardi. Constraint Satisfaction, Bounded Treewidth, and Finite-Variable Logics. In *Proc. of CP'02*, pp. 310–326, 2002.
6. G. Gottlob, N. Leone, and F. Scarcello. The complexity of acyclic conjunctive queries. *Journal of the ACM*, 48(3), pp. 431–498, 2001.
7. G. Gottlob, N. Leone, and F. Scarcello. Hypertree decompositions and tractable queries. *Journal of Computer and System Sciences*, 64(3), pp. 579–627, 2002.
8. G. Gottlob, Z. Miklós, and T. Schwentick. Generalized hypertree decompositions: NP-hardness and tractable variants. *Journal of the ACM*, 56(6), 2009.
9. G. Greco and F. Scarcello. The power of tree projections: local consistency, greedy algorithms, and larger islands of tractability. In *Proc. of PODS'10*, pp. 327–338.
10. N. Goodman and O. Shmueli. The tree projection theorem and relational query processing. *Journal of Computer and System Sciences*, 29(3), pp. 767–786, 1984.
11. G. Greco and F. Scarcello. Tree Projections: Hypergraph Games and Minimality. In *Proc. of ICALP'08*, pp. 736–747, 2008.
12. M. Grohe and D. Marx. Constraint solving via fractional edge covers. In *Proc. of SODA '06*, pp. 289–298, 2006.
13. N. Robertson and P.D. Seymour. Graph minors III: Planar tree-width. *Journal of Combinatorial Theory, Series B*, 36, pp. 49–64, 1984.
14. Y. Sagiv and O Shmueli. Solving Queries by Tree Projections. *ACM Transaction on Database Systems*, 18(3), pp. 487–511, 1993.
15. O. Reingold. Undirected ST-connectivity in log-space, *Journal of the ACM*, 55(4), 2008.
16. R.E. Tarjan, and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566-579, 1984.
17. M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. of VLDB'81*, pp. 82–94, 1981.

A Constructive Framework for View Updating
(Extended Abstract)

Enrico Franconi and Paolo Guagliardo

franconi@inf.unibz.it — paolo.guagliardo@stud-inf.unibz.it

KRDB Research Centre, Free University of Bozen-Bolzano, Italy

1 Introduction

Updating a database by means of a view requires the changes made on the view to be properly propagated to the underlying database. This task of "translating" a view update into a suitable database update is non-trivial and it poses several challenges indeed. The most subtle kind of update anomaly is given by changes not directly wanted nor explicitly made by the user, originating in the view as a "side-effect". An additional difficulty is represented by the fact that there can be more database updates corresponding to a given update on the view. Lastly, yet another complication concerns updates that modify the database even though this is not strictly necessary in order to reflect the changes.

A general and precise understanding of the view update problem is due to the seminal work [1] by Bancilhon and Spyratos, who devise an abstract framework in which they formalise the problem and provide an elegant solution to it. They introduce the fundamental notion of *view complement*, representing what is missing from a view in order to have the same informative content of the underlying database. Moreover, they introduce the *constant complement* principle, stating that the changes done on a view must not influence the content of its complement during the translation process. Bancilhon and Spyratos provide no constructive characterisation of their approach, stating that "computational algorithms (if they exist) must be sought in specific problems: for example, schemata defined by functional dependencies and views derived by projections". Most of the subsequent work on view updates is centred on the framework [1] and, in particular, its application to the relational model.

Cosmadakis and Papadimitriou [4] consider a restricted setting that consists of a single database relation, in which views are just projections over the attributes of such a "universal" relation. They give necessary and sufficient conditions for the translatability of insertions, deletions and replacements under constant complement and they also study the complexity of finding a suitable complement that makes a given update translatable. Even though an effective method based on the Chase is provided for checking translatability, it is only applicable in the very limited setting consisting of a single database relation with projective views and functional and join dependencies at the database level.

In the context of SQL databases, Lechtenbörger [6] gives a characterisation of the constant complement principle in terms of undo operations, showing that view updates are translatable under constant complement precisely if users have the chance to undo all effects of their updates using further view updates. It is then argued that testing whether this holds could be an alternative to checking whether users can observe all effects of their updates in the view schema, but no effective method for doing so is proposed.

Gottlob et al. [5] extend the results of Bancilhon and Spyratos to the class of so-called *consistent views*, which properly contains the class of views translating under constant complement. The main difference between the two is that, during the translation of an update on a consistent view, the complement is not required to remain invariant, but it is allowed to "decrease" according to a suitable partial order. Indeed, in the case in which the partial order is the equality, the framework coincides with the one in [1]. Also in this generalised framework, no constructive characterisation and effective methods for checking the translatability of updates and computing their translations are provided.

In [1], Bancilhon and Spyratos deal with constraints in an implicit way and only at the database level. Here, we revisit their framework by considering explicit constraints on both the database and the view schemas, along with inter-schema constraints that implicitly define the view mappings. All the constraints are expressed as a theory in first-order logic (FOL). We introduce the new notion of *view under constraints*, we give a constructive characterisation, based on logical definability, of such views, their inverses and complements, and we provide an effective method for checking whether a FOL-expressible view update can be translated under the constant complement principle.

2 Preliminaries

An *n-ary relation* on a set A, where $n \in \mathbb{N}$ is called the *arity* of the relation, is a set of n-tuples of elements of A. A *signature* is a finite set of relation symbols, each of which has an associated arity. A *relational structure* $\mathcal{I}$ over a signature σ is a pair $\langle \Delta^{\mathcal{I}}, \cdot^{\mathcal{I}} \rangle$, where $\Delta^{\mathcal{I}}$ is a (possibly infinite) domain of objects and $\cdot^{\mathcal{I}}$ is an *interpretation function* that associates each symbol $r \in \sigma$ with a relation $r^{\mathcal{I}}$ on $\Delta^{\mathcal{I}}$ of appropriate arity (that is, if r is an n-placed relation symbol, then $r^{\mathcal{I}}$ is an n-ary relation). For relational structures $\mathcal{I} = \langle \Delta, \cdot^{\mathcal{I}} \rangle$ and $\mathcal{J} = \langle \Delta, \cdot^{\mathcal{J}} \rangle$ over two disjoint signatures σ and σ', respectively, $\mathcal{I} \uplus \mathcal{J} = \langle \Delta, \cdot^{\mathcal{I}} \cup \cdot^{\mathcal{J}} \rangle$ is a relational structure over $\sigma \cup \sigma'$.

A *constraint* is a closed first-order logic formula φ. We denote by $\mathsf{sig}(\varphi)$ the set of relation symbols occurring in φ and we say that φ is over a signature σ if $\mathsf{sig}(\varphi) \subseteq \sigma$. We extend $\mathsf{sig}(\cdot)$ to sets of constraints in the natural way. Given two disjoint signatures σ and σ' and a finite set Σ of constraints over $\sigma \cup \sigma'$, we say that a relation symbol $r \in \sigma$ is *implicitly defined* by the relation symbols in σ' under Σ if and only if, for every two Σ-models $\mathcal{I}$ and $\mathcal{J}$ such that $\Delta^{\mathcal{I}} = \Delta^{\mathcal{J}}$, we have that $r^{\mathcal{I}} = r^{\mathcal{J}}$ whenever $p^{\mathcal{I}} = p^{\mathcal{J}}$ for every $p \in \sigma'$. We say that $r \in \sigma$ is *explicitly defined* by the relation symbols in σ' under Σ if and only if there exists a FOL formula $\phi_r(\overline{x})$ over σ', with as many free variables as the arity of r, such that $\Sigma \models \forall \overline{x} \big(r(\overline{x}) \equiv \phi_r(\overline{x}) \big)$. The formula $\phi_r(\overline{x})$ is called an *explicit definition* (or simply *definition*) of r with respect to σ' under Σ.

Clearly, if a relation symbol $r \in \sigma$ has an explicit definition w.r.t. σ' under constraints Σ, then r is also implicitly defined by σ' under Σ, because fixing the interpretation of the relation symbols in σ' determines the interpretation of r. In other words, explicit definability always implies implicit definability. In general, the converse does not hold, that is, knowing that a certain symbol is implicitly defined by some other ones does not mean that we can find an explicit definition of it in terms of those symbols. A fundamental result due to Beth [2] establishes that this is actually the case for FOL.

Theorem (Beth's Definability). *Let σ and σ' be disjoint signatures, and let Σ be a finite set of constraints over $\sigma \cup \sigma'$. If $r \in \sigma$ is implicitly definable from σ' under Σ, then r has an explicit definition with respect to σ' under Σ.*

Note, however, that the above does not hold for all fragments of first-order logic, because even though a FOL explicit definition is always guaranteed to exist, this might not belong to the particular fragment under consideration. In this work, we consider constraints expressed in FOL, for which it is possible to effectively check for implicit definability and for which explicit definitions can be constructively obtained by means of rewriting techniques based on interpolation, e.g., as described in [3]. So, by reducing our problem to testing for implicit definability and computing explicit definitions, we have found a constructive solution to it.

A *database schema* is a signature $\mathcal{R} = \{R_1, \ldots, R_n\}$ of *database symbols* and a *view schema* is a signature $\mathcal{V} = \{V_1, \ldots, V_k\}$ of *view symbols* not occurring in $\mathcal{R}$. A *database state* is a relational structure over $\mathcal{R}$ and a *view state* is a relational structure over $\mathcal{V}$. We denote the sets of all database and view states by S and T, respectively. For a database state $s \in S$ and a view state $t \in T$ having the same domain, the relational structure $s \uplus t$ is called a *global state* over $\mathcal{R} \cup \mathcal{V}$. We consider a satisfiable finite set Σ of *global constraints* over the signature $\mathcal{R} \cup \mathcal{V}$. As $\mathcal{R}$ and $\mathcal{V}$ are disjoint, Σ is partitioned into subsets $\Sigma_{\mathcal{R}}$, $\Sigma_{\mathcal{V}}$ and $\Sigma_{\mathcal{R}\mathcal{V}}$, which we call the *database constraints*, the *view constraints* and the *inter-schema constraints*, respectively. A database state s (resp., view state t) is *Σ-consistent* (or *globally consistent*, or *consistent with the global constraints*) iff there exists a view state t (resp., database state s) with the same domain such that $s \uplus t$ is a model of Σ. We denote the set of Σ-consistent database states (resp., view states) by S_Σ (resp., T_Σ). A *renaming* over a signature σ is a bijective function $\mathsf{ren}\colon \sigma \to \sigma'$, where σ' is disjoint from σ. We extend $\mathsf{ren}(\cdot)$ to signatures, relational structures and (sets of) constraints in the natural way.

The central notion we rely on is that of *view under constraints*, which naturally extends with explicit constraints the definition of view used in [1]. Indeed, when only database constraints are present the two notions essentially coincide, although in [1] constraints (at the database level) are considered only implicitly.

Definition 1 (View under constraints). A *view* from $\mathcal{R}$ to $\mathcal{V}$ under Σ is a total mapping $f\colon S_\Sigma \to T_\Sigma$ such that $s \uplus f(s) \models \Sigma$ for every $s \in S_\Sigma$. ■

We write $\mathcal{R} \twoheadrightarrow_\Sigma \mathcal{V}$ to indicate that every $V \in \mathcal{V}$ is implicitly definable from $\mathcal{R}$ under constraints Σ. In addition, we also use $\mathcal{V} \twoheadrightarrow_\Sigma \mathcal{R}$, $\mathcal{R} \not\twoheadrightarrow_\Sigma \mathcal{V}$ and $\mathcal{V} \not\twoheadrightarrow_\Sigma \mathcal{R}$ with the obvious meaning. Since $\mathcal{R}$ and $\mathcal{V}$ are disjoint, every model of Σ has the form $s \uplus t$, where $s \in S$ and $t \in T$ are (globally consistent) states with the same domain. Therefore, we can characterise definability in terms of states as follows.

Definition 2. We say that $\mathcal{R}$ *defines* $\mathcal{V}$ under Σ (written $\mathcal{R} \twoheadrightarrow_\Sigma \mathcal{V}$) if and only if, for every $s \in S$ and $t, t' \in T$, whenever $s \uplus t \models \Sigma$ and $s \uplus t' \models \Sigma$, it is the case that $t = t'$. ■

In general, there might exist more than one view satisfying a given set of constraints. An important connection between views under constraints and definability is that one and only one mapping is possible exactly when each view symbol is defined by the database symbols under the given constraints.

Theorem 1. *$\mathcal{R} \twoheadrightarrow_\Sigma \mathcal{V}$ iff there is one and only one view from $\mathcal{R}$ to $\mathcal{V}$ under Σ.*

The above theorem gives a characterisation of the views that are expressible by means of constraints in FOL. In what follows, we write $\mathcal{R} \twoheadrightarrow^{f}_{\Sigma} \mathcal{V}$ to indicate that $\mathcal{R} \twoheadrightarrow_{\Sigma} \mathcal{V}$ and f is the (one and only) view *induced by the constraints* Σ from $\mathcal{R}$ to $\mathcal{V}$. Every function f can be made surjective by restricting its codomain to its image: we call the resulting function the *surjection induced by* f or the *surjective restriction of* f. We use concatenation to indicate composition, e.g., fg denotes the composition of f with g.

3 The View Update Problem

In this section, we formally state the problem of view update, by reviewing and adapting some of the "classical" definitions given in [1] to our logic-based setting with constraints.

A *database update* is a function $d\colon S \to S$ that associates each database state with another, possibly the same. Similarly, a *view update* is a function $u\colon T \to T$ associating each view state with another, possibly the same. We denote by $U_{\mathcal{R}}$ and $U_{\mathcal{V}}$ the sets of all database and view updates, respectively. An update $u \in U_{\mathcal{V}}$ is called *strict on* f iff there exists $t \in f(S_{\Sigma})$ such that $u(t) \neq t$. In other words, a strict update does not coincide with the identity mapping on the image of f. The set of all the updates that are strict on f is denoted by U_f.

Given a view under constraints and a view update, we want to find a suitable database update that propagates the changes to the base relations in a consistent way. More specifically, the view update should be translated as a database update that brings the database into a new state from which, by applying the view definition, we reach exactly the updated view state. In addition, we also want to avoid unjustified and unnecessary changes in the database, in the sense that if the view update does not modify the view state, then the database update must not modify the corresponding database state. These requirements are formalised below (cf. Definition 3.1 in [1]).

Definition 3 (Translation). Let f be a view from $\mathcal{R}$ to $\mathcal{V}$ under Σ, let $d \in U_{\mathcal{R}}$ and $u \in U_{\mathcal{V}}$. We say that d is a *translation* of u (w.r.t. f) iff (1) $uf = fd$ and (2) $\forall s \in S_{\Sigma},\ uf(s) = f(s) \implies d(s) = s$. ■

A translation of a given update on a view f can only exist if the updated view state lies in the image of f; otherwise, there would be no chance of reaching the new view state through f from some database state, which is what Definition 3 indeed requires. Hence, before we start looking for a translation, we should first make sure that the given view update allows for one.

Definition 4 (Translatability). Let $f\colon S_{\Sigma} \to T_{\Sigma}$ be a view from $\mathcal{R}$ to $\mathcal{V}$ under Σ. A view update $u \in U_{\mathcal{V}}$ is *translatable* (w.r.t. f) if and only if for each $s \in S_{\Sigma}$ there exists $s' \in S_{\Sigma}$ such that $f(s') = uf(s)$. ■

Note that the condition of translatability given in Definition 4 is equivalent to saying that u is translatable iff $u\big(f(S_{\Sigma})\big) \subseteq f(S_{\Sigma})$.

Translatability of view updates ensures that there is a translation, but does not rule out the possibility that more than one might exist, which is problematic because we would not know how to choose one. Therefore, we are only interested in view updates that are *uniquely translatable*, that is, for which there exists one and only one translation.

When the view mapping is injective, a view update is translatable if and only if it is uniquely translatable and the following theorem gives a characterisation of its unique translation.

THEOREM 2. *Let f be an injective view under Σ, let $u \in U_{\mathcal{V}}$ be translatable and let $d \in U_{\mathcal{R}}$. Let $\hat{f}$ denote the surjection induced by f and let $\hat{u}$ be obtained from u by restricting its domain and codomain to $f(S_\Sigma)$.*[1] *Then, d is a translation of u if and only if $d = \hat{f}^{-1}\hat{u}\hat{f}$.*

In order to be able to apply the result of Theorem 2, we need to know, in the first place, whether a view is injective. More importantly, once in the presence of an injective view, we also need some way of computing the inverse of its surjective restriction, so as to effectually obtain the unique translation of any translatable view update. Therefore, below we study the injectivity and surjectivity of views under constraints using logical definability, with the aim of giving a constructive characterisation of their inverse.

LEMMA 1. *Let f be a view from $\mathcal{R}$ to $\mathcal{V}$ under Σ and let $\mathcal{V} \twoheadrightarrow^h_\Sigma \mathcal{R}$. Then, 1) f is injective; 2) the restriction of h to the image of f is the inverse of the surjection induced by f.*

Note that in the above lemma, f can be any view under constraints. In the case of a view induced by the constraints, we have the following:

LEMMA 2. *Let $\mathcal{R} \twoheadrightarrow_\Sigma \mathcal{V}$ and let f be the view from $\mathcal{R}$ to $\mathcal{V}$ induced by Σ. Then, 1) f is surjective; 2) f is injective if and only if $\mathcal{V} \twoheadrightarrow_\Sigma \mathcal{R}$.*

Assuming the view to be induced by the constraints is a restriction almost always satisfied in practise. In fact, a view is usually specified by providing an explicit definition for each view symbol, in terms of the database symbols. For instance, each (virtual) view table in SQL is defined by means of a named `SELECT`-query over the database tables.

The following is an important consequence of Lemma 1 and Lemma 2, stating that it is possible to invert a view induced by a set of constraints iff the database symbols are implicitly defined by the view symbols under the same constraints, in which case the inverse is also effectively computable. In such a situation, the constraints induce two views that are indeed one the inverse of the other.

THEOREM 3. *Let $\mathcal{R} \twoheadrightarrow^f_\Sigma \mathcal{V}$. Then, f is invertible if and only if $\mathcal{V} \twoheadrightarrow^h_\Sigma \mathcal{R}$, and in such a case $h = f^{-1}$.*

The next step towards the application of Theorem 2 is the translatability of view updates, of which we give an interesting characterisation in what follows. The general idea consists in imposing additional constraints on the view schema so that every *legal* view update is translatable.

A consistent set of constraints over $\mathcal{R} \cup \mathcal{V}$ is $\mathcal{R}$*-defining* iff it contains only formulas, one for each $R \in \mathcal{R}$, of the form $\forall \overline{x}\big(R(\overline{x}) \equiv \phi_R(\overline{x})\big)$, with $\mathsf{sig}\big(\phi_R(\overline{x})\big) \subseteq \mathcal{V}$. Clearly, an $\mathcal{R}$-defining set Θ is such that $\mathcal{V} \twoheadrightarrow_\Theta \mathcal{R}$ and induces a function θ from $\mathcal{V}$ to $\mathcal{R}$. Since Θ does not contain nor entail any database or view constraints, every view state $t \in T$ is Θ-consistent and therefore in the domain of θ. We know by Beth's theorem that whenever $\mathcal{V} \twoheadrightarrow_\Sigma \mathcal{R}$ there is an explicit definition for each

[1] As u is assumed to be translatable, $u\big(f(S_\Sigma)\big) \subseteq f(S_\Sigma)$.

of the database symbols in terms of the view symbols, that is, the constraints entail an $\mathcal{R}$-defining set Θ. In such a case, we call the *$\mathcal{V}$-embedding* of Σ the set $\widetilde{\Sigma}_{\mathcal{V}}$ of view constraints obtained by replacing, for each $R \in \mathcal{R}$, every occurrence of $R(\overline{x})$ in Σ with the definition $\phi_R(\overline{x})$ given in Θ. The $\mathcal{V}$-embedding of a set Σ of global constraints is a set of view constraints having the same "restrictiveness" of the whole Σ, but with the advantage that they can be checked locally on the view schema. Indeed, it turns out that a view state is Σ-consistent iff it is a model of $\widetilde{\Sigma}_{\mathcal{V}}$ and this is of particular importance for surjective views.

Theorem 4. *Let f be a surjective view from $\mathcal{R}$ to $\mathcal{V}$ under Σ, let $\mathcal{V} \twoheadrightarrow_{\Sigma} \mathcal{R}$ and let $u \in U_{\mathcal{V}}$. Then, u is translatable if and only if $u(t) \models \widetilde{\Sigma}_{\mathcal{V}}$ for every $t \in T_{\Sigma}$.*

Note that, under the assumptions of Theorem 4, every globally consistent view state is in the image of the view and, moreover, satisfies the $\mathcal{V}$-embedding of the global constraints. Thus, the above result essentially says that we have to make sure that, by updating a view state that is legal w.r.t. the embedded constraints, we always end up in another legal view state.

Let ren be a renaming over $\mathcal{R} \cup \mathcal{V}$ and let $\mathsf{ren}(\mathcal{V}) = \mathcal{V}'$. Then, a $\mathcal{V}'$-defining set Ξ of constraints over $\mathcal{V} \cup \mathcal{V}'$ represents a view update. Indeed, the function ξ induced by Ξ takes a view state t over $\mathcal{V}$ and returns an updated view state $\xi(t)$ over $\mathcal{V}'$. The view update *expressed* by Ξ is the function associating each $t \in T$ with $\mathsf{ren}^{-1}\big(\xi(t)\big)$. From Theorem 4, we then get the following characterisation of the translatability of those view updates expressible as a logical theory.

Theorem 5. *Let f be a surjective view from $\mathcal{R}$ to $\mathcal{V}$ under Σ, let $\mathcal{V} \twoheadrightarrow_{\Sigma} \mathcal{R}$ and let $u \in U_{\mathcal{V}}$ be expressed by Ξ. Then, u is translatable iff $\widetilde{\Sigma}_{\mathcal{V}} \cup \Xi \models \mathsf{ren}\big(\widetilde{\Sigma}_{\mathcal{V}}\big)$.*

Under the assumptions of the above theorem, the view f is injective by Lemma 1. Hence, by Theorem 2 every translatable view update u has the unique translation $f^{-1}uf$. However, we might not be able to compute f^{-1} unless $\mathcal{R} \twoheadrightarrow_{\Sigma} \mathcal{V}$, in which case Theorem 3 ensures that the inverse of f is the view from $\mathcal{V}$ to $\mathcal{R}$ induced by Σ. If $\mathcal{R} \twoheadrightarrow_{\Sigma} \mathcal{V}$, $\mathcal{V} \twoheadrightarrow_{\Sigma} \mathcal{R}$ and Ξ expresses a translatable view update u, we have that $\mathcal{V} \twoheadrightarrow_{\Xi} \mathsf{ren}(\mathcal{V})$ and $\mathsf{ren}(\mathcal{V}) \twoheadrightarrow_{\mathsf{ren}(\Sigma)} \mathsf{ren}(\mathcal{R})$, therefore the unique translation of u is the database update expressed by the set Υ such that $\mathcal{R} \twoheadrightarrow_{\Upsilon} \mathsf{ren}(\mathcal{R})$, obtained by replacing in $\mathsf{ren}(\Sigma)$ every occurrence of $\mathsf{ren}(V)$ with its definition in terms of $\mathcal{V}$ and, in turn, every occurrence of V with its definition in terms of $\mathcal{R}$.

4 The View Complement

The lack of injectivity in a view causes a loss of information due to the fact that distinct database states are mapped to the same view state. In order to be able to distinguish between distinct database states, we need some extra "hints" that, combined with what is already known from the view itself, give a full account of the database content. This additional data is provided by another view, called a *view complement*, as it "complements" the partial information of a lossy view.

For the rest of this section, let $\mathcal{R}$, $\mathcal{V}$ and $\mathcal{W}$ be pairwise disjoint signatures, and let Σ and Γ be finite sets of constraints over $\mathcal{R} \cup \mathcal{V}$ and $\mathcal{R} \cup \mathcal{W}$, respectively, such that their union is consistent.

Definition 5 (View complement). Let f be a view from $\mathcal{R}$ to $\mathcal{V}$ under Σ and let g be a view from $\mathcal{R}$ to $\mathcal{W}$ under Γ. We say that g is a *complement* of f iff (1) $S_{\Sigma} = S_{\Gamma}$ and (2) $\forall s, s' \in S_{\Sigma},\ s \neq s' \wedge f(s) = f(s') \implies g(s) \neq g(s')$. ■

In other words, a complement of f is a view g operating on the same domain of f and capable of distinguishing between distinct database states which f maps to the same view state. Note that there exists at least one complement for every view, namely the "identity" mapping over the whole database.

The idea of view complement was first introduced by Bancilhon and Spyratos in [1]. Our definition is indeed based on their work (cf. Theorem 4.2 in [1]) with the additional requirement that f and g must have the same domain, which has to be explicitly enforced here as we are in a setting with views under constraints. Since the notion of view complement is symmetric, in that g is a complement of f iff f is a complement of g, we will sometimes simply say that two views f and g are "complementary".

Given two views f and g under constraints Σ and Γ, respectively, their *union* is the function $f \uplus g$ associating each $s \in S_\Sigma \cap S_\Gamma$ with the state $f(s) \uplus g(s)$. The union of f and g turns out to be a view under $\Sigma \cup \Gamma$ and, when f and g are induced by their associated constraints, $f \uplus g$ is indeed induced by $\Sigma \cup \Gamma$. The connection between complementarity and injectivity of views is given by the fact that two views under constraints and with the same domain are complementary if and only if their union is injective.

Lemma 3. *Let f be a view from $\mathcal{R}$ to $\mathcal{V}$ under Σ, let g be a view from $\mathcal{R}$ to $\mathcal{W}$ under Γ and let $S_\Sigma = S_\Gamma$. Then, $f \uplus g$ is injective iff f and g are complementary.*

Below we give a characterisation of complementarity between two views induced by constraints in terms of logical equivalence and definability.

Theorem 6. *Let $\mathcal{R} \twoheadrightarrow^{f}_{\Sigma} \mathcal{V}$ and $\mathcal{R} \twoheadrightarrow^{g}_{\Gamma} \mathcal{W}$. Then, f and g are complementary if and only if $\widetilde{\Sigma}_\mathcal{R} \equiv \widetilde{\Gamma}_\mathcal{R}$ and $\mathcal{V} \cup \mathcal{W} \twoheadrightarrow_{\Sigma \cup \Gamma} \mathcal{R}$.*[2]

Therefore, one way of finding a complement of a given view f induced by a set of constraints Σ consists in abducing another set of constraints Γ consistent with Σ and satisfying the conditions of Theorem 6, which guarantees that the view g induced by such a Γ is indeed a complement of f.

Following the rationale that the only purpose for which a view complement is made available is that of allowing for a lossy view to be updatable, we demand that the information it provides be invariant during the update process. In other words, view updates must never modify, neither directly nor indirectly, any data that belongs to the view complement. Putting together translatability of updates and invariance of the complement results in the formal notion given below (cf. Definition 5.1 in [1]).

Definition 6 (g-translatability). Let f be a view under Σ and let g be a complement of f. A view update $u \in U_\mathcal{V}$ is called *g-translatable* iff for each $s \in S_\Sigma$ there exists $s' \in S_\Sigma$ such that (1) $f(s') = uf(s)$ and (2) $g(s') = g(s)$. ■

That is, a view update is g-translatable if it is translatable (according to Definition 4) and, in addition, leaves the complement g unchanged. For this reason, we say that such an update is *translatable under constant complement*. In general, there might be more than one complement of a given view, and an update is g-translatable or not depending on the particular complement g we consider. Thus,

[2] $\widetilde{\Sigma}_\mathcal{R}$ and $\widetilde{\Gamma}_\mathcal{R}$ denote the $\mathcal{R}$-embeddings of Σ and Γ, respectively.

the choice of a complement defines an “update policy” by assigning unambiguous semantics to the view updates.

The following theorem establishes an important relationship between translatability w.r.t. a view under constant complement and translatability w.r.t. the union of a view and its complement.

Theorem 7. *Let f and g be complementary, let $u \in U_{\mathcal{V}}$ and let $v \in U_{\mathcal{W}} \setminus U_g$. Then, u is g-translatable w.r.t. f if and only if $u \uplus v$ is translatable w.r.t. $f \uplus g$.*

This allows us to extend the result obtained for the translatability of updates on injective views to the case of g-translatability.

Theorem 8. *Let $\mathcal{R} \twoheadrightarrow^{f}_{\Sigma} \mathcal{V}$, let $\mathcal{R} \twoheadrightarrow^{g}_{\Gamma} \mathcal{W}$ and let g be a complement of f. Let $\widetilde{\Pi}$ be the $(\mathcal{V} \cup \mathcal{W})$-embedding of $\Sigma \cup \Gamma$ and let* ren *be a renaming over $\mathcal{R} \cup \mathcal{V} \cup \mathcal{W}$. Let $u \in U_{\mathcal{V}}$ be expressed by Ξ and let Ω be the $\mathcal{W}$-defining set such that $\forall \overline{x} \,.\, W(\overline{x}) \equiv \mathsf{ren}\big(W(\overline{x})\big)$ for each $W \in \mathcal{W}$. Then, u is g-translatable if and only if $\widetilde{\Pi} \cup \Xi \cup \Omega \models \mathsf{ren}(\widetilde{\Pi})$.*

5 Conclusion

We presented a framework for view update based on the notion of “view under constraints”. On the one hand, such a framework “extends”—so to say—the one of Bancilhon and Spyratos by adding explicit constraints also at the view level. Indeed, when there are no inter-schema constraints nor constraints on the view schema, the notion of view under constraints coincides with the notion of view used in [1]. On the other hand, our framework is an instance of Bancilhon and Spyratos’ abstract one, in that we essentially consider only view mappings that are expressible by means of first-order logic constraints.

Using logical definability, we gave a constructive characterisation of when and whether a view induced by a set of constraints is invertible, so as to being able to effectively compute the (unique) translation of a view update that is translatable and expressible in FOL. Indeed, we also provided an applicable method, based on the Beth’s definability property and the idea of local “embedding” of the constraints, for testing whether a FO-expressible view update is translatable (under constant complement). We have an experimental tool, based on a FOL theorem prover, that checks for implicit definability and derives explicit definitions, and thus it can be used for testing the criterion of translatability we presented here and for computing the corresponding translation.

References

1. F. Bancilhon and N. Spyratos. Update semantics of relational views. *ACM Transactions on Database Systems*, 6(4):557–575, December 1981.
2. E. W. Beth. On Padoa’s method in the theory of definition. *Indagationes Mathematicae*, 15:330–339, 1953.
3. A. Borgida, J. de Bruijn, E. Franconi, I. Seylan, U. Straccia, D. Toman, and G. Weddell. On finding query rewritings under expressive constraints. In *Proc. SEBD*, 2010.
4. S. S. Cosmadakis and C. H. Papadimitriou. Updates of relational views. *Journal of the Association for Computing Machinery*, 31(4):742–760, October 1984.
5. G. Gottlob, P. Paolini, and R. Zicari. Properties and update semantics of consistent views. *ACM Transactions on Database Systems*, 13(4):486–524, December 1988.
6. J. Lechtenbörger. The impact of the constant complement approach towards view updating. In *Proceedings of PODS 2003*, pages 49–55, San Diego, CA, June 2003.

Top-k Query Processing with Parallel Probing of Data Sources

Adnan Abid and Marco Tagliasacchi

Dipartimento di Elettronica e Informazione – Politecnico di Milano,
Piazza Leonardo da Vinci, 32 – 20133 Milano, Italy
{abid,tagliasa}@elet.polimi.it

Abstract Rank join problem performs a relational join between the given relations by assigning numeric scores to the join results based on the given scoring function, and returns the join results with the highest scores. These rank join operators compute the top-K join results by accessing a subset of input relations. This paper addresses the problem of getting top-K join results from two or more search services characterized by non-negligible response times. There are two main objectives, to minimize the overall data access time, and also avoiding the access to the data that does not contribute to get the top-K join results. This paper proposes a rank join operator that achieves these objectives by using a score guided data pulling strategy. This data pulling strategy minimizes the data access time by extracting the data in parallel from all Web services, while it avoids the access to the data not useful to get the top-K join results by pausing and resuming the data access from these Web services, based on the score values of the retrieved tuples. An extensive experimental study validates the importance of the proposed approach and shows that it accomplishes the task in much less time, while incurring few extra data accesses, as compared to the existing best approaches in terms of making least data accesses.

Keywords: rank joins, rank queries, score guided data pulling, top-K queries

1 Introduction

Consider a person who wants to plan his visit to Paris by searching for a good quality hotel and a restaurant, which are situated close to each other and are highly recommended by their customers. This can be accomplished by extracting information from suitable Web services and merging the information to get the top rated resultant combinations, as contemplated in Search Computing [3]. A sample rank query based on the above example is the following:

SELECT h.name, r.name, 0.6*h.rating+0.4*r.rating as score
FROM Hotels h, Restaurants r
WHERE h.zip = r.zip AND h.city= ‘Paris’ AND r.city = ‘Paris’
RANK BY 0.6*h.rating+0.4*r.rating

The recent solutions to rank join problem [5][6][8] focus on providing instance optimal algorithms regarding the I/O cost i.e. to minimize the amount of data to be accessed in order to find the top-K join results. *Hash Rank Join* (HRJN*) [6] is an instance optimal algorithm in terms of I/O cost and it introduces a physical rank join operator. HRJN* has been further improved in [5] and [8]. These algorithms access data from the data sources in a serial manner, i.e. they access data from one source, process it and then fetch the data from the next *most suitable* source. The latter is selected based on a *pulling strategy*, which determines the source to be accessed to, in order to minimize the I/O cost. However, in the context of using Web services as data sources, data processing time is found to be negligible as compared to data fetching time. So, most of the time is spent in waiting for retrieving the data. This requires the design of a rank join operator that is specifically conceived to meet the objectives of getting top-K join results quickly and restricting access to unwanted data, when using Web services or similar data sources.

2 Preliminaries

Consider a query Q whose answer requires accessing a set of Web services $S_1, ..., S_m$, that can be wrapped to map their data in the form of tuples as in relational databases. Each tuple $t_i \in S_i$ is composed of an identifier, a join attribute, a score attribute and other named attributes. The tuples in every Web service are sorted in descending order of score, where the score reflects the relevance with respect to the query. Let $t_i^{(d)}$ denote a tuple at depth d of S_i. Then $\sigma(t_i^{(d)}) \geq \sigma(t_i^{(d+1)})$, where $\sigma(t_i)$ is the score of the tuple t_i. Without loss of generality, we assume that the scores are normalized in the [0,1] interval.

Each invocation to a Web service S_i retrieves a fixed number of tuples, referred to as chunk. Let (CS_i) denote the *chunk size*, i.e. the number of tuples in a chunk. Furthermore, S_i provides one chunk of tuples in a specified time, which is referred to as its *average response time* (RT_i). Let $t = t_1 \bowtie \mathrm{t}_2 \bowtie ... \mathrm{t}_m$ denote a join result formed by combining the tuples retrieved from the Web services, where t_i is a tuple that belongs to the Web service S_i. This join result is assigned an aggregated score based on a monotone score aggregation function, $\sigma(t) = f(\sigma(t_1), \sigma(t_2), .., \sigma(t_m))$. The join results obtained by joining the data from these Web services are stored in a buffer S_{result} in descending order of their aggregate score.

Bounding Schemes: Let τ_i denotes the local threshold of a Web service S_i which represents an upper bound on the possible score of a join result that can be computed by joining any of the unseen tuples of S_i to either seen or unseen data of the rest of the Web services. The global threshold τ of all the Web services is the maximum among the local thresholds i.e. $\tau = \max\{\tau_1, \tau_2, ..., \tau_m\}$. The bounding scheme is responsible for computing τ, the upper bound of any join result formed by unseen data. The global threshold can be computed based on two possible bounding schemes. The *corner bound* and *tight bound* [5].

3 Methodology

We propose a new rank join operator *Controlled Parallel Rank Join* (cPRJ) which involves Web services characterized by non-negligible response time.

3.1 Data Pulling Strategy

A naïve pulling strategy can be to keep on extracting data from all the Web services in parallel until their local threshold becomes equal or below the score of $K - th$ seen join result i.e. $\tau_i \leq \sigma(t^{(K)}_{result})$. We call this strategy *Parallel Rank Join* (PRJ). However, PRJ may result extracting unwanted data if a Web service stops before the others, that is, its local threshold has reached below the score of, the *then* top-$K - th$ join result in the output buffer S_{result}. In this case, there is a possibility that the other Web services having higher local thresholds produce join results with better aggregate score values and terminate with an even higher local threshold. Resultantly, the Web service which has stopped earlier has incurred extra data fetches. Thus, our proposed data pulling strategy not only uses parallel data extraction to reduce the time to fetch the data, but at the same time, controls over the access to the unwanted data by pausing and resuming data extraction from the Web services.

3.1.1 State Machine: In order to refrain from accessing the data that do not contribute to the top-K join results every Web service is controlled by using a state machine shown in Figure 1. The Web services are assigned a particular state after the completion of data fetch from any Web service. The *Ready* state means that the data extraction call should be made for this Web service. The $Wait$ state means that the data extraction from this Web service should be paused. The *Stop* state means that further data extraction from this Web service will not contribute to determining the top-K join results. Lastly, the $Finish$ state means that all the data from this Web service has been retrieved. The state transitions are exemplified below in Section 3.1.3. On retrieving a chunk of tuples from Web service S_i the following operations are performed in order:

1. Its local threshold τ_i is updated and global threshold is updated accordingly.
2. New join results are computed from the current chunk of S_i and are stored in the buffer S_{result} in descending order of score. The size of the buffer S_{result} is bound by the value of K.
3. All join results having aggregated score above τ are reported to the user.
4. If this is the last chunk then S_i is put to $Finish$ state and τ_i is set to 0.

Apart from this the following operations are also performed:

1. Every Web service S_i, which is not in *Stop* or $Finish$ state, is checked and is put into *Stop* state, if $\sigma(t^{(K)}_{result}) \geq \tau_i$.
2. A Web service S_w that is in $Wait$ state is kept in $Wait$ state, if there is any other Web service S_j which is in *Ready* state and $\tau_j > \tau_w$, and the time needed to bring τ_j less than τ_w, is greater than both RT_w and RT_j. Otherwise, it is put to *Ready* state otherwise.

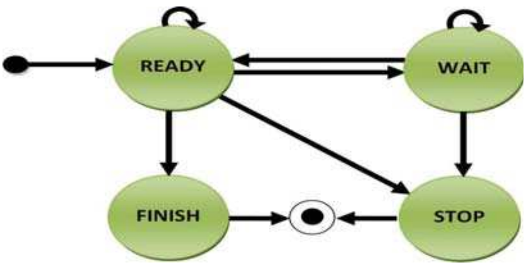


Figure 1. The state machine according to which each Web service is manipulated

3.1.2 Time to Reach (ttr) The decisions to put a service from *Ready* to $Wait$, and $Wait$ to *Ready* state are based on the computation of time to reach (ttr), which is the time to bring down the local threshold of a Web service lower than any other Web service's threshold. The estimation of ttr involves the calculation of decay in score for a Web service S_j having higher local threshold than S_i. We use Autoregressive Moving Average forecasting method [2] for the calculation of score decay. After estimating the unseen score values we can compute the total number of tuples needed to bring the τ_j lower than the value of τ_i. This number is then divided by the *chunk size* of S_j i.e. CS_j, to get the number of *chunks* to bring the threshold down. If number of *chunks* are greater than one then number of *chunks* are multiplied by RT_j, and the elapsed time ET_j, the time since the last data extraction call is issued for S_j is subtracted i.e. $ttr_j = (chunks \times RT_j) - ET_j$, otherwise ttr_j is set to 0.

3.1.3 State Transitions in the State Machine: The state transitions shown in Figure 1 are exemplified below with the help of Figure 2(a). There are 3 Web services S_1, S_2 and S_3 with $RT_1 = 400ms, RT_2 = 700ms$ and $RT_3 = 900ms$, for simplicity, score decay for all Web services is kept linear.

Ready to Finish: If a Web service has been completely exhausted, i.e. all the data from it has been retrieved then its state is changed from *Ready* to $Finish$.

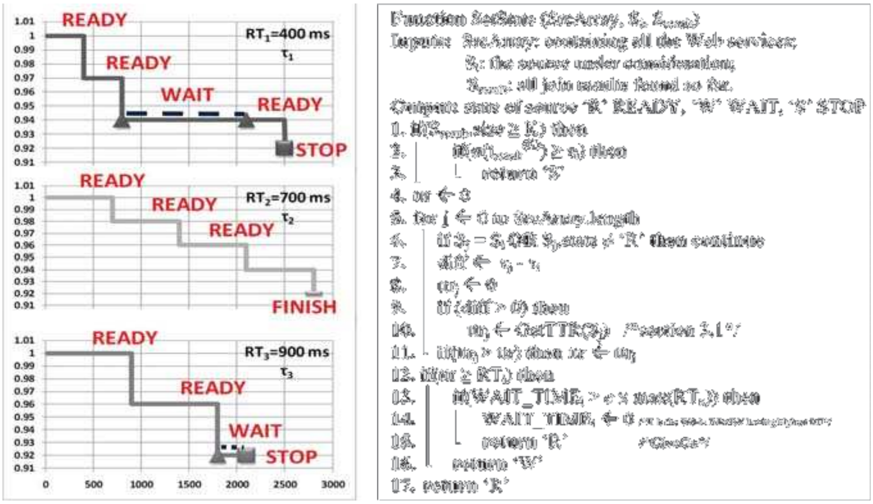
Ready to Stop and Wait to Stop: If a S_i is in *Ready* or $Wait$ states then it should be put into *Stop* state if $\sigma(t^{(K)}_{result}) \geq \tau_i$.

Ready to Ready, Ready to Wait, Wait to Ready and Wait to Wait: A Web service S_i in *Ready* state is put to $Wait$ state, or a Web service S_i in $Wait$ state is put to *Ready* state by analyzing the local thresholds of all other Web services which are in *Ready* state. Figure 2(b) presents the algorithm for *setState* function, lines 4-17 present the algorithm for this process. Below is the explanation of the algorithm for a Web service S_i:

- Consider a set J containing all the Web services having local thresholds greater than that of τ_i and are in *Ready* state. The algorithm estimates the *time to reach* (ttr_j), for every Web service $S_j \in J$ to bring τ_j lower than τ_i as explained in Section 3.1.2. The ttr is the highest value among ttr_j.
- If ttr ≥RT$_i$ then S_i is put to $Wait$ state, otherwise to *Ready* state.

Bootstrapping: The phase before extraction of at least one chunk from all Web services is considered as bootstrapping phase. Any Web service is allowed to make maximum 2 data fetches in this phase.

Fetch Call for Long Awaited Web Services: If a Web service is put to $Wait$ state for a long time in this case the algorithm gives *one* data extraction call to S_i by putting it to *Ready* state. This call is issued if, the $WAIT\_TIME_i$, time since S_i has been retained in $Wait$ state is more than c ×RT$_x$, where S_x, is the Web service with maximum response time which is not in *Stop* or $Finish$ state (lines 13-15 of Figure 2(b)).



```
Function SetState (SrcArray, S_i, Z_result)
Inputs:  SrcArray: containing all the Web services;
         S_i: the source under consideration;
         Z_result: all join results found so far.
Outputs: state of source 'R' READY, 'W' WAIT, 'S' STOP
1.  if(Z_result.size ≥ K) then
2.      if(τ(t_local^{S_i}) ≥ τ_k) then
3.          return 'S'
4.  tr ← 0
5.  for j ← 0 to SrcArray.length
6.      if S_j = S_i OR S_j.state ≠ 'R' then continue
7.      diff ← τ_j - τ_i
8.      tr_j ← 0
9.      if (diff > 0) then
10.         tr_j ← GetTTR(S_j)   /*section 3.1*/
11.     if(tr_j > tr) then tr ← tr_j
12. if(tr ≥ RT_i) then
13.     if(WAIT_TIME_i > c × max(RT_x)) then
14.         WAIT_TIME_i ← 0 [illegible]
15.         return 'R'            /*GiveCall*/
16.     return 'W'
17. return 'R'
```

(a) (b)

Figure 2. (a) Execution of the cPRJ with 3 Web services, over time line against local thresholds. (b) The setState algorithm

4 Experimental Study and Discussion

4.1 Methodology

Data Sets: We have conducted the experiments on both synthetic data, and real Web services. The experiments are based on the query in Example 1 by generating many different synthetic data sources with various parameter settings. The relevant parameters are presented in Table 1. The real Web services used for the experiments are Yahoo! local and Yelp.com for points of interest in a city. We also used eatinparis.com for restaurants and venere.com for hotels. For fairness, we compute these metrics for 5 different cities in case of real Web services, and over 10 different data sets for synthetic data, and report the average.

Table 1. Operating Parameters (defaults in bold)

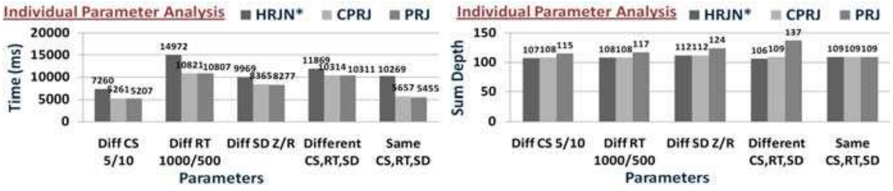
Full Name	Parameter	Tested Values
Number of results	K	1,**20**,50,100
Join Selectivity	JS	0.005, 0.01, **0.015**, 0.02
Score Distribution	SD	Uniform Distrib., **Zipfian Distrib.**, Linear Distrib., Mixed
Response Time	RT	**500/500**, 500/1000, 500/1500
Chunk Size	CS	**5/5**, 5/10, 5/15
Number of relations	m	**2**,3,4

Approaches: We compare three algorithms: HRJN* augmented with a tight bounding scheme, PRJ and the proposed cPRJ. An important consideration is that HRJN* augmented with *tight bound* threshold cannot be beaten in terms of I/O cost, whereas PRJ cannot be out-performed in terms of time taken, provided the time taken for joining the data is negligible.

Evaluation Metrics: The major objective of the proposed approach is to reduce the time taken to get the top-K results by minimizing the data acquisition time with the help of parallelism. So, we consider *time taken* (wall clock time) as the primary metric. The reduction in time is obtained by compromising on possibly some unwanted data extraction. Therefore, we consider *sum depths* [5], total number of tuples retrieved from all Web services, as the other metric.

4.2 Results

Experiments with Synthetic Data: In Figure 3 we show the results of the experiments for CS, RT and SD parameters. Figure 3(b) shows that cPRJ incurs 0-3% more and PRJ incurs 8-10% more I/O cost than HRJN* in case of different values for one of CS, RT and SD. Whereas, If we augment all these in one scenario then cPRJ incurs 3% more I/O cost than HRJN* and PRJ costs 29% more I/O cost than HRJN*. Whereas, Figure 3(a) shows that for all cases the time taken by both parallel approaches is almost same and is 10-40% less than HRJN*.



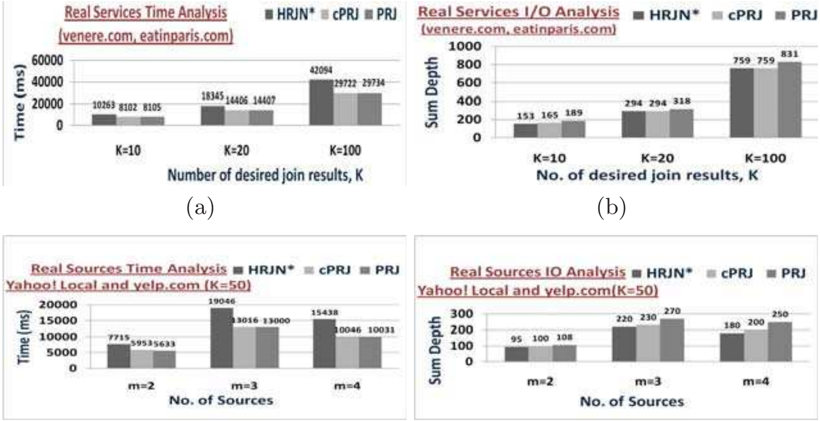
(a) (b)

Figure 3. Performance comparison of the algorithms on synthetic data sources for the parameters show in Table 1.

Real Web Services: We performed experiments with real Web services for different values of K and different values of m. Figures 4(a) and 4(c) show that both parallel approaches take same amount of time which is 20-25% less than HRJN* in case of different values of K and is 14-35% less than HRJN* in case of different values of m. Figures 4(b) and 4(d) show that the I/O cost incurred by proposed cPRJ is 4-11% more than ideal HRJN*, whereas, PRJ takes 8-38% extra data fetches.

As a whole, the overall performance of cPRJ is much better than PRJ as it has almost same I/O cost as of HRJN* whereas, it takes almost same time as of PRJ, whereas, PRJ has higher I/O cost than HRJN*. Thus, in the diverse settings it brings the best of both worlds.

The other three parameters JS, m and K do not have any impact, *individually*. However, if SD, RT and CS have heterogeneous values then JS, m and K also come into play as shown in Figure 4.



(a) (b)

(c) (d)

Figure 4. Performance of the algorithms with real services. Figures (a) and (b) are for the experiments with venere.com and eatinparis.com. Figures (c) and (d) are experiments with different number of sources using Yahoo! Local and yelp.com

5 Related Work

We discuss the existing solutions which involve only sorted access to the data. The NRA algorithm [4] finds the top-K answers by exploiting only sorted accesses to the data. But, this algorithm may not report the exact object scores.

Another example of no random access top-K algorithms is the J* algorithm [1]. It uses a priority queue containing partial and complete join results, sorted on

the upper bounds of their aggregate scores. The algorithm completes and reportes the join result at the top of the queue. This algorithm is expensive in terms of memory and I/O costs as compared to HRJN* in most of the cases.
HRJN [6] is based on symmetrical hash join. The operator maintains a hash table for each relation involved in the join process, and a priority queue to buffer the join results in the order of their scores. It also maintains a threshold τ and uses a data pulling strategy to compute join results. Some recent improvements in HRJN algorithm are presented in [5] and [8]. These algorithms use *tight bound* to compute top-K join results and show their comparative analysis.
Another interesting and objectively similar work has been done in [7], but the proposed algorithm *Upper* incorporates both serial and random accesses to the data sources, whereas, in our case we only use sorted access to the data sources.

6 Conclusion

We have proposed a new rank join algorithm cPRJ, for multi-way rank join while using parallel data access. This algorithm is specifically designed for distributed data sources which have a non-negligible response time e.g. the Web services available on the Internet. It uses a score guided data pulling strategy which helps computing the top-K join results. The results based on the experiments conducted on synthetic and real Web services show that the I/O cost of the proposed approach is nearly as low as optimal I/O cost of HRJN*, and it computes the join results as quick as PRJ approach which cannot be beaten in terms of time taken. As a next step, we anticipate that this parallel rank join operator can be enhanced for pipe joins.

7 Acknowledgments

This research is part of the “Search Computing” project, funded by the European Research Council, under the 2008 Call for “IDEAS Advanced Grants”.

References

1. Nastev A., Chang Y., Smith J. R., Li C, and Vittor J. S. Supporting incremental join queries on ranked inputs. In *VLDB Conference*.
2. P. J. Brockwell. *Encyclopedia of Quantitative Finance*. 2010.
3. Stefano Ceri. *Search Computing: Challenges and Directions*. LNCS. 2010.
4. R. Fagin, A. Lotem, and M. Naor. Optimal aggregation algorithms for middleware. *Journal of computer and system sciences*, 66(4):614–656, 2003.
5. Jonathan Finger and Neoklis Polyzotis. Robust and efficient algorithms for rank join evaluation. In *SIGMOD Conference*, pages 415–428, 2009.
6. I. Ilyas, W. Aref, and A. Elmagarmid. Supporting top-k join queries in relational databases. *The VLDB journal*, 13(3):207–221, 2004.
7. Amélie Marian, Nicolas Bruno, and Luis Gravano. Evaluating top- queries over web-accessible databases. *ACM Trans. Database Syst.*, 29(2):319–362, 2004.
8. Karl Schnaitter and Neoklis Polyzotis. Optimal algorithms for evaluating rank joins in database systems. *ACM Trans. Database Syst.*, 35(1), 2010.

ChaseT: A Tool For Checking Chase Termination

Andrea De Francesco, Francesca Spezzano and Irina Trubitsyna

DEIS, Università della Calabria, 87036 Rende, Italy
{adefrancesco,fspezzano,irina}@deis.unical.it

Abstract. The chase algorithm is a fixpoint algorithm whose aim is to fix inconsistencies of database instances with respect to a set of data dependencies. The chase procedure may be non-terminating and several techniques and criteria for checking chase termination have been proposed. This paper presents *ChaseT*, a tool that allows users to design data dependencies and apply different criteria and algorithms for checking chase termination. Moreover, *ChaseT* is able to execute the chase procedure in order to repair the possible inconsistent database provided by the user. This paper starts introducing the chase algorithm and the techniques for checking chase termination and then focuses on the architecture and the use-case scenarios of *ChaseT*.

1 Introduction

The Chase is a simple fixpoint algorithm, proposed 30 years ago [1, 11], testing and enforcing implication of data dependencies in database systems. It plays important roles in database theory as well as in practical applications such as data exchange, data integration, data warehousing, federated databases and others [2, 4, 5, 10]. It is also used, directly or indirectly, on an everyday basis by people who design databases, and it is used in commercial systems to reason about the consistency and correctness of a data design.

The problem that the chase wants to solve is the following: given a database instance D and a set of constraints Σ, both defined over an underlying database schema $\mathbf{R}$, fix the database consistency if $D \not\models \Sigma$ (i.e. the constraints are not satisfied by the current instance). The algorithm works by inserting tuples, that may contain null values because of the existentially quantification in the constraints, and replacing labeled nulls (with either constants or other labeled nulls) so that the resulting database satisfies Σ. Since, in general, it is undecidable whether the chase algorithm terminates [6], it is important to identify criteria for Σ which guarantee that the chase terminates for all database instances. These criteria ensure to design 'safe' sets of constraints and are useful in all database fields where inconsistencies may arise, including data exchange applications, database integration, data warehousing and federated databases.

The following example shows a set of constraints for which the chase could be non-terminating.

Example 1. Consider the set of constraints Σ_1:

$$\begin{array}{l} \forall x\; S(x) \rightarrow \exists y\; E(x,y) \\ \forall x\, \forall y\; E(x,y) \rightarrow S(y) \end{array}$$

and the database instance consisting of the tuple $S(a)$. Since the database does not satisfy the first constraint, a tuple $E(a, n_1)$ should be inserted, where n_1 is a new labeled null value. At this point the second constraint is not satisfied and the tuple $S(n_1)$ should be added to the database. Continuing with the chase process of adding tuples to make the database consistent, an infinite number of tuples $E(n_1, n_2)$, $S(n_2)$, $E(n_2, n_3), S(n_3), \ldots$ should be inserted. □

Generally, in the literature two different chase procedures are considered: *standard* and *oblivious*. Intuitively, a standard chase step applies only when there exists a mapping from the body of a constraint to the database instance and the head of the constraint is not satisfied, while an oblivious one always applies when there exists the mapping from the body to the instance, even if the constraint is satisfied. Two different types of oblivious chase have been used in the literature: *naive* and *skolem* [13, 12].

Example 2. Consider the constraint

$$r: \ \forall x\, \forall z\ E(x, z) \rightarrow \exists y\ E(x, y)$$

and the database $D = \{E(a, b)\}$. Under the standard chase, the constraint is satisfied and the chase terminates without any application of a chase step. Under the oblivious skolem chase only a tuple $E(a, n_1)$ is added, whereas under the oblivious naive chase an infinite number of tuples $E(a, n_1), E(a, n_2), E(a, n_3)$... is added. □

This paper offers a survey on the well-known chase termination criteria [7, 6, 13, 12] and rewriting approaches [8] and presents *ChaseT*, a tool supporting the application of the chase algorithm. The user can submit the database instance, possibly inconsistent w.r.t. a given set of constraints, and fix its consistency by executing the desired chase procedure. The tool can be also used in database design to avoid the definition of dependencies where the chase could be non-terminating. Indeed, *ChaseT* allows users to design data dependencies and apply different criteria and algorithms for checking chase termination.

Organization. The paper is organized as follows. Section 2 introduces basic notions on chase algorithms. Sections 3 and 4 present a survey on the well-known chase termination conditions and rewriting techniques, respectively. Finally, in Section 5 the *ChaseT* system is described and in Section 6 conclusions are drawn.

2 Preliminaries

We introduce the following disjunct sets of symbols: (i) an infinite set *Consts* of *constants*, (ii) an infinite set *Nulls* of *labeled nulls* and (iii) an infinite set *Vars* of variables.

A *relational schema* $\mathbf{R}$ is a set of relational predicates R, each with its associated arity $ar(R)$. An *instance* of a relational predicate R of arity n is a set of ground atoms in the form $R(c_1, \ldots, c_n)$, where $c_i \in Consts \cup Nulls$. Such (ground) atoms are also called tuples or facts. We denote by D a database instance constructed on $Consts$ and by J, K the database instances constructed on $Consts \cup Nulls$. Given an instance K, $Nulls(K)$ (resp. $Consts(K)$) denotes the set of labeled nulls (resp. constants) occurring in K. An *atomic formula* (or *atom*) is of the form $R(t_1, \ldots, t_n)$ where R is a relational predicate, $t_1, \ldots, t_n$ are terms belonging to the domain $Consts \cup Vars$ and $n = ar(R)$.

Let K be a database over a relational schema $\mathbf{R}$ and $S \subseteq \mathbf{R}$, then $K[S]$ denotes the subset of K consisting of instances whose predicates are in S (clearly $K = K[\mathbf{R}]$). Analogously, if we have a collection of databases $K_C = \{K_1, \ldots, K_n\}$ where each K_i is defined over a schema $\mathbf{R}_i$ and let $S \subseteq \cap_{i \in [1 \ldots n]} \mathbf{R}_i$, then $K_C[S] = \{K_1[S], \ldots, K_n[S]\}$.

Given a relational schema $\mathbf{R}$, a *tuple generating dependency* (TGD) over $\mathbf{R}$ is a formula of the form $\forall \mathbf{x} \forall \mathbf{z}\ \phi(\mathbf{x}, \mathbf{z}) \rightarrow \exists \mathbf{y}\ \psi(\mathbf{x}, \mathbf{y})$, where $\phi(\mathbf{x}, \mathbf{z})$ and $\psi(\mathbf{x}, \mathbf{y})$ are conjunctions of atomic formulas over $\mathbf{R}$; $\phi(\mathbf{x}, \mathbf{z})$ is called the *body* of r, denoted as $Body(r)$, while $\psi(\mathbf{x}, \mathbf{y})$ is called the *head* of r, denoted as $Head(r)$. An *equality generating dependency* (EGD) over $\mathbf{R}$ is a formula of the form $\forall \mathbf{x}\ \phi(\mathbf{x}) \rightarrow x_1 = x_2$, where x_1 and x_2 are among the variables in $\mathbf{x}$.

In the following we will often omit the universal quantification, since we assume that variables appearing in the body are universally quantified and variables appearing only in the head are existentially quantified. In some cases we also assume that the head and body conjunctions are sets of atoms.

Definition 1 (Homomorphism). Let K_1 and K_2 be two instances over $\mathbf{R}$ with values in $Consts \cup Nulls$. A *homomorphism* $h : K_1 \rightarrow K_2$ is a mapping from $Consts(K_1) \cup Nulls(K_1)$ to $Consts(K_2) \cup Nulls(K_2)$ such that: (1) $h(c) = c$, for every $c \in Consts(K_1)$, and (2) for every fact $R_i(t)$ of K_1, we have that $R_i(h(t))$ is a fact of K_2 (where, if $t = (a_1, ..., a_s)$, then $h(t) = (h(a_1), ..., h(a_s))$). □

Similar to homomorphisms between instances, a homomorphism h from a conjunctive formula $\phi(\mathbf{x})$ to an instance J is a mapping from the variables $\mathbf{x}$ to $Consts(J) \cup Nulls(J)$ such that for every atom $R(x_1, \ldots, x_n)$ of $\phi(\mathbf{x})$ the fact $R(h(x_1), \ldots, h(x_n))$ is in J.

For any database instance D and set of constraints Σ over a database schema $\mathbf{R}$, a *solution* for (D, Σ) is an instance J such that $D \subseteq J$ and $J \models \Sigma$ (i.e. J satisfies all constraints in Σ). A *universal solution* J is a solution such that for every solution J' there exists a homomorphism $h : J \rightarrow J'$. The set of universal solutions for (D, Σ) will be denoted by $USol(D, \Sigma)$.

Chase step Let K be a database instance.

1. Let r be a TGD $\phi(\mathbf{x}, \mathbf{z}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y})$. Let h be a homomorphism from $\phi(\mathbf{x}, \mathbf{z})$ to K such that there is no extension of h to a homomorphism h' from $\phi(\mathbf{x}, \mathbf{z}) \wedge \psi(\mathbf{x}, \mathbf{y})$ to K. We say that r can be *applied* to K with homomorphism h. Let K' be the union of K with the set of facts obtained by: (a) extending h to h' such that each variable in $\mathbf{y}$ is assigned a fresh labeled null, followed by (b) taking the image of the atoms of ψ under h'. We say that the result of applying r to K with h is K', and write $K \overset{r,h}{\rightarrow} K'$. (A variant of this step is the *oblivious* one that applies to an instance K if there is a homomorphism h

from $\phi(\mathbf{x})$ to K. We write $K \overset{*,r,h}{\rightarrow} K'$ to denote the application of an oblivious chase step.)

2. Let r be an EGD $\phi(\mathbf{x}) \rightarrow x_1 = x_2$. Let h be a homomorphism from $\phi(\mathbf{x})$ to K such that $h(x_1) \neq h(x_2)$. We say that r can be *applied* to K with homomorphism h. More specifically, we distinguish two cases.
   (a) If both $h(x_1)$ and $h(x_2)$ are in *Consts* the result of applying r to K with h is "failure", and $K \overset{r,h}{\rightarrow} \perp$.
   (b) Otherwise, let K' be K where we identify $h(x_1)$ and $h(x_2)$ as follows: if one is a constant, then the labeled null is replaced everywhere by the constant; if both are labeled nulls, then one is replaced everywhere by the other. We say that the result of applying r to K with h is K', and write $K \overset{r,h}{\rightarrow} K'$.

Definition 2 (Chase [7]). Let Σ be a set of TGDs and EGDs, and let K be an instance.

- A *chase sequence of* K *with* Σ is a sequence (finite or infinite) of chase steps $K_i \overset{r,h_i}{\rightarrow} K_{i+1}$, with $i = 0, 1, ..., K_0 = K$ and r a dependency in Σ.
- A *finite chase of* K *with* Σ is a finite chase sequence $K_i \overset{r,h_i}{\rightarrow} K_{i+1}$, $0 \leq i < m$, with the requirement that either (a) $K_m =\perp$ or (b) there is no dependency r of Σ and there is no homomorphism h_m such that r can be applied to K_m with h_m. We say that K_m is the result of the finite chase. We refer to case (a) as the case of a *failing finite chase* and we refer to case (b) as the case of a *successful finite chase.* □

In [7] it has been shown that, for any instance D and set of constraints Σ: (i) if J is the result of some successful finite chase of $\langle D, \Sigma \rangle$, then J is a universal solution; (ii) if some failing finite chase of $\langle D, \Sigma \rangle$ exists, then there is no solution.

Constraints equivalence The equivalence between two sets of constraints Σ_1 and Σ_2 defined, respectively, over two schemas $\mathbf{R}_1$ and $\mathbf{R}_2$, is given with respect to two sets of relations $R, S \subseteq \mathbf{R}_1 \cap \mathbf{R}_2$ called, respectively, input and output relations.

Given two sets of constraints Σ_1 and Σ_2 over the two database schemas $\mathbf{R}_1$ and $\mathbf{R}_2$, respectively and two sets of relations $R, S \subseteq \mathbf{R}_1 \cap \mathbf{R}_2$, we say that $\langle \mathbf{R}_1, \Sigma_1 \rangle \sqsubseteq_{R/S} \langle \mathbf{R}_2, \Sigma_2 \rangle$ if for every database D over R, $\mathit{USol}(D, \Sigma_1)[S] \subseteq \mathit{USol}(D, \Sigma_2)[S]$. Moreover, we say that $\langle \mathbf{R}_1, \Sigma_1 \rangle$ and $\langle \mathbf{R}_2, \Sigma_2 \rangle$ are equivalent with respect to R/S and write $\langle \mathbf{R}_1, \Sigma_1 \rangle \equiv_{R/S} \langle \mathbf{R}_2, \Sigma_2 \rangle$ if both $\langle \mathbf{R}_1, \Sigma_1 \rangle \sqsubseteq_{R/S} \langle \mathbf{R}_2, \Sigma_2 \rangle$ and $\langle \mathbf{R}_2, \Sigma_2 \rangle \sqsubseteq_{R/S} \langle \mathbf{R}_1, \Sigma_1 \rangle$. When $R = S = \mathbf{R}_1 \cap \mathbf{R}_2$ we simply write $\langle \mathbf{R}_1, \Sigma_1 \rangle \sqsubseteq \langle \mathbf{R}_2, \Sigma_2 \rangle$ and $\langle \mathbf{R}_1, \Sigma_1 \rangle \equiv \langle \mathbf{R}_2, \Sigma_2 \rangle$.

3 Chase termination conditions

This section presents a brief overview on the well-known chase termination conditions that guarantee for every database D the termination of all chase sequences in polynomial time in the size of D.

Weak acyclicity. Fagin et al. [7] introduced a criterium, called *weak acyclicity* (*WA*), checking whether the set of constraints does not present cyclic conditions for which a new null value forces (directly or indirectly) the introduction of another null in the same position. Let Σ be a set of TGDs over a database schema $\mathbf{R}$, then $pos(\Sigma)$ denotes the set of positions R_i such that R denotes a relational predicate of $\mathbf{R}$, i is the i-wise attribute of R, and there is an R-atom appearing in Σ. Weak acyclicity is based on the construction of a directed graph $dep(\Sigma) = (pos(\Sigma), E)$, called the *dependency graph*, where E is defined as follows. For every TGD $\phi(\mathbf{x}, \mathbf{z}) \rightarrow \exists \mathbf{y} \psi(\mathbf{x}, \mathbf{y})$ in Σ, then: i) for every x in $\mathbf{x}$ occurring in position R_i in ϕ and in position S_j in ψ, add an edge $R_i \rightarrow S_j$ (if it does not already exist); ii) for every x in $\mathbf{x}$, appearing in position R_i in ϕ and for every y in $\mathbf{y}$ appearing in position T_k in ψ, add a special edge $R_i \xrightarrow{*} T_k$ (if it does not already exist). Σ is *weakly acyclic* if $dep(\Sigma)$ has no cycle going through a special edge.

Stratification. Deutsch et al. proposed an extension of weak acyclicity called *stratification* (Str) [6]. The extension states that chase termination can be checked locally by considering subsets of constraints that might cyclically cause to fire each other. Given a set of constraints Σ and two constraints $r_1, r_2 \in \Sigma$, we say that $r_1 \prec r_2$ iff there exists a relational database instance K_1 and two homomorphisms h_1 and h_2 such that i) $K_1 \not\models h_1(r_1)$ ii) $K_1 \xrightarrow{r_1, h_1} K_2$, iii) $K_2 \not\models h_2(r_2)$ and iv) $K_1 \models h_2(r_2)$. Intuitively, $r_1 \prec r_2$ means that firing r_1 can cause the firing of r_2. We say that Σ is stratified iff the constraints in every cycle of the *chase graph* $G(\Sigma) = (\Sigma, \{(r_1, r_2) | r_1 \prec r_2\})$ are weakly acyclic.

Example 3. Consider the set Σ_3:

$$r: \; E(x, y) \wedge E(y, x) \rightarrow \exists u, v \; E(x, u) \wedge E(u, v) \wedge E(v, x)$$

stating that each node involved in a cycle of length 2 is also involved in a cycle of length 3 and the two cycles share an edge. Since $r \not\prec r$ (i.e. r does not fire itself) the chase graph $G(\Sigma_3)$ is acyclic and, therefore, Σ_3 is stratified. □

For every database D, stratification ensures the existence of a chase sequence which terminates in polynomial time in the size of D [6]. In order to guarantee the termination of all chase sequences, since stratification does not, the improvement of stratification criterium, called *c-stratification* ($CStr$) has been proposed [13]. Basically, c-stratification defines a different chase graph G_c and applies a constraints whenever the body of a constraints is satisfied. In particular, $r_1 \prec_c r_2$ iff i) $K_1 \xrightarrow{*, r_1, h_1} K_2$, ii) $K_2 \not\models h_2(r_2)$ and iii) $K_1 \models h_2(r_2)$.

Safety. Meier et al. proposed a different extension of weak acyclicity called *safety* ($\mathcal{SC}$) [13]. The improvement is based on the fact that only the effective propagation of null values should be considered in the graph. A position R_i is said to be *affected* if there is a constraint $r : \phi(\mathbf{x}, \mathbf{z}) \rightarrow \exists \mathbf{y} \, \psi(\mathbf{x}, \mathbf{y})$ in Σ and either i) there is a variable y in $\mathbf{y}$ appearing in position R_i in ψ, or ii) there is a variable x in $\mathbf{x}$ appearing both in position R_i in ψ and only in affected positions in the body of r. The set of affected positions of Σ is denoted by $aff(\Sigma)$.

Given a set of TGDs $\varSigma$, the *propagation graph* of $\varSigma$, denoted as $prop(\varSigma) = (aff(\varSigma), E')$, is a subset of $dep(\varSigma) = (pos(\varSigma), E)$ such that E' contains the edges in E whose positions are affected (since $aff(\varSigma) \subseteq pos(\varSigma)$). Moreover, $\varSigma$ is said to be safe if $prop(\varSigma)$ does not contain cycles with special edges.

Example 4. Consider the below set of constraints $\varSigma_4$:

$$\begin{array}{l} S(x) \rightarrow \exists y\ E(x,y) \\ N(y) \wedge E(x,y) \rightarrow S(y) \end{array}$$

which is not weakly acyclic since $S_1 \xrightarrow{*} E_2$ and $E_2 \rightarrow S_1$. However, E_2 does not propagate null values to S_1 as the variable y also appears in the relation N which does not contain null values. So, position S_1 cannot contain nulls as well, i.e. it is not affected, and $\varSigma_4$ is safe. □

Basically, safety conditions consider only affected positions, i.e. positions which may actually contain null values [3]. Both stratification and safety extend weak acyclicity, but they are not comparable (i.e. there are sets of constraints which only satisfy one of the two criteria).

Super-weak acyclicity. A different extension has been introduced in [12] under the name of *Super-weak Acyclicity* (SwA). Basically, SwA takes into account the fact that variables may appear more than once in the body of constraints and, therefore, when different nulls are inserted in positions associated with the same variable, constraints are not fired. The super-weak acyclicity builds a *trigger graph* $\varUpsilon(\varSigma) = (\varSigma, E)$ where edges define relations among constraints. An edge $r_i \rightsquigarrow r_j$ means that a null value introduced by a constraint r_i is propagated (directly or indirectly) into the body of r_j.

Example 5. Let $\varSigma_5$ be the below set of constraints:

$$\begin{array}{l} N(x) \rightarrow \exists y,z\ E(x,y,z) \\ E(x,y,y) \rightarrow N(y) \end{array}$$

$\varSigma_5$ is super-weakly acyclic and stratified but not safe. □

Let $\varSigma$ be a set of TGDs and let $sk(\varSigma)$ be the logic program obtained by skolemizing $\varSigma$, i.e. by replacing each existentially quantified variable y appearing in the head of a TGD r by the skolem function $f_y^r(\mathbf{x})$, where $\mathbf{x}$ is the set of variables appearing both in the body and in the head of r. A *place* is a pair (a,i) where a is an atom of $sk(\varSigma)$ and $0 \leq i \leq ar(a)$. Given a TGD r and an existential variable y in the head of r, we define $Out(r,y)$ as the set of places (called *output places*) in the head of $sk(r)$ where a term of the form $f_y^r(\mathbf{x})$ occurs. Given a TGD r' and a universal variable x' of r', $In(r',x')$ denotes the set of places (called *input places*) in the body of r' where x' occurs.

Given a set of variables V, a substitution θ of V is a function mapping each $v \in V$ to a finite term $\theta(v)$ built upon constants and function symbols. Two places (a,i) and (a',i) are unifiable and we write $(a,i) \sim (a',i)$ iff there exist two substitutions θ and θ' of (respectively) the variables a and a' such that $a[\theta] = a'[\theta']$. Given two sets of places Q and Q' we write $Q \sqsubseteq Q'$ iff for all $q \in Q$ there exists some $q' \in Q'$ such that $q \sim q'$.

Given a set Q of places, $Move(\Sigma, Q)$ denotes the smallest set of places Q' such that $Q \subseteq Q'$, and for every rule $r = B_r \rightarrow H_r$ in $sk(\Sigma)$ and every variable x, if $\Pi_x(B_r) \sqsubseteq Q'$ then $\Pi_x(H_r) \subseteq Q'$, where $\Pi_x(B_r)$ and $\Pi_x(H_r)$ denote the sets of places in B_r and H_r where x occurs.

Given a set Σ of TGDs and two TGDs $r, r' \in \Sigma$, we say that r triggers r' in Σ and we write $r \rightsquigarrow r'$ iff there exists an existential variable y in the head of r, and a universal variable x' occurring both in the body and head of r' such that $In(r'x') \sqsubseteq Move(\Sigma, Out(r, y))$. A set of constraints Σ is super-weakly acyclic iff the trigger graph $\Upsilon(\Sigma) = (\Sigma, \{(r, r')|r \rightsquigarrow r'\})$ is acyclic.

The super-weak acyclicity extends safety [8] but is not comparable with (c)-stratification.

Other criteria guaranteeing chase termination in PTIME have been also proposed. In particular, a criterium generalizing both c-stratification and safety, but not comparable with SwA, has been introduced in [13] under the name of *inductive restriction* (IR). In addition, the criterium for the *Termination of Oblivious Chase* (TOC) has been defined in [12] as well; it is more general than SwA, but undecidable. Recently new criteria have been proposed in [9]. More specifically, a more powerful notion of c-stratification, called *WA-Stratification* has been given, and a new criterium, called *local stratification* (LS), combining the chase graph of WA-stratification and the unifying notion of SwA, has been proposed as well. LS results more general than IR, WA-stratification and SwA.

4 Rewriting approach

In [8] an orthogonal technique has been proposed which is able to extend all criteria above discussed. The technique consists in rewriting the original set of constraints Σ into an 'equivalent' set Σ^α, where predicate symbols are adorned, and verifying the structural properties for chase termination on Σ^α. More specifically, an adorned predicate is of the form $p^{\alpha_1 \cdots \alpha_m}(x_1, ..., x_m)$ where $\alpha_i = b$ means that the variable x_i is bounded, otherwise ($\alpha_i = f$) we say that x_i is free.

Intuitively, bounded terms can take values from finite domains; consequently, constant terms are always adorned with the symbol b. If each body variable of a TGD is associated with a unique adornment we say that the adornment of the body is *coherent*. Given a TGD $r : \phi(\mathbf{x}, \mathbf{z}) \rightarrow \exists \mathbf{y}\, \psi(\mathbf{x}, \mathbf{y})$ and let α be a coherent adornment for the body atoms, then $HeadAdn(r, \phi^\alpha(\mathbf{x}, \mathbf{z}))$ denotes the *adorned head* of r (with respect to the adorned body $\phi^\alpha(\mathbf{x}, \mathbf{z})$) obtained by adorning head atoms as follows: i) every universally quantified variable has the same adornment of the body occurrences, ii) constants are adorned as b; iii) existentially quantified variables are adorned as f.

Rewriting algorithm. Given a set of TGDs Σ over a schema $\mathbf{R}$ the corresponding rewriting set $Adn(\Sigma)$ consists of the union of four sets of TGDs: the base set $Base(\Sigma)$, the derived set $Derived(\Sigma)$, the input set $In(\Sigma)$ and the output set $Out(\Sigma)$. The rewriting is performed by means of the function Adn. It starts by adorning, for each TGD, body predicates with strings of b symbols and adorning heads according to the body adornments by using the function *HeadAdn* (base set); then, each new adorned predicate symbol is used to generate new adorned constraints until all adorned predicate symbols are used (derived set); at the end, TGDs mapping source relations into relations adorned with strings of b symbols (input set) and TGDs mapping relations having the same predicate and different adornments into a unique relation (output set) are added.

Example 6. Consider the following set of constraints Σ_6:

$$\begin{array}{l} N(x) \rightarrow \exists y\ E(x,y) \\ S(x) \wedge E(x,y) \rightarrow N(y) \end{array}$$

Initially, $Adn(\Sigma_6)$ contains two constraints derived by adorning the body variables as bound ($Base(\Sigma_6)$)

$$\begin{array}{l} r_1 :\ N^b(x) \rightarrow \exists y\ E^{bf}(x,y) \\ r_2 :\ S^b(x) \wedge E^{bb}(x,y) \rightarrow N^b(y) \end{array}$$

In the second step two new constraints are generated ($Derived(\Sigma_6)$). Due to the new predicate E^{bf}, the following constraint, derived from constraint r_2, has been introduced:

$$r_3 :\ S^b(x) \wedge E^{bf}(x,y) \rightarrow N^f(y)$$

At this point the new predicate symbol N^f has been generated and, thus, a new constraint derived from r_1 is added:

$$r_4 :\ N^f(x) \rightarrow \exists y\ E^{ff}(x,y)$$

From the new predicate E^{ff} no new constraint is generated since the variable x in the body of the second constraint is bounded as it also appears in the predicate S^b. The rewritten set of constraints is weakly acyclic. □

To show the equivalence between Σ and Σ^α the following definition has been introduced. For any input database schema $\mathbf{R}$ and set of constraints Σ over $\mathbf{R}$, we shall denote with (i) $\hat{\mathbf{R}} = \{\hat{p}(A_1, ..., A_n) \mid p(A_1, ..., A_n) \in \mathbf{R}\}$ the output schema derived from $\mathbf{R}$, (ii) $Adn(\mathbf{R}, \Sigma) = \mathbf{R} \cup \{p^\alpha(A_1, ..., A_n) \mid p(A_1, ..., A_n) \in \mathbf{R} \wedge p^\alpha$ *appears in* $Adn(\Sigma)\} \cup \hat{\mathbf{R}}$ the schema obtained by adding to $\mathbf{R}$ the schemas of the relations introduced in the rewriting of constraints, (iii) $Map(\mathbf{R}) = \mathbf{R} \cup \hat{\mathbf{R}}$ the union of the input and output schemas, and (iv) $Map(\Sigma) = \Sigma \cup \{p(x_1, ..., x_n) \rightarrow \hat{p}(x_1, ..., x_n) | p(A_1, ..., A_n) \in \mathbf{R}\}$ the set of constraints containing, in addition to Σ, a set of TGDs mapping tuples over the input schema to tuples over the output schema.

Theorem 1. [8] For every set of TGDs Σ over a database schema $\mathbf{R}$,

$$\langle Map(\mathbf{R}), Map(\Sigma)\rangle \equiv_{\mathbf{R}/\hat{\mathbf{R}}} \langle Adn(\mathbf{R}, \Sigma), Adn(\Sigma)\rangle.$$

The previous theorem states that for every database D over a schema $\mathbf{R}$ and for each universal solution J derived by applying the source TGDs Σ to D there is a universal solution K derived by applying the rewritten constraints $Adn(\Sigma)$ to D such that $J[\hat{\mathbf{R}}] = K[\hat{\mathbf{R}}]$ and vice versa.

It is worth noting that if Σ satisfies a chase termination criterium $\mathcal{T}$, defined on the base of structural properties, then the rewritten set Σ^α satisfies $\mathcal{T}$ as well, but the vice versa is not true, that is there are significant classes of constraints for which Σ^α satisfies $\mathcal{T}$ and Σ does not.

Improvements (Adn$^+$ function [8]). The rewriting technique can be further improved by using different types of adornments for free variables. Thus, instead of simply using f to denote that a position may contain null values, we will use adornments of the form f_i, where i is a fresh subscript. In order to have terminating sequences we will also use substitutions for adornments sequences and we will add a constraint only if there is no adorned constraint having similar structure, i.e. there is no substitution θ such that $r^\alpha\theta = r^\beta$. In particular, a substitution θ is a set of pairs f_i/f_j such that $i \neq j$; obviously, the same symbol cannot be used in both left and right sides of substitutions, i.e. a symbol f_j used to replace a symbol f_i cannot be substituted in θ by a symbol f_k.

Given a set of TGDs Σ over a schema $\mathbf{R}$ we denote by $Adn^+(\Sigma)$ the corresponding set of constraints obtained by rewriting Σ by means of the function Adn^+. Moreover, the new schema $Adn^+(\mathbf{R}, \Sigma)$ is defined analogously as $Adn(\mathbf{R}, \Sigma)$.

Theorem 2. [8] For every set of TGDs Σ over a database schema $\mathbf{R}$,

$$\langle Map(\mathbf{R}), Map(\Sigma)\rangle \equiv_{\mathbf{R}/\hat{\mathbf{R}}} \langle Adn^+(\mathbf{R}, \Sigma), Adn^+(\Sigma)\rangle.$$

The following theorem states that the rewriting of constraints allows us to recognize larger classes of constraints for which chase termination is guaranteed.

Theorem 3. *[8] For* $T \in \{\mathcal{WA}, \mathcal{S}tr, \mathcal{CS}tr, \mathcal{SC}, \mathcal{S}w\mathcal{A}\}$, $\mathcal{T} \subsetneq Adn\text{-}\mathcal{T} \subsetneq Adn^+\text{-}\mathcal{T}$. □

5 System Description

ChaseT implements the above criteria and techniques for checking chase termination and allows users to execute chase algorithms. Its architecture is depicted in Fig. 1 and consists of six main modules which allow users to define data dependencies, check chase termination properties, visualize explanations and execute chase algorithms. The Graphical User Interface (GUI) allows the user to provide the set Σ of data dependencies and three parameters: i) γ, denoting the type of chase she/he is interested in (standard, skolem oblivious, naive oblivious), ii) τ, denoting the selected termination criterium (*WA*, *SC*, *SwA* or *CStr*), and iii) ρ, denoting the possible rewriting technique she/he wants to apply (Adn, Adn+ or none). Through the GUI is is also possible to submit a database instance D (stored as set of facts in a text file) and fix its possible inconsistencies by running the desired chase algorithm. If the user wants to check the termination of the selected chase algorithm by applying a rewriting technique, the input set of dependencies Σ is rewritten (by the module *Rewriter*) into a set of adorned dependencies Σ^α. Since the rewriting output also depends on the particular chase procedure the user wants to check, the Rewriter module receives in input Σ and

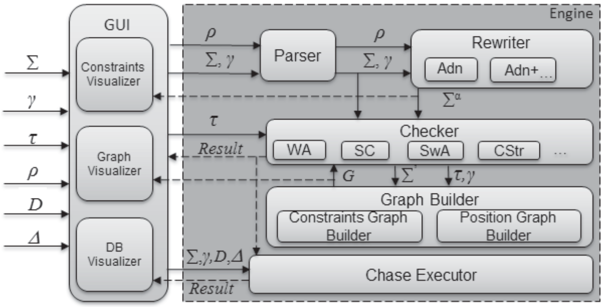


Fig. 1. *ChaseT* architecture

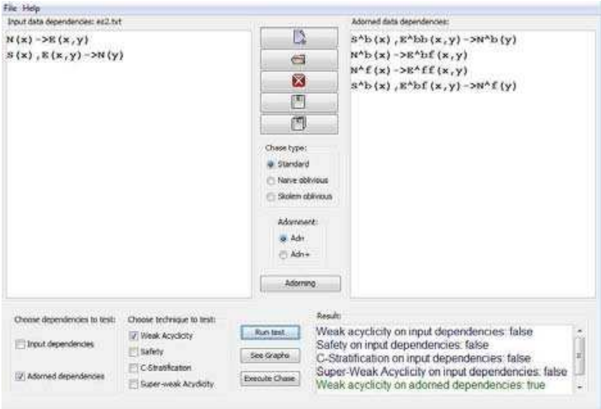
the parameters ρ and γ and gives in output the rewritten set of constraints Σ^α. The system also allows users to check termination conditions without indicating any specific criterium. In such a case all techniques are applied and the system returns the properties of the input dependencies (see the bottom right window in Fig. 2). Fig. 2 shows how the user interacts with the system through the GUI. The left window shows the input set of dependencies, while the rewritten set of dependencies is showed in the right window; parameters are introduced through check boxes.

It is worth noting that the data dependencies defined by the user are first parsed (by the module *Parser*) to check syntactic errors and inconsistencies (e.g. the use of predicates having the same name and different arity).

For the analysis of the structural properties of a set of dependencies Σ, the *Checker* builds two specific graphs for the selected criterium: the *constraints graph* shows how constraints may activate each other, while the *position graph* shows how nulls may propagate through positions. For instance, the constraints graph denotes i) the chase graph if the underlying criterium is c-stratification, and ii) the trigger graph if the underlying criterium is super-weak acyclicity. Concerning the position graph, it could denote the dependency graph or the propagation graph according to the selected criterium. The construction of the graphs is performed by the module *Graph Builder* which receives in input the set of dependencies Σ, the criterium τ and the type of chase γ. The graphs can be visualized using a graph visualization tool for a better understanding of data dependency properties (see Fig. 3).

The module *Chase Executor* applies the desired chase algorithm (specified by the parameter γ) to an input database D and a set of data dependencies Σ; the result of the execution is showed by the *DB Visualizer* and stored in a new text file. This module also receives in input a time limit Δ used for dependencies which are not recognized as terminating by the *Checker* to stop possible non terminating executions.

The system has been developed in Java using *Eclipse IDE* and is downloadable from *wwwinfo.deis.unical.it/chaset.* The GUI has been written using the *Swing Java* libraries and the open source library *JGraphX* for the visualization of graphs. The interactions among the different modules are carried out through



Input data dependencies: es2.txt

```
N(x)->E(x,y)
S(x),E(x,y)->N(y)
```

Adorned data dependencies:

```
S^b(x),E^bb(x,y)->N^b(y)
N^b(x)->E^bf(x,y)
N^f(x)->E^ff(x,y)
S^b(x),E^bf(x,y)->N^f(y)
```

Chase type:
● Standard
○ Naive oblivious
○ Skolem oblivious

Adornment:
● Adn
○ Adn+

Choose dependencies to test:
□ Input dependencies
☑ Adorned dependencies

Choose technique to test:
☑ Weak Acyclicity
□ Safety
□ C-Stratification
□ Super-weak Acyclicity

Result:

```
Weak acyclicity on input dependencies: false
Safety on input dependencies: false
C-Stratification on input dependencies: false
Super-Weak Acyclicity on input dependencies: false
Weak acyclicity on adorned dependencies: true
```

Fig. 2. *ChaseT* User Interface

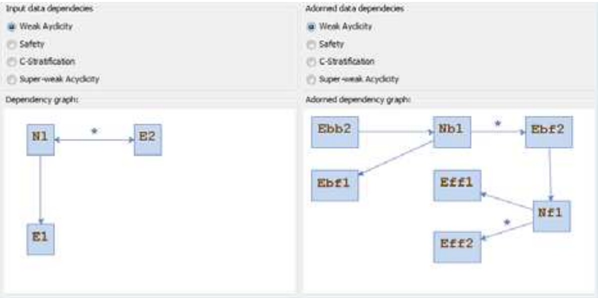
interfaces so that each module can be easily modified without any inference on the other modules.

In the following, a typical use-case scenario of *ChaseT* is shown. Suppose, for instance, that the user wants to check the termination of the standard chase procedure for the set of constraints of Example 6. As shown in Fig. 2, the user introduces the set of constraints Σ in the "*Input data dependency*" window, selects "*standard*" as chase type and tries to test the known termination conditions (*Run test* button) by taking into account that more general techniques require greater computational effort and the explanation is more complex. For the example shown in Fig. 2 the application of all termination conditions to the original set of dependencies produces a negative result. However, the application of the simplest rewriting generates the equivalent set $Adn(\Sigma)$, which is weakly acyclic. The rewritten set of constraints can be visualized in the "*Adorned data dependency*" window (using the *Adorning* button).

The user can also visualize on the screen the graphs generated by the *Graph Builder* in order to analyze graphs and understand the behaviors of the different termination criteria. As shown in Fig 3, the dependency graph of Σ contains a cycle going through a special edge ("*Dependency graph*" window) instead the dependency graph of $Adn(\Sigma)$ is acyclic (see "*Adorned dependency graph*" window). In order to execute the chase algorithm (*Execute Chase* button) the user also loads a database instance and fixes the time limit Δ in the "*Chase Executor*" window. The result can be viewed in the "*Result*" window and stored in the selected textual file.

6 Conclusion

This paper has presented a tool supporting the application of the chase algorithm. In particular, we have reported the chase termination criteria and the rewriting techniques implemented in the current version of *ChaseT* and then



Input data dependecies

● Weak Aydicity
○ Safety
○ C-Stratification
○ Super-weak Acyclicity

Adorned data dependecies

● Weak Aydicity
○ Safety
○ C-Stratification
○ Super-weak Acyclicity

Fig. 3. Graph visualizer

described the architecture of the system and the use-case scenarios. As future work we plan to extend the set of chase termination criteria with new criteria and techniques proposed in [9].

References

1. Alfred V. Aho, Catriel Beeri, and Jeffrey D. Ullman. The theory of joins in relational databases. *ACM Trans. Database Syst.*, 4(3):297–314, 1979.
2. Leopoldo E. Bertossi. Consistent query answering in databases. *SIGMOD Record*, 35(2):68–76, 2006.
3. Andrea Calì, Georg Gottlob, and Michael Kifer. Taming the infinite chase: Query answering under expressive relational constraints. In *Description Logics*, 2008.
4. Jan Chomicki. Consistent query answering: Five easy pieces. In *ICDT*, pages 1–17, 2007.
5. Giuseppe DeGiacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. On reconciling data exchange, data integration, and peer data management. In *PODS*, pages 133–142, 2007.
6. Alin Deutsch, Alan Nash, and Jeffrey B. Remmel. The chase revisited. In *PODS*, pages 149–158, 2008.
7. Ronald Fagin, Phokion G. Kolaitis, Renée J. Miller, and Lucian Popa. Data exchange: semantics and query answering. *Theor. Comput. Sci.*, 336(1):89–124, 2005.
8. Sergio Greco and Francesca Spezzano. Chase termination: A constraints rewriting approach. *PVLDB*, 3(1):93–104, 2010.
9. Sergio Greco, Francesca Spezzano, and Irina Trubitsyna. Stratification criteria and rewriting techniques for checking chase termination. *PVLDB*, 4(7), 2011. *To appear.*
10. Maurizio Lenzerini. Data integration: A theoretical perspective. In *PODS*, pages 233–246, 2002.
11. David Maier, Alberto O. Mendelzon, and Yehoshua Sagiv. Testing implications of data dependencies. *ACM Trans. Database Syst.*, 4(4):455–469, 1979.
12. Bruno Marnette. Generalized schema-mappings: from termination to tractability. In *PODS*, pages 13–22, 2009.
13. Michael Meier, Michael Schmidt, and Georg Lausen. On chase termination beyond stratification. *CoRR*, abs/0906.4228, 2009.

The Open Source release of the MOMIS Data Integration System

Sonia Bergamaschi[1], Domenico Beneventano[1], Alberto Corni[1], Entela Kazazi[2], Mirko Orsini[2], Laura Po[1], Serena Sorrentino[1]

[1] DII, Università di Modena e Reggio Emilia
Via Vignolese 905, 41125 Modena - Italy
firstname.lastname@unimore.it
[2] Data River S.r.l.,
Via Vignolese 905, 41125 Modena - Italy
firstname.lastname@datariver.it

Abstract. MOMIS (Mediator EnvirOnment for Multiple Information Sources) is an Open Source Data Integration system able to aggregate data coming from heterogeneous data sources (structured and semistructured) in a semi-automatic way. DataRiver[3] is a Spin-Off of the University of Modena and Reggio Emilia that has re-engineered the MOMIS system, and released its Open Source version both for commercial and academic use. The MOMIS system has been extended with a set of features to minimize the integration process costs, exploiting the semantics of the data sources and optimizing each integration phase. The Open Source MOMIS system have been successfully applied in several industrial sectors: Medical, Agro-food, Tourism, Textile, Mechanical, Logistics. This paper describes the features of the Open Source MOMIS system and how it is able to address real data integration challenges.

1 Introduction

Data integration involves combining data residing on different sources and providing users with a unified view of these data [9]. In the past years the problem of data integration has been largely discussed by the research community [8]. With the increasing volumes of data and the growing need to share information, data integration has become a fundamental process in the field of business applications.

The size of the market for data integration tools has been estimated at approximately $1.35 billion as of the end of 2009. A projected five-year compound annual rate of approximately 9.4% will yield a market of approximately $2.1 billion by 2014 [14].

As pointed out in [14], “Customers are seeking low-cost, good enough data integration capabilities”. MOMIS is distributed by DataRiver as an Open Source

[3] http://www.datariver.it/

tool to be competitive compared to big vendors and to benefit from the contribution of the Open Source community that can develop and make available extensions of the system.

The goal of the Open Source MOMIS system is the minimization of the integration process costs. A data integration project is often developed by integration designers that have a partial knowledge of the data sources and of the application domain. Even if the designers have a good knowledge of the application domain, often they are not skilled on the techniques to integrate the data stored in the data sources. In traditional Data Integration Systems, designers have to manually build the integrated schema, defining all the mappings between each global class/attribute and the corresponding local classes/attributes on the local data sources, thus the integration process requires several days/weeks depending on the size of the integration project. Another drawback is due to the fact that designers can see the global result of the integration only at the end of the overall integration process, and it is only at that time that they can refine mappings in order to improve the integrated schema.

To overcome these problems, a first result of integration is semi-automatically derived by MOMIS and proposed to the designer in few minutes; she/he can then improve this integration result, through an iterative refinement process and a set of features (described in details in Section 3). The main features proposed in MOMIS are: (1) a GUI that facilitates the integration process, (2) a set of explore and preview tools that allow the designer to preview the integration result during each phase, (3) the possibility to create different unified views to explore the global result of the data integration process, (4) a suite of tools to semantically annotate data sources w.r.t. a common lexical reference; these tools allow the designer to import/export the local source annotations, and permit to extend the lexical reference itself with domain glossaries, (5) a preview of the query plan that allows the designer to visualize, for each executed global query, the set of queries that compose the query plan.

Commercial and Open Source Data Integration Systems on the market, do not provide the semi-automatic generation of the Global Schema and the automatic generation of mappings. The MOMIS System helps the designer integrating data sources in a semi-automaic way, exploiting the semantics of data sources. In [3], the MOMIS system have been demonstrated to be able to support all the twelve queries of the THALIA benchmark for data Integration Systems, by simply extending and combining the declarative translation functions available in MOMIS and without any overhead of new code.

MOMIS development started in 1997 and the research activity continued within several national and international projects through the years. MOMIS was successfully exploited in several scenarios, e.g., for the integration of molecular and phenotypic data sources and the development of an integrated information system for cereals breeders in the CEREALAB project[4] and for the integration of several tourism web sites and the development of a Tourism Vertical Web Portal

[4] http://www.cerealab.unimore.it/

in the WISDOM project[5]. Moreover, the Open Source MOMIS system has been used on real-data sets to integrate clinical data of patients. This work has been conducted in the Olive Tree Project for sharing data about Cancer registries of ten different countries in the Mediterranean Sea. The application of the MOMIS system have been demonstrated to be effective with a considerable saving in time, compared to the manual building of the integrated schema implemented in traditional Data Integration Systems.

The paper is organized as follows: Section 2 presents the MOMIS system, by describing its architecture and by identifying the main phases of the data integration process. In Section 3, we describe the features that have been introduced in the open source version of the tool. In Section 4, the web site, documentation, tutorials and community of the Open Source MOMIS system are presented. At the end, Section 5 sketches out the future development directions.

2 The Data Integration Process and Architecture

In this section, we present the MOMIS architecture and the main phases of the data integration process. A full and detailed description of MOMIS is out of our scope and can be found in [6, 2, 5].

MOMIS builds a unified schema, called Global Schema (GS), of several (heterogeneous) data sources (also called local sources), and allows users to formulate queries on it. It follows a Global-As-View (GAV) approach for the definition of mappings between the GS and local schemas: the GS is expressed in terms of the local schemas. MOMIS performs data integration following a virtual approach that preserves the autonomy and security of the original data sources. The GS generation process is composed by four main phases:

1. **Local Source Upload:** (see Figure 1-1) the integrator designer exploits the *wrapper* tool (see Figure 2) to logically extract the schema of each local source and convert it into the common language ODL$_{I^3}$[6].
2. **Local Source Annotation:** (see Figure 1-2) the designer is asked to *annotate* the local sources, i.e. to associate to class and attribute names (in the following also called terms) one or more meanings w.r.t. a common lexical reference, that in our case is the lexical database WordNet [10]. WordNet is a thesaurus for the English language, that groups terms (called lemmas in the WordNet terminology) into sets of synonyms called synsets, provides short definitions (called gloss), and connects the synsets through a wide network of semantic relationships.
   The designer can manually select a base form and the appropriate WordNet meaning(s) (i.e. synset(s)) for each term and/or perform automatic annotation (see Section 3). Moreover, in the MOMIS Open Source version, the designer can extend WordNet with domain glossaries (see Section 3). The

[5] http://www.dbgroup.unimo.it/wisdom/

[6] ODL$_{I^3}$is an object-oriented language, with an underlying Description Logic, deriving from the standard ODMG.

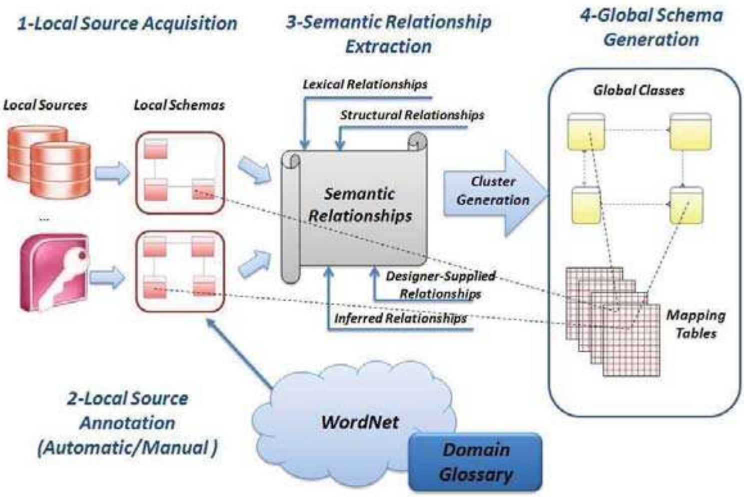


Fig. 1. The MOMIS data integration process.

Local Source Annotation phase is performed by the *Global Schema Designer* tool (see Figure 2).

3. **Semantic Relationships Extraction:** (see Figure 1-3) starting from the annotated local schemas, MOMIS derives a set of intra and inter-schema semantic relationships in the form of: synonyms (SYN), broader terms/narrower terms (BT/NT) and related terms (RT) relationships. The set of semantic relationships is incrementally built by adding: *structural* relationships (deriving from the structure of each schema), *lexical* relationships (deriving from the element annotations, by exploiting the WordNet semantic network), *designer-supplied* relationships (representing specific domain knowledge) and *inferred* relationships (deriving from Description Logics equivalence and subsumption computation). The Semantic Relationship Extraction phase is performed by the Global Schema Designer tool (see Figure 2).
4. **GS generation:** starting from the discovered semantic relationships and the local source schemas, MOMIS generates a GS consisting of a set of global classes, plus *Mapping Tables* which contain the mappings to connect the global attributes of each global class with the local source attributes . The GS generation is a process where classes describing the same or semantically related concepts in different sources are identified and clusterized into the same global class (see Figure 1-4).

   The designer may interactively refine and complete the proposed integration result through the GUI provided by the Global Schema Designer tool. In particular, he can: modify the proposed global classes and mappings; select the appropriate *Join Function* for each global class; define *Transformation Functions* in order to transform the local attribute values into the corresponding global attribute values; and solve possible data conflicts through

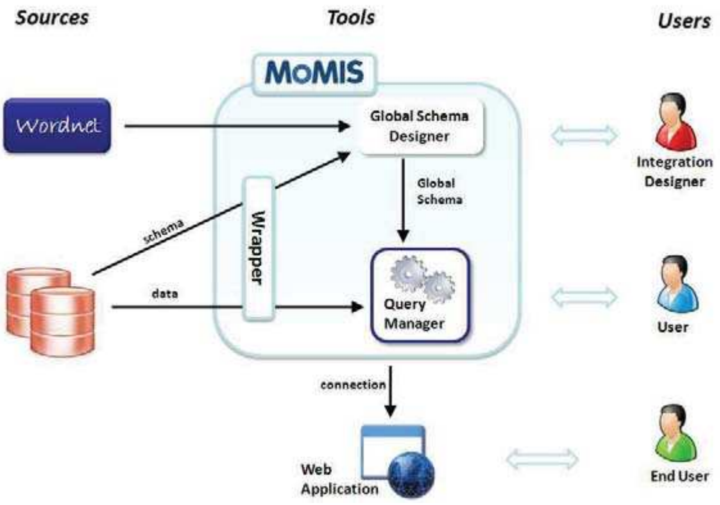


Fig. 2. The MOMIS architecture.

the definition of *Resolution Functions* (applied to each global attribute to obtain, starting from the values computed by the Transformation Functions the corresponding value of the global attribute).

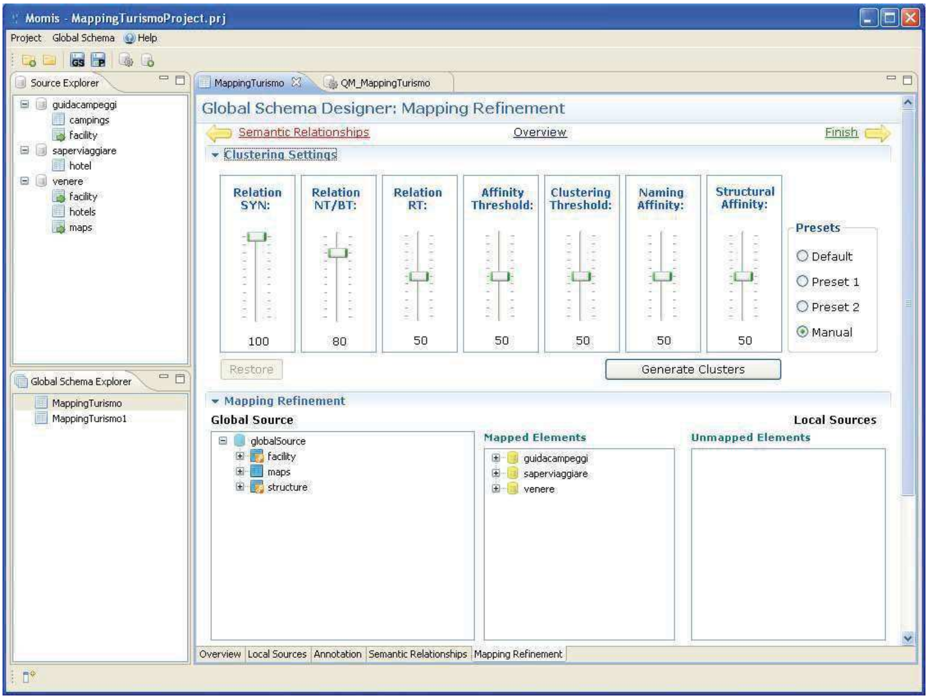
Finally, once obtained the desired integration result, a user can pose queries on the GS by using the *Query Manager* tool (see Figure 2). As MOMIS follows a GAV approach, the query processing is performed by means of query unfolding [3]. The query unfolding process generates for each global query (i.e. a query on the GS) a Query Plan composed by a set of queries:

- a set of local queries that have to be executed on the local sources simultaneously by means of wrapper,
- a mapping query for merging the partial results (defined by means of the join function),
- a final query to apply the resolution functions and residual clauses.

In the Open Source version of MOMIS, we implemented *the Query Manager Web Service* which allows to integrate MOMIS with other applications (e.g. Business Intelligence solutions). Moreover, a user-friendly *Web Application* (see Figure 2) has been implemented to guide an end-user, without experience on data integration solutions, to easily compose and execute queries on the integrated schema.

3 Features

The MOMIS system has been re-engineered and extended by DataRiver with a set of features and components to address several important data integration challenges and speed up data integration projects:



Global Schema Designer: Mapping Refinement

Relation SYN: 100

Relation NT/BT: 80

Relation RT: 50

Affinity Threshold: 50

Clustering Threshold: 50

Naming Affinity: 50

Structural Affinity: 50

Presets: ○ Default ○ Preset 1 ○ Preset 2 ● Manual

Fig. 3. The MOMIS GUI.

Multiple Global Schemas. Within the MOMIS system, each project can be composed by several alternative Global Schemas representing different views of the set of the underlying data sources. The creation of a new project is performed by following few steps: first of all, the designer creates a new project, then uploads the local sources and starts the creation of a new GS by editing each section that composes the integration process; once completed the GS the designer can:

- pose queries on the created GS
- upload other sources
- create a new GS on the local sources (or a subset of the local sources)

The Global Schemas can be easily imported/exported from a Data Integration project to other projects.

User-friendly and flexible GUI. In order to guide the designer through the integration process, a very intuitive, flexible and user-friendly interface has been designed. MOMIS has been developed as an Eclipse Rich Client Platform[7] (RCP) application that allows developers to use the IBM's open source popular

[7] http://wiki.eclipse.org/Rich_Client_Platform

Eclipse platform[8] to create flexible and extensible desktop applications. All system components are built as plug-ins of the Eclipse development environment, which supports also an easy incorporation of new tools. As shown in Figure 3, the MOMIS GUI is divided in three main sections: Source Explorer, Global Schema Explorer and Global Schema Designer.

Annotation Suite. The annotation phase is one of the most critical and expensive step because it deeply affects the subsequent phases.

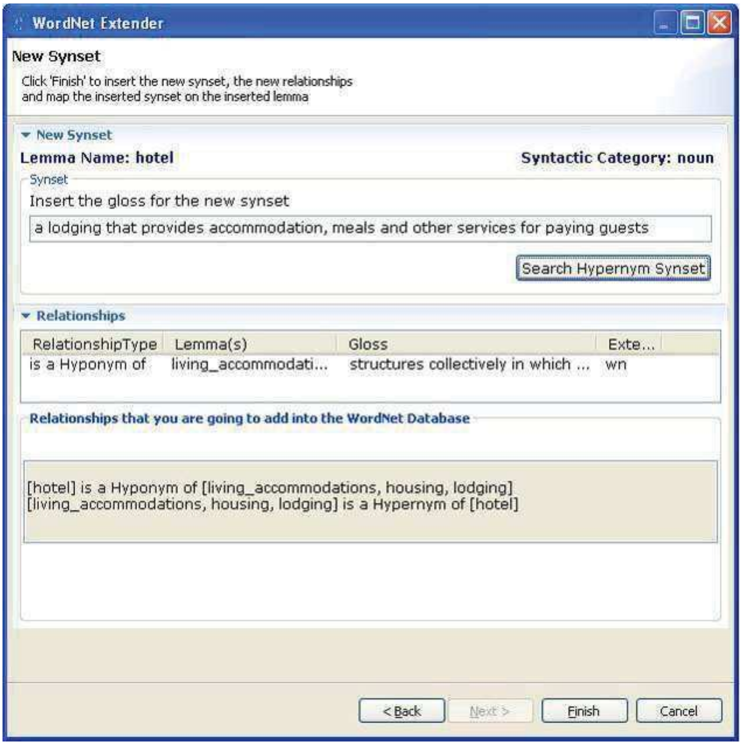
Usually integration projects involve large data sources, with hundreds of tables and attributes, coming from a particular domain of interest (e.g. medical, biological, tourism). The manual annotation of each data source element is a time consuming and potentially boring work that can lead to omissions and errors. Moreover, the semantics of local schemas could not be represented in the lexical resource, in such cases the designer is unable to select the exact meaning for a term, thus generating missing or inaccurate annotations.

Therefore, a set of tools have been developed in order to optimize the annotation phase and help the designer during the extension of the lexical reference:

- **Annotation Importer**: the reuse of previous annotations is an important feature. For this reason, a tool for easily importing source annotations from a GS to another GS has been developed.
- **WordNet Extender**: the WordNet Extender [1] tool enables the extension of the lexical reference with domain glossaries. An intuitive GUI (see Figure 4), for the extension of the lexical reference, guides the designer to perform step-by-step operations such as providing new terms (lemmas), writing definitions for new concepts (glosses) and building relationships between the added concepts and the pre-existing ones. In order to optimize the annotation phase and increase the annotation accuracy, we implemented an automatic annotation algorithm which includes stemming and stop words removal functionalities.
- **Automatic Annotation**: The main advantage of automatic annotation is simply speed: wholly or partially automated methods facilitate the annotation of large sets of classes. If the lexical reference has been extended, the automatic annotation algorithm associates to each data source element the more recent meaning of the domain glossary, else, it associates to the data source element the first meaning present in WordNet[9].
- **Hypernym Graph Viewer**: to help the designer to build sound relationships between the added synsets and the pre-existing ones, we implemented the Hypernym Graph Viewer tool (see Figure 5). A hypernym relationship is a WordNet semantic relationship that connects two synsets where the first generalizes the second (e.g. *animal* is a hypernym of *dog*); the opposite of hypernym is the hyponym relationship (e.g. *student* is a hyponym of *person*). The hypernym relationships chain of a specific lemma (more precisely,

[8] http://www.eclipse.org/

[9] Synsets in WordNet are ranked in the order of their utilization frequency.



WordNet Extender

New Synset

Click 'Finish' to insert the new synset, the new relationships and map the inserted synset on the inserted lemma

New Synset

Lemma Name: hotel

Syntactic Category: noun

Synset

Insert the gloss for the new synset

a lodging that provides accommodation, meals and other services for paying guests

Relationships

RelationshipType	Lemma(s)	Gloss	Exte...
is a Hyponym of	living_accommodati...	structures collectively in which ...	wn

Relationships that you are going to add into the WordNet Database

[hotel] is a Hyponym of [living_accommodations, housing, lodging]
[living_accommodations, housing, lodging] is a Hypernym of [hotel]

Fig. 4. Wordnet Extender.

of the set of synsets associated to a lemma) or of a specific synset is shown by an interactive graph. The designer can navigate the graph by focusing on a specific synset to view only the branch of its hypernyms, or by using the keyword search.

Figure 4 and 5 show an example; a designer has to integrate data coming from tourism data sources, in particular data that refer to hotels and camps. Let us suppose the designer does not find satisfactory the meaning associated to hotel (i.e.“*a building where travelers can pay for lodging and meals and other services*). From the Hypernym Graph Viewer the designer can notice that the synset associated to hotel is a hyponym of the synset “*a structure that has a roof and walls*”. The designer creates a new synset for the lemma *hotel* by introducing the gloss “*a lodging that provides accommodation, meals and other services for paying guests*”. Then, the designer links the new synset of hotel with the synset associated to the lemma *living_accommodations* and defines a new hypernym relationship.

– **Lexical shared Repository**: once the lexical reference has been extended, the designer can export the domain glossary and reuse it in other projects. Moreover, the glossary can be shared by different designers at the same time. WordNet is distributed as a set of data files. The WordNet internal organi-

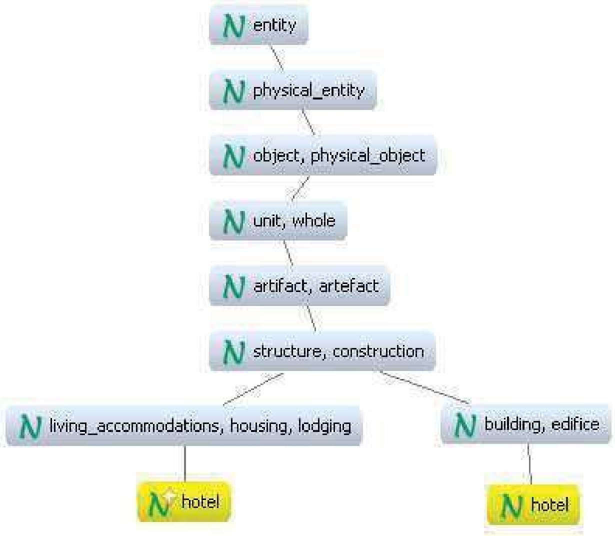


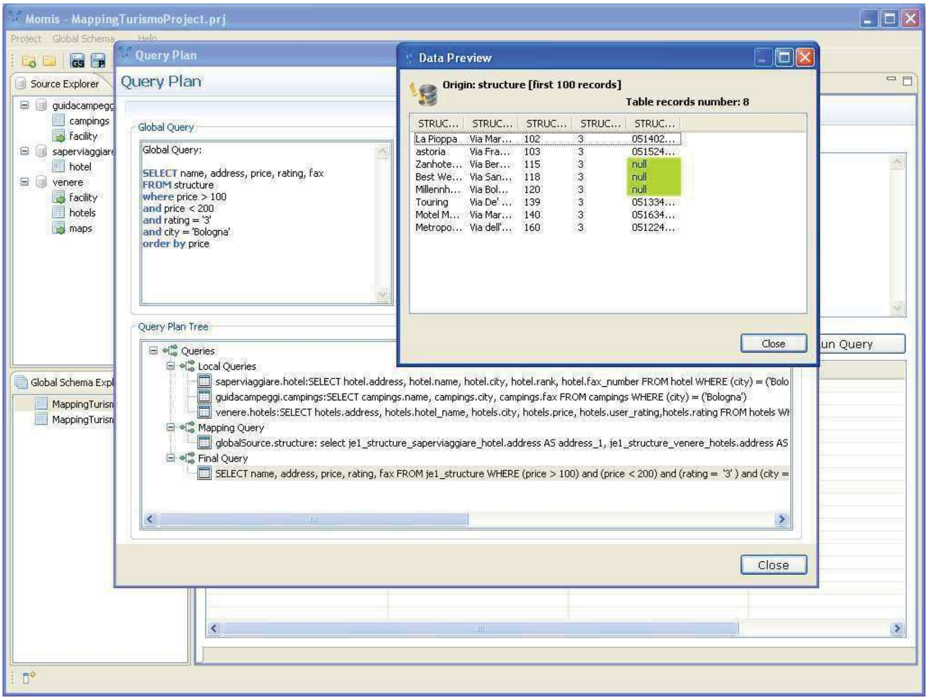
Fig. 5. Hypernym Graph.

zation has been extrapolated and all the terms, definitions and relationships are stored in an embedded relational database. We have chosen the HyperSQL[10] DBMS, a lightweight and Open Source DBMS written in Java. The HyperSQL WordNet database has been embedded and distributed within the MOMIS system, so that no configuration is required. If an extended lexical reference has to be shared by an organization, a shared repository can be created by using the Open Source MySQL[11] DBMS Server to store data. The MOMIS system can be configured to use this shared lexical reference and so, the designer can exploit already defined domain glossaries.

Data Preview functionalities. A data preview tool is helpful in each phase of the integration process as it provides an instant view of data at any step. By this tool, designers can explore the content of local source attributes, preview the partial integration results and then refine the GS. As described before, the annotation phase is one of the most critical as it deeply affects the subsequent phases. Table and attribute names are often labeled with abbreviations or company codes. The data preview tool may help the designer choosing the right meaning of a term in all these cases where the labels do not represent the instances they contain. The GS, automatically generated by the system, can be refined interactively via a set of editors that help the designer during the definition of *Join*, *Transformation* and *Resolution Functions*. Through the

[10] http://hsqldb.org/

[11] http://www.mysql.com/



```
Global Query:

SELECT name, address, price, rating, fax
FROM structure
where price > 100
and price < 200
and rating = '3'
and city = 'Bologna'
order by price
```

```
Queries
  Local Queries
    saperviaggiare.hotel:SELECT hotel.address, hotel.name, hotel.city, hotel.rank, hotel.fax_number FROM hotel WHERE (city) = ('Bolo
    guidacampeggi.campings:SELECT campings.name, campings.city, campings.fax FROM campings WHERE (city) = ('Bologna')
    venere.hotels:SELECT hotels.address, hotels.hotel_name, hotels.city, hotels.price, hotels.user_rating,hotels.rating FROM hotels WH
  Mapping Query
    globalSource.structure: select je1_structure_saperviaggiare_hotel.address AS address_1, je1_structure_venere_hotels.address AS
  Final Query
    SELECT name, address, price, rating, fax FROM je1_structure WHERE (price > 100) and (price < 200) and (rating = '3' ) and (city =
```

Origin: structure [first 100 records]

Table records number: 8

STRUC...	STRUC...	STRUC...	STRUC...	STRUC...
La Pioppa	Via Mar...	102	3	051402...
astoria	Via Fra...	103	3	051524...
Zanhote...	Via Ber...	115	3	null
Best We...	Via San...	118	3	null
Millennh...	Via Bol...	120	3	null
Touring	Via De' ...	139	3	051334...
Motel M...	Via Mar...	140	3	051634...
Metropo...	Via dell'...	160	3	051224...

Fig. 6. Query Plan and Data Preview.

data preview tool, the designer can explore the content of global classes and attributes. Accordingly to the information obtained from the preview, the designer can define: how to change the proposed mappings; which is the appropriate *Join Function* for each global class; which *Transformation* and *Resolution Function* should be applied to each local and global attribute.

Query Plan Viewer. A relational DBMS gives support to the *Query Manager* (QMDB) for the fusion of partial results that are stored in temporary tables. We have chosen as DBMS for the *Query Manager*, HyperSQL, so the installation of the MOMIS system doesn't need any configuration at all. Through the Query Plan Viewer (see Figure 6), for each executed global query the designer can visualize the set of queries that compose the Query plan, and can make a preview of the data contained in the temporary tables created on the QMDB (see Figure 6). In this way, the query execution process is completely visible to the designer.

4 The MOMIS Toolkit

MOMIS is an Open Source Software released by DataRiver under the GNU General Public License (GPLv2), which permits use, modification and incorporation into Open Source products. We encourage both developers and researchers to download the version 1.1 of the software (from http://www.datariver.it), and to contribute to the future developments of the MOMIS system. The developer documentation is available with the source code. Together with the version 1.1, DataRiver published on the website a detailed user manual and a set of video tutorials to learn quickly how to integrate data sources with the MOMIS system.

MOMIS can serve as an open research platform, providing many useful components that can be extended by developers and researchers. We invite both developers and researchers to discuss specific issues, ideas and design with the DataRiver team[12].

5 Future Work

The Roadmap of the Open Source MOMIS system includes improvements about Provenance, Automatic annotation, Object identification and Collaboration:

1. **Provenance** (or lineage) describes where data came from and how it was derived. It provides valuable information that can be exploited for many purposes, ranging form statistical resumes presented to the end-user, to more complex applications such as data cleaning (identifying and correcting data errors) [4];
2. **Advanced Automatic Annotation** techniques will be included for a faster integration process: combination of several annotation methods, also probabilistic methods [11]; abbreviations and acronyms expanded by using the information provided by the schemata and abbreviation dictionaries; compound nouns (composed of more words) automatically interpreted and annotated on the basis of their constituents [13, 12];
3. **Object Identification** techniques (also known as record linkage or duplicate detection) identifies instantiation of the same object in different sources. The current technique (exact matching) will be extended introducing advanced methods based on similarity measures [7];
4. **Collaboration environment** to enable real-time collaboration between integration designers during each phase of the integration process will be developed. Teams of integration designers will be able to share domain glossaries, annotations, integrated schemas, and whole integration projects, reducing the cost of data integration.

6 Acknowledgment

We would like to thank all DataRiver members and collaborators that took part in the re-engineering of the MOMIS system: Antonio Sala, Andrea Prandini, Silvano Pancaldi, Daniele Miselli, Sara Quattrini, Enrico Calanchi, Matteo Di Gioia, Fabrizio Orlandi.

[12] Please contact the DataRiver team if you are interested in contributing to the MOMIS system at info@datariver.it.

References

1. R. Benassi, S. Bergamaschi, A. Fergnani, and D. Miselli. Extending a lexicon ontology for intelligent information integration. In R. L. de Mántaras and L. Saitta, editors, *ECAI*, pages 278–282. IOS Press, 2004.
2. D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. Synthesizing an Integrated Ontology. *IEEE Internet Computing Journal*, pages 42–51, Sep-Oct 2003.
3. D. Beneventano, S. Bergamaschi, M. Vincini, M. Orsini, and R. C. Nana Mbinkeu. Getting through the thalia benchmark with momis. In *Proceedings of the Third International Workshop on Database Interoperability (InterDB 2007) held in conjunction with the 33rd International Conference on Very Large Data Bases, VLDB 2007, Vienna, Austria, September 24*, 2007.
4. D. Beneventano, A. R. Dannoui, and A. Sala. Data lineage in the MOMIS data fusion system. In *First International Workshop on Managing Data Throughout its Lifecycle (DaLi'2011), in conjunction with ICDE 2011, April 11-16 Hannover*, 2011.
5. S. Bergamaschi, D. Beneventano, F. Guerra, and M. Orsini. Data integration. In D. W. Embley and B. Thalheim, editors, *Handbook of Conceptual Modeling Springer, Berlin, Germany*, pages 443–478, 2011.
6. S. Bergamaschi, S. Castano, M. Vincini, and D. Beneventano. Semantic integration of heterogeneous information sources. *Data Knowl. Eng.*, 36(3):215–249, 2001.
7. M. D. Gioia, D. Beneventano, and M. Scannapieco. Multi-source object identification with constraints. In P. L. Bowen, A. K. Elmagarmid, H. Österle, and K.-U. Sattler, editors, *ICIQ*, pages 266–267. HPI/MIT, 2009.
8. A. Y. Halevy, A. Rajaraman, and J. J. Ordille. Data integration: The teenage years. In U. Dayal, K.-Y. Whang, D. B. Lomet, G. Alonso, G. M. Lohman, M. L. Kersten, S. K. Cha, and Y.-K. Kim, editors, *VLDB*, pages 9–16. ACM, 2006.
9. M. Lenzerini. Data Integration: A Theoretical Perspective. In L. Popa, editor, *PODS*, pages 233–246. ACM, 2002.
10. G. A. Miller, R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. WordNet: An on-line lexical database. *International Journal of Lexicography*, 3:235–244, 1990.
11. L. Po and S. Sorrentino. Automatic generation of probabilistic relationships for improving schema matching. *Information Systems, Special Issue on Semantic Integration of Data, Multimedia, and Services*, 36(2):192–208, 2011.
12. S. Sorrentino, S. Bergamaschi, and M. Gawinecki. NORMS: an automatic tool to perform schema label normalization. In *ICDE 2011, April 11-16, Hannover, Germany*, 2011. Demo paper.
13. S. Sorrentino, S. Bergamaschi, M. Gawinecki, and L. Po. Schema label normalization for improving schema matching. *DKE Journal*, 69(12):1254–1273, 2010.
14. E. T. Ted Friedman, Mark A. Beyer. Magic quadrant for data integration tools. Gartner RAS Core Research Note G00207435, November 2010.

Emerging Applications for Schema Mappings
(Extended Abstract)

Paolo Papotti

Dipartimento di Informatica ed Automazione
Università degli Studi Roma Tre
papotti@dia.uniroma3.it

1 Introduction

There are many different classes of applications that need to exchange, correlate, or integrate data. In light of this, the need of a unifying theoretical framework for all these applications has been advocated by database researchers [4]. In particular, data exchange and schema mappings yield a principled approach to improving data management.

Data exchange addresses the common problem of providing unified and transparent view to a collection of data stored in autonomous and heterogeneous data sources. The unified view is achieved through a target schema, and is realized through some form of *mapping* from the source repositories to a target, materialized database [10]. In contrast, in data integration setting the same objective is achieved through a virtualization mechanism based on querying of the target (mediated) schema [15]. *Mappings*, also called *schema mappings*, are expressions that specify how an instance of the source repository should be translated into an instance of the target repository. In order to be useful in practical applications, they should have an executable implementation – for example, under the form of SQL queries for relational data, or XQuery scripts for XML. This latter feature is a key requirement in order to embed the execution of the mappings in more complex application scenarios, that is, in order to make mapping a plug and play component of integration systems.

Ten years ago, a first generation of schema-mapping tools [18, 21] started to support the process of generating complex logical dependencies (i.e., tuple generating dependencies) based on a user-friendly abstraction of the mapping provided by the users. Once the dependencies are computed, the tools transform them into executable scripts to generate a target solution in a scalable and portable way. Given the effectiveness of such systems in the mapping generation and given the solid theoretical foundations, schema mappings have been largely studied in the research community and transferred, to some extent, into commercial products (e.g., [14]). Nevertheless, despite the solid results both in system and theory fields, the adoption of mapping systems in real-life integration applications, such as ETL workflows or Enterprise Information Integration (EII), has been quite slow. This seems to be due to three main factors.

A first reason is the *quality of the solutions* produced by mapping systems. To make schema mappings a building-block for a larger class of practical applications, in the last two years a *second generation* of tools [23, 17, 16] have emerged to address the problem of handling functional dependencies and the one of generating solutions of optimal quality, i.e., solution formalized in data-exchange theory under the notion of a *core universal solution* [11]. Such features have been introduced while preserving a crucial requirement for portability and scalability of mapping systems: the possibility to express transformation exclusively by means of standard SQL.

A second factor is the recent advancements in the formal characterization of *new applications* for mapping tools. In fact, new theoretical works have widened the settings and the class of problems that can be handled with mapping systems, thus highlighting important applications that have not been completely exploited in systems yet [1, 13, 3, 24].

Finally, until recent times there were no schema mapping tools available as open-source. Re-implementing a mapping system is a non-trivial task and many researchers have been working on such topics without access to the existing tools. Lately, some of these tools are starting to be offered as open-source, thus enabling a larger class of people to work on such technology [22].

This new vitality in the research community opens new challenges that can lead to significant contributions. In fact, results both in the system aspects of the problem and in the theoretical studies of mapping properties suggest new applications, beyond traditional data exchange and data integration tasks. Examples of data management problems that can benefit of the new results are in the data fusion field [5] (using FDs in the target), in ETL settings (going beyond existing results [9] by using composition and mapping reuse), and for the schema evolution problem (exploiting the new results in the inversion of mappings).

In this paper we first expose the basics about data exchange and schema mappings. We then show how recent advances can positively impact three data management problems. The paper concludes with some open problems.

2 Data Exchange and Schema Mappings

Research on schema mappings has provided a lot of building-blocks towards the goal of a natural, effective and efficient paradigm for the data translation problem. For the sake of presentation, we start by introducing the notions of data exchange and we then introduce the basic ideas behind the first generation of tools for schema mappings generation.

Dependencies and Mapping Scenarios Data exchange systems rely on *embedded dependencies* in order to specify mappings. These dependencies are logical formulas of two forms: *tuple-generating dependencies* (tgds) or *equality-generating dependencies* (egds). There are two classes of constraints. *Source-to-target tgds* (s-t tgds), i.e., tgds that use source relations in the premise and target relations in the conclusion, are used to specify which tuples should be present in the target based on the tuples that appear in the source. In an operational interpretation, they state how to "translate" data from the source to the target. Target schemas are also modeled with constraints: *target tgds* are used to specify foreign-key constraints on the target; while *target egds* are used to encode functional dependencies, such as keys. We assume the standard definition of a relational schema and relational instance over *constants* ∪ *labelled nulls* [10]. Labeled nulls are used to "invent" values according to existential variables in tgd conclusions. For example, consider a source database concerning agencies and their fundings, *Agency(name, city)*, *Funds(agency,amount)* that needs to be moved to a target database with two tables, *Company(id,name,symbol)* and *Grant(amount,company)*, with a key constraint on *Company.name*, and a foreign-key constraint from *Grant.company* to *Company.id.* In the data-exchange framework, such dependencies are typically presented with a first-order logic syntax:

Source-to-Target Tgds
m_1. $\forall n, c, a$: $\mathit{Agency}(n, c) \wedge \mathit{Funds}(n, a) \rightarrow \exists I, S$: $\mathit{Company}(I, n, S) \wedge \mathit{Grant}(a, I)$
Target Tgds
t. $\forall a, c$: $\mathit{Grant}(a, c) \rightarrow \exists N, S$: $\mathit{Company}(c, N, S)$
Target Egds
e. $\forall n, n', i, i', s$: $\mathit{Company}(i, n, s) \wedge \mathit{Company}(i', n', s) \rightarrow (i = i') \wedge (n = n')$

A *mapping scenario* (also called *data exchange scenario* or *schema mapping*) is a quadruple $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, where $\mathbf{S}$ is a source schema, $\mathbf{T}$ is a target schema, Σ_{st} is a set of source-to-target tgds, and Σ_t is a set of target dependencies that may contain tgds and egds [11].

Given two disjoint schemas, $\mathbf{S}$ and $\mathbf{T}$, we denote by the pair $\langle \mathbf{S}, \mathbf{T} \rangle$ the schema $\{S_1 \ldots S_n, T_1 \ldots T_m\}$. If I is an instance of $\mathbf{S}$ and J is an instance of $\mathbf{T}$, then the pair $\langle I, J \rangle$ is an instance of $\langle \mathbf{S}, \mathbf{T} \rangle$. A target instance J is a *solution* [11] of $\mathcal{M}$ and a source instance I iff $\langle I, J \rangle \models \Sigma_{st} \cup \Sigma_t$, i.e., I and J together satisfy the dependencies. Given a mapping scenario $\mathcal{M} = (\mathbf{S}, \mathbf{T}, \Sigma_{st}, \Sigma_t)$, a *pre-solution* for $\mathcal{M}$ and a source instance I is a solution over I for scenario $\mathcal{M}_{st} = (\mathbf{S}, \mathbf{T}, \Sigma_{st})$, obtained from $\mathcal{M}$ by removing target constraints. In essence, a pre-solution is a solution for the s-t tgds only, and it does not necessarily enforce the target constraints.

A mapping scenario may have multiple solutions on a given source instance: each tgd only states an inclusion constraint and does not fully determine the content of the target. Among the possible solutions we restrict our attention to *universal* solutions, which only contain information from I and $\Sigma_{st} \cup \Sigma_t$. Universal solutions have a crucial property: they have a *homomorphism* (i.e., a constant-preserving mapping of values) to all the solutions for a data exchange problem. It is common to prefer universal solutions of minimal size, that is, *core universal solutions* [11]. Under a condition of weakly acyclicity of the target tgds, an universal solution for a mapping scenario and a source instance can be computed in polynomial time by resorting to the classical *chase procedure* [10]. A solution generated by the chase is called a *canonical solution*, while chasing only the s-t tgds in our example scenario generates the canonical, *universal pre-solution.* After a canonical pre-solution has been generated by chasing the s-t tgds, to generate an actual universal solution it is necessary to chase the target dependencies.

In general, it is possible to use chase engines (e.g., [20]) to solve a mapping scenario $\mathcal{M}$ given a source instance I. As an alternative, the chase of a set of s-t tgds on I can be naturally implemented using SQL. Given a tgd $\phi(\overline{x}) \rightarrow \exists\overline{y}(\psi(\overline{x},\overline{y}))$, in order to chase it over I we may see $\phi(\overline{x})$ as a first-order query Q_ϕ with free variables $\overline{x}$ over $\mathbf{S}$. We execute $Q_\phi(I)$ using SQL in order to find all vectors of constants that satisfy the premise and we then insert the appropriate tuple into the target instance to satisfy $\psi(\overline{x},\overline{y})$. Skolem functions [21] are typically used to automatically "generate" some fresh nulls for $\overline{y}$. Note that, in the presence of target tgds and egds, the generation of SQL queries to compute the desired solutions for data exchange, may no longer be possible, in general. We shall discuss later recent research that attacks this problem for various specific cases.

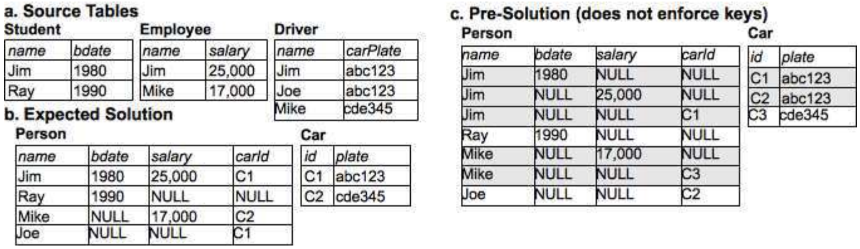
Early Mapping Generation Tools Traditionally, data transformation has been approached as a manual task, where experts had to understand the design of the schemas and write scripts to translate data. As this work is time-consuming and prone to human errors, mapping generation tools have been created to make the process more abstract from the actual scripts, thus easier to handle for a larger class of people. This goal has been pursued by introducing a GUI which allows the users to draw arrows, or *correspondences*, between the schemas in order to define the desired transformation. A correspondence maps atomic elements between a source and a target schema, independently of the underlying data model or of logical design choices, and can be derived automatically with schema matching components.

Correspondences are easy to create and understand, but are a "poor" language to express the full semantics of data transformations. In fact, they do not offer a precise semantics and let the user describe ambiguous transformations, e.g., the same set of arrows can be consistent with different schema mappings. For this reason, a schema mapping tool should be able to interpret the semantics the user wants to express with a set of correspondences. The "intelligence" required to interpret correspondences can be sees as one of the main differences between commercial tools (such as Mapforce and Stylus Studio) and research mapping systems [18, 21, 7]. In fact, the ability to produce mappings that express the desired transformation with a minimal user intervention motivated Clio [18], the first mapping system able to generate schema mappings (expressed as s-t tgds) from a GUI. A following version of the tool [21] introduced support for complex nesting in the schemas (such those common in XML scenarios) and management of unspecified attributes, i.e., required attributes in the target schema for which there is no correspondence to specify their value. The last step in a data exchange scenario is to materialize the target instance, i.e. generate the solution. An important property of mappings is the possibility to execute them with different engines (e.g., SQL, XQuery, XSLT or Java), leaving the choice to the user.

3 Novel Applications

We now briefly discuss examples of novel application for the recent advancements in mapping systems.

Data-fusion The *data-fusion* scenario in Figure 1 requires to merge together data from three different source tables (Figure 1.a.). Based on the correspon-



a. Source Tables

Student

name	bdate
Jim	1980
Ray	1990

Employee

name	salary
Jim	25,000
Mike	17,000

Driver

name	carPlate
Jim	abc123
Joe	abc123
Mike	cde345

b. Expected Solution

Person

name	bdate	salary	carId
Jim	1980	25,000	C1
Ray	1990	NULL	NULL
Mike	NULL	17,000	C2
Joe	NULL	NULL	C1

Car

id	plate
C1	abc123
C2	cde345

c. Pre-Solution (does not enforce keys)

Person

name	bdate	salary	carId
Jim	1980	NULL	NULL
Jim	NULL	25,000	NULL
Jim	NULL	NULL	C1
Ray	1990	NULL	NULL
Mike	NULL	17,000	NULL
Mike	NULL	NULL	C3
Joe	NULL	NULL	C2

Car

id	plate
C1	abc123
C2	abc123
C3	cde345

Fig. 1. Fusing data: source instance (a) and two possible target instances (b)(c).

dences among elements typical of mappings systems, a first-generation system would generate several s-t tgds and a set of target egds that encode the key constraints, as follows (in the following universal quantifiers will be omitted):

$$
\begin{array}{l}
m_1.\ \mathit{Student}(n, bd) \rightarrow \exists Y_1, Y_2 : \mathit{Person}(n, bd, Y_1, Y_2)\\
m_2.\ \mathit{Employee}(n, s) \rightarrow \exists Y_1, Y_2 : \mathit{Person}(n, Y_1, s, Y_2)\\
m_3.\ \mathit{Driver}(n, plate) \rightarrow \exists Y_1, Y_2, Z : (\mathit{Person}(n, Y_1, Y_2, Z) \wedge \mathit{Car}(Z, plate))\\
e_1.\ \mathit{Person}(n, b, s, c) \wedge \mathit{Person}(n, b', s', c') \rightarrow (b = b') \wedge (s = s') \wedge (c = c')\\
e_2.\ \mathit{Car}(i, p) \wedge \mathit{Car}(i', p) \rightarrow (i = i')
\end{array}
$$

However, by using such s-t tgds the best we can achieve is to generate a pre-solution, as shown in Figure 1.c. This solution can be generated efficiently, but it violates the required key constraints and suffers from an unwanted *entity fragmentation* effect: information about the same entities (e.g., *Jim*, *Mike* or the car plate *abc1123*) is spread across several tuples. If we take into account the usual dimensions of data quality [5], it is clear that such an instance must be considered of low quality in terms of compactness (or minimality). Based on these requirements, it is natural to desire the generation of a solution as the one shown in Figure 1.b. Such optimal solution can be materialized by chasing the dependencies above with a post-processing step to minimize the pre-solution. Chase engines are capable of performing this task in very general settings, but in practice there are orders of magnitudes between the execution time needed to compute the pre-solution and the one needed to achieve the optimal one (e.g., seconds vs hours for the same database) [17].

Although it is not always possible, in general, to enforce a set of egds using a first-order language as SQL, there exists a best-effort algorithm that rewrites the above mapping into a new set of s-t tgds that directly generate the target tuples that are produced by chasing the original tgds first and then the egds [16]. As egds merge and remove tuples from the pre-solution, to correctly simulate their effect the algorithm put together different s-t tgds and use negation to avoid the generation of unneeded tuples in the result, for instance by rewriting tgd m_1 as:

$$
\begin{array}{l}
m'_1.\ \mathit{Student}(n_1, b_1) \wedge \neg(\mathit{Employee}(n_2, s_2) \wedge n_1 = n_2) \wedge \neg(\mathit{Employee}(n_2, s_2)\\
\qquad \wedge \mathit{Driver}(n_3, p_3) \wedge n_1 = n_2 \wedge n_2 = n_3) \wedge \neg(\mathit{Driver}(n_3, p_3) \wedge n_1 = n_3)\\
\qquad\qquad \rightarrow \mathit{Person}(n_1, b_1, S_1, C_1)
\end{array}
$$

Moreover, these rewritings are used as a basis for the generation of optimal solutions by relying on the core-computation algorithm developed for scenarios without egds [23, 17]. While the above scenario is already supported by second generation mapping systems, it is interesting to notice that it can be pushed further by considering also the presence of *null values in the sources* [13]. This is a realistic scenario, as in practice a source database may contain labeled nulls as the result of another s-t tgd (e.g., in a chain of schema mappings).

Declarative ETL The most widely used systems in data warehousing environments to express data transformations as a composition of operators in a procedural fashion are known as ETL tools. Operators vary from simple data mappings between tables to more complex manipulations, such as joins, splits of data, and merging of data from different sources. Usually, these tools are used by developers that want to achieve an efficient implementation of a data exchange task. The popularity, in practice, of ETL systems when compared to mapping systems is due to their richer semantics, which allow them to express more operations [9], and to the declarative nature of schema mapping tools that can become a limit with complex transformations where many intermediate steps to manipulate data are needed. For this reason, it is important to study scenarios where flows of mappings, defined using simple intermediate results, are preferable to a single, monolithic mapping with a large number of complex s-t tgds. Recent advancements allow the possibility to manipulate flows of mappings by enabling their composition in sequences over intermediate results [12, 19], the automatic combination of "parallel" mappings [1], and the reuse of transformations defined in similar settings as components [24]. Moreover, the new results discussed above permit to enforce functional dependencies in the target with schema mappings only [16] thus obtaining optimal solution with a clean semantics and execution times comparable to the ones obtained with ETL tools [23, 17].

We can give the intuition of how the expression of data exchange scenarios by mapping tools is preferable to ETL systems in terms of ease of use by comparing a very simple scenario implemented with the two paradigms. To give an idea of the minimal input required by a mapping system, consider that for the ETL scenario in Figure 2 only two lines and two labels are required in a schema mapping GUI to express the same data exchange scenario, as exemplified by the following s-t tgd:

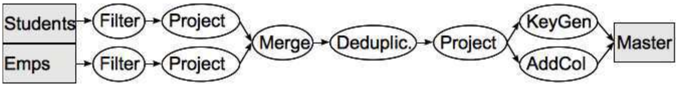


Fig. 2. A simple ETL graph.

$$m_a.\ \mathit{Students}(n_1, b_1, c_1, p_1) \wedge \mathit{Emps}(n_1, d_1, p_1, e_1) \wedge (p_1 = \text{`}\mathtt{Msc}\text{'})$$
$$\rightarrow \mathit{Master}(N_1, b_1, d_1, r_1) \wedge (r_1 = \text{`}\mathtt{M}\text{'})$$

Schema Evolution Although we have presented tools that help the users in the design of schema mappings, their development and refinement can be a time-consuming process in complex integration scenarios. For this reason, when there are changes in the schema, it is desirable for the users to have tools that facilitate the adaptation and reuse of the existing schema mappings. The most promising approach to this *schema evolution* problem is inspired from the model management framework [4], which suggests to automatically adapt the mappings through the use of schema mapping operators. The two most important operators developed to support this process are composition and inversion [6, 12, 19, 13, 3].

For example, assume that we are given the schema mapping m_a above. In order to reuse it when both *Students* in the source and *Master* in the target evolve to a new version, we need to handle their changes by combining m_a with the new mappings m_b, from *Students* to its new version, and m_c, from *Master* to its new version, respectively. In this case, we first need to invert m_b into schema mapping m_b^{-1} and then compose the chain of mappings in the final mapping computed by $m_b^{-1} \circ m_a \circ m_c$.

If we allow the use of more expressive languages, such as SO s-t tgds [12], the composition of two schema mappings always exists and has been implemented successfully in mapping systems [8]. On the contrary, in general there are schema mappings for which inversions that recover all the original data back do not exist. Problems with the lack of *exact inverse* in many common cases have been highlighted and recently researchers have looked to several relaxed notions of invertibility [13, 3]. The most important feature of these notions is their pragmatic approach: when an exact inverse does not exist, they recover the original source data as much as possible. It is interesting to see that such relaxed notions are still useful for data exchange applications, although they have been tested only in restricted settings [8].

4 Open Problems

We believe that there are quite a lot of interesting research opportunities in this area, which we briefly discuss below. A more modular and fluid approach to schema mapping can certainly alleviate the problem of accessing and combining data from multiple sources [4]. Unfortunately, it is still an open problem to identify the unifying schema-mapping language that has good algorithmic properties and is closed under both composition and the various flavors of inverses.

Lot of work is also needed in the field of schema mapping reuse [24], in order to being able to avoid the re-definition of the same transformation in similar scenarios. There are many ways to tackle this problem and contributions are needed from experts in related problems involving semantic properties of the schema (i.e., how to automatically identify previously defined mappings that can be useful with the actual schemas).

Although in data exchange theory optimal solutions has been formalized only for relational settings, the rewriting algorithms are already able to generate also for nested scenarios solutions that present less redundancy than the solutions generated by other mapping systems (following the semantics discussed in [2]). A formal definition of core solutions for nested scenarios is still missing, but it is certainly a needed tool in order to enlarge the use of mappings with optimal solutions in many practical scenarios.

References

1. B. Alexe, M. A. Hernández, L. Popa, and W. C. Tan. Mapmerge: Correlating independent schema mappings. *PVLDB*, 3(1):81–92, 2010.
2. M. Arenas and L. Libkin. XML Data Exchange: Consistency and Query Answering. *J. of the ACM*, 55(2):1–72, 2008.
3. M. Arenas, J. Pérez, J. L. Reutter, and C. Riveros. Inverting schema mappings: Bridging the gap between theory and practice. *PVLDB*, 2(1):1018–1029, 2009.
4. P. A. Bernstein and S. Melnik. Model Management 2.0: Manipulating Richer Mappings. In *SIGMOD*, pages 1–12, 2007.
5. J. Bleiholder and F. Naumann. Data fusion. *ACM Comp. Surv.*, 41(1):1–41, 2008.
6. A. Bohannon, B. C. Pierce, and J. A. Vaughan. Relational lenses: a language for updatable views. In *PODS*, pages 338–347, 2006.
7. A. Bonifati, E. Q. Chang, T. Ho, L. Lakshmanan, R. Pottinger, and Y. Chung. Schema Mapping and Query Translation in Heterogeneous P2P XML Databases. *VLDB J.*, 41(1):231–256, 2010.
8. C. Curino, H. J. Moon, and C. Zaniolo. Graceful database schema evolution: the prism workbench. *PVLDB*, 1(1):761–772, 2008.
9. S. Dessloch, M. A. Hernandez, R. Wisnesky, A. Radwan, and J. Zhou. Orchid: Integrating Schema Mapping and ETL. In *ICDE*, pages 1307–1316, 2008.
10. R. Fagin, P. Kolaitis, R. Miller, and L. Popa. Data Exchange: Semantics and Query Answering. *TCS*, 336(1):89–124, 2005.
11. R. Fagin, P. Kolaitis, and L. Popa. Data Exchange: Getting to the Core. *ACM TODS*, 30(1):174–210, 2005.
12. R. Fagin, P. Kolaitis, L. Popa, and W. Tan. Composing Schema Mappings: Second-Order Dependencies to the Rescue. *ACM TODS*, 30(4):994–1055, 2005.
13. R. Fagin, P. G. Kolaitis, L. Popa, and W. C. Tan. Reverse data exchange: coping with nulls. In *PODS*, pages 23–32, 2009.
14. L. M. Haas, M. A. Hernández, H. Ho, L. Popa, and M. Roth. Clio Grows Up: from Research Prototype to Industrial Tool. In *SIGMOD*, pages 805–810, 2005.
15. M. Lenzerini. Data integration: a Theoretical Perspective. In *PODS*, 2002.
16. B. Marnette, G. Mecca, and P. Papotti. Scalable data exchange with functional dependencies. *PVLDB*, 3(1), 2010.
17. G. Mecca, P. Papotti, and S. Raunich. Core Schema Mappings. In *SIGMOD*, pages 655–668, 2009.
18. R. J. Miller, L. M. Haas, and M. A. Hernandez. Schema Mapping as Query Discovery. In *VLDB*, pages 77–99, 2000.
19. A. Nash, P. A. Bernstein, and S. Melnik. Composition of mappings given by embedded dependencies. *ACM Trans. Database Syst.*, 32(1):4, 2007.
20. R. Pichler and V. Savenkov. DEMo: Data Exchange Modeling Tool. *PVLDB*, 2(2):1606–1609, 2009.
21. L. Popa, Y. Velegrakis, R. J. Miller, M. A. Hernandez, and R. Fagin. Translating Web Data. In *VLDB*, pages 598–609, 2002.
22. L. Seligman, P. Mork, A. Halevy, K. Smith, M. J. Carey, K. Chen, C. Wolf, J. Madhavan, A. Kannan, and D. Burdick. OpenII: an Open Source Information Integration Toolkit. In *SIGMOD*, pages 1057–1060, 2010.
23. B. ten Cate, L. Chiticariu, P. Kolaitis, and W. C. Tan. Laconic Schema Mappings: Computing Core Universal Solutions by Means of SQL Queries. *PVLDB*, 2(1):1006–1017, 2009.
24. R. Wisnesky, M. A. Hernández, and L. Popa. Mapping polymorphism. In *ICDT*, pages 196–208, 2010.

Segmentation of Geo-Referenced Queries

Mamoun Abu Helou

Politecnico di Milano, Dipartimento di Elettronica ed Informazione,
V. Ponzio 34/5, 20133 Milano, Italy
abuhelou@elet.polimi.it

Abstract. The last generation of search engines is confronted with complex queries, whose expression goes beyond the capability of simple keywords and requires the systems which understand query sentences, possibly very simple. Among these queries, huge importance is taken by geo-referenced queries, i.e. queries whose understanding requires localizing objects of interest; such queries are dominant in mobile applications, where the user location is the most important parameter, but are also common to many desktop searches. In this paper, we focus on geo-referenced queries and show how natural language analysis can be used to decompose queries into sub-queries and associating them suitable real-world objects. In this paper we propose a syntactic and semantic approach, which uses syntactic query segmentation techniques and the ontological notion of geographic concepts to produce good query interpretations; an analysis of the method shows its practical viability.

Keywords: Query Segmentation, Query Understanding, Geo-Referenced Query, Multi-Domain Query.

1 Introduction

Search engines perform poorly on complex queries [3]. When a query involves multiple domains and their interconnections, i.e., queries over multiple semantic fields of interest, search engines fail in understanding the query's meaning, also because they try to use all the query information in order to locate one page containing all the results. In this paper, we propose an approach to complex query understanding which focuses on the sub-problem of query segmentation. Such step is essential for decomposing a complex query into sub-queries, and then answering each sub-query independently, as supported on the Search Computing [2]. For example a query such as *"I want a vegetarian restaurant near Eiffel tower and a comfortable hotel"* requires the user to manually extract and combine the answers from various queries, and this is an intricate and tedious job. The purpose of this paper is to understand how the above query can be decomposed so as to present each query as a distinct interaction. Such query decomposition can help to automatically route the query to a corresponding vertical search engine or a Web service.

However, understanding a natural language query requires the application of syntactical, semantic and conceptual knowledge to resolve the ambiguity that abounds in natural language. The output desired from a query understanding process must include the objects, properties of objects and relationships among the query objects.

In this paper, we focus upon geo-referenced queries, e.g. queries which ask about properties of objects which are placed at specific positions. These queries are very much used in practice, and are the majority of queries which are asked from mobile devices.

This paper is structured as follows. Section 2 presents the related work. Section 3 describes the design and implementation. Section 4 presents the experimental results. Finally, we conclude and present some plans for future development.

2 Related Work

Research on *Query Segmentation (QS)* is focused on how to decompose a query into sub-queries; this problem has been tackled in different ways. In [4], authors have shown that query segmentation has a positive impact on the retrieval performance. The segmentation process takes a user's query and automatically tries to separate the query words into segments so that each segment maps to a semantic component. Ideally, each segment should map to exactly one "concept" [7].

One of the earliest approaches to Web *QS* is presented in [6]. They segment queries by computing so-called "connexity scores", that measure mutual information (*MI*) within a segment and the segment frequency in a query log. However, according to [7], the *MI* method cannot capture correlations among more than two words. Recent work in *QS* [1] used a supervised learning method. However, this approach, as all supervised learning methods, requires a significant number of labeled training samples and well designed features to achieve good performance. This makes it hard to adapt in real applications. As an alternative, [7] suggests unsupervised method based on expectation maximization. This approach, as it happens with most unsupervised learning methods, heavily relies on corpus statistics. In some cases, highly frequent patterns with incomplete semantic meaning may be produced.

All These segmentation algorithms take into account the sequential ordering of words, and do not study non-adjacent terms, therefore these approaches just deal with keyword-based short queries but hardly adapt to long natural language queries, including complex queries. Also, they do not try to identify named entities or to assign class labels.

3 Geo-SeCo Framework

3.1 Framework Overview

The proposed approach consists mainly of two parts: *i)* concept extraction, which helps identifying the geo spatial *concepts*; and *ii)* relation identification, which helps discovering relationships among the identified entities. The *concepts* extracted in the first part represent the kernel nodes for the sub-queries. The relationship identification phase combines the syntactical representation of the query with several heuristic rules to sustain the query context by relating the aforementioned extracted parts.

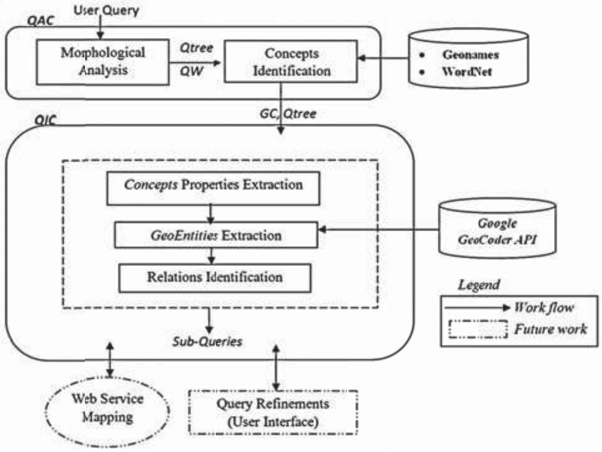


Fig. 1. Overview of the *Geo-SeCo* framework

We have addressed the problem by defining a framework called *Geo-SeCo*. The framework as illustrated in Figure 1 consists of two main components: Q*uery Analysis Component (QAC)*, where the key *concepts/"Web objects",* that is gaining popularity as a new way to reinterpret concept organization in the Web and go beyond the unstructured organization of Web page, from the user's query will be extracted; and Q*uery Interpretation Component (QIC)*, where the possible *object* properties and entities are identified, and also the relationships among the extracted *objects* are discovered.

To preserve the original information conveyed by the query and maintain the query context, we make use of the syntax tree. We use Stanford parser [5] for this purpose; Figure 2 illustrates the syntax tree of the example query (*Qtree*). The syntax tree is an (ordered, rooted) tree that represents the syntactic structure of a string according to some formal grammar.

In a parse tree, each node is either a *root* node, a *branch* node, or a *leaf* node. In Figure 2 the *ROOT* is a root node. *NP* , *VP*, and *PP* are branch nodes, while the sentence words {*i, ... , hotel,}* are the leaf nodes. A node can also be referred to as parent node or a child node. A *parent* node is one that has at least one other node linked by a branch under it. In the example, the upper *NP* is a parent and join node of both NP and PP. A *child* node is one that has at least one node directly above it to which it is linked by a branch of the tree. The phrase *head* is the first preterminal node in the phrase, a *preterminal* node is defined to be a node with one child which is itself a leaf. In the example above the *IN* node which is the proposition identifier for the *PP* is the *head* of this phrase. The following conventional part of speech tag abbreviations have been used through this paper: *NP* (e.g. *a vegetarian restaurant*), *VP* (e.g. *want a ... hotel*), and *PP*(e.g. *near Eiffel tower…hotel*); for noun, verb, and proposition phrases respectively. *JJ (*e.g. *vegetarian)* used for adjectives. *NNP* (e.g. *Eiffel*) *and NN* (e.g. *tower*) used for plural and singular nouns respectively. And *IN* (*e.g. near*) used as proposition. Throw the following sections we will refer to Figure 2 as an example which demonstrates *Geo-SeCo*.

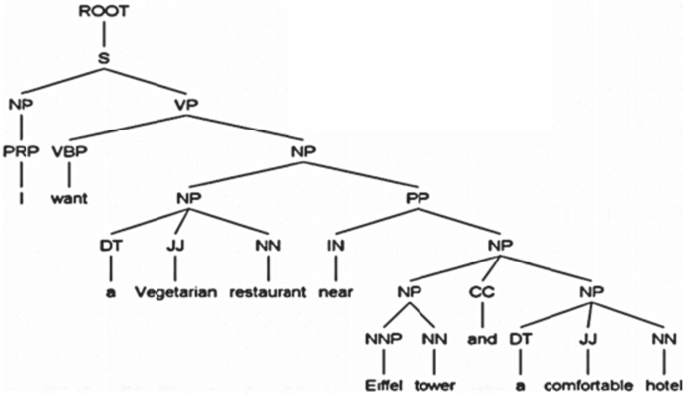


Fig. 2. The syntactic tree of the example query

3.2 Query Analysis Component (*QAC*)

The *QAC* focuses on the identification of the key *concepts* in the query. *QAC* decomposes the user query into *concepts*, where each *concept* represents one search objective in a specific domain. The *concept* can either be a *simple concept* which consists of one word (e.g. hotel), or a *complex concept* consisting of multiple words (e.g. train station). *QAC* employs two steps.

***Step 1**: Morphological Analysis,* on the user's query; the full syntactic parse tree (*Qtree)* is produced, and also running a tokenizer, part-of-speech tagger using the Stanford parser.

***Step 2**: Concept Identification,* which identifies the key *concepts* "*geo concepts*" (*GC*) in the query, a *concept* is a *GC* if it is mapped to one of the *geonames*[1] features, or one of its synonyms or hyperonym synsets using WordNet[2]. Knowing that the *GCs* are nouns, only the nouns among the *QW* are examined. Running the *QAC* over the example query the *GC* list is :{ *restaurant, tower, hotel*}.

3.3 Query Interpretation Component (*QIC*)

QIC tries to extract any possible geo spatial information "*geoEntities" (*i.e. address, or geo spatial entity name). It also extracts the *concept* properties which could be seen as service invocation parameters and filtering criterion. Besides that, it identifies the relation among the extracted parts based on *Qtree*. *QIC* performs the following steps.

***Step 1**: Concept properties (concept to property "cp" relation) extraction.* The adjectives and nouns associated with the *concepts* represent either a property of the entity or a more specific type for that entity than the type expressed by the *concepts*

[1] http://www.geonames.org/export/codes.html

[2] http://wordnet.princeton.edu/wordnet

itself (e.g. *Eiffel* tower, *comfortable* hotel). For that, the adjectives and entities name are extracted based on the *NP* which has a *concept*. The *concept's* consecutive nouns are considered as entity name, while the adjectives are filtering criteria. Running step1 over the example query the name property "*Eiffel*" and the adjectives "*vegetarian* and *comfortable*" were extracted for the concepts *"tower, restaurant*, and *hotel*" respectively.

***Step 2**: GeoEntity extraction.* The *geoEntity* (*GE*), i.e. the entity with a permanent physical location on Earth can be described by geographical coordinates, consisting of latitude and longitude. Any geo-spatial entity/*concept* identified in the query should be associated with at least one component of the *address* field (street name, zip/postal code, city, country). The query may also contain the name for a *geo concept* (name of the geo-spatial entity), which would be extracted in (step1) e.g. *Eiffel tower*.

First candidate *GEs* are collected, we process *Qtree* for this purpose, and assume such entities are the *NPs* which are a child of a *PP*. And also the *GC* will be filtered and checked if any *concept* with its extracted entities name (in step 1) would express a *GE*. Then, the *geo validation* process is performed to confirm these candidate entities and check if they are real world geo spatial entities. The validation process is performed via *Google GeoCoder API*[3]*,* or any similar APIs; the test succeeds if we are able to retrieve the address components of the candidate entity. Later, the extracted entities are processed to identify the address components by finding the best match based on the edit distance between the name of the candidate entity and the *GeoCoder* results components. Running step 2 over the example query, *"Eiffel tower"* was recognized and its address components were extracted.

***Step 3**: Relation identification.* To achieve the best possible query interpretation, we retrieve and analyze the potential relations between the identified *concepts* and entities. These relations are very important as they add descriptions to the *concepts*. For instance, the user might look for a hotel close to a restaurant in a particular city, which is different than just extracting the hotels and the restaurants in that city. To resolve these relations the following steps are employed.

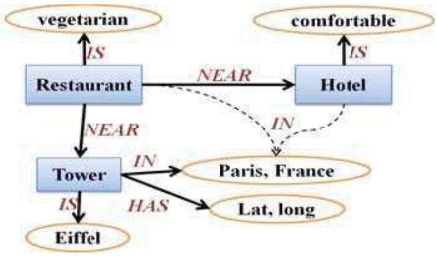
1. *Concept to concept "cc" relation extraction*. Connecting the *concepts* is handled in two ways. The First uses the *PP* in *Qtree*; if a parent node of a *concept* node has a *PP* child, and such *PP* has a *concept* child node, then we bind the parent *concept* node with the child one using the internal *PP* head. The second way simply use existing keywords (e.g. near, close to, or similar keywords); if the query has such keywords then the closest concepts node to the keyword node based on the tree edge counting distance are attached together using the keyword, else a conjunctive (default) connection will be used by relating the closest concepts. After this step the following relations among the *GC* have been extracted from the example query:{ "restaurant *near* tower", "restaurant *near* hotel"}.

[3] http://code.google.com/apis/maps/documentation/geocoding

2. *Concept to geoEntity "ce" relation extraction*. This relation is resolved at first by connecting the *concept* with its consecutive *PP geoEntity's* using *PP* head. If a *concept* does not have such consecutive *PP*, accordingly, we connect the *concept* that has the first join node with a *PP* parent node and such *concept* is not a child node of this *PP*, with the extracted address from this *PP*. If any *concept* is still not attached to a *geoEntity*, any two *concepts* have a relation and one of them is not attached to any *geoEntity*, then this *concept* will inherit the connected *concept's geoEntities*. Running this step over the example query, both the *restaurant* and *hotel* were mapped to the *tower* address components.

3. *GeoEntity to geoEntity "ee" relation extraction.* The relation among the geoEntities is performed based on the *GeoCoder* result. For example *"show me a good italian restaurant on el camino in palo alto ?"*, with *GeoCoder* the street "*el camino*", and the city "*palo alto*" were recognized. Moreover, we process the *Qtree* and the PP *head* (if exist) between the identified geoEntities (i.e. "in" for this example) will be used to connect the two entities. hence the street "*el camino*", is located in the city "*palo alto*"

Figure 3 shows the result of running the system over the example query as a directed graph, where the rectangular nodes are the *concepts* in the user's query. The ellipses are the *GEs*, and the concept's properties which will serve as the services invocation parameter. The graph edges are the relations among these nodes that *SeCo* engine [2] will utilize as a filtering and join criteria; each concept node with its attached properties would be recognized as a sub query which will be mapped to a Web service.



Query:
"I want a vegetarian restaurant near Eiffel tower and a comfortable hotel"
Sub-queries*:*
vegetarian **restaurant** (*in* Paris-France)
near
comfortable **hotel** (*in* Paris-France)
AND
vegetarian **restaurant** (*in* Paris-France)
near
Eiffel **tower** *in* Paris-France

Fig. 3. The result of processing the example query

4 Experiments

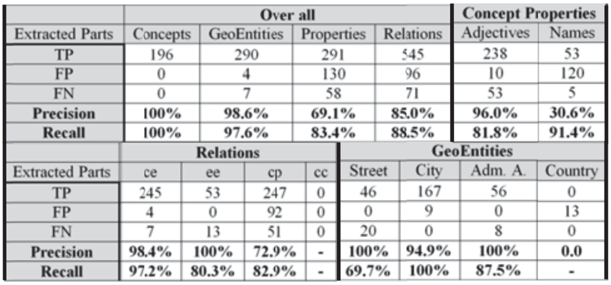
The RestQueries dataset provided by Mooney's group[4] was used in the experiments, consisting of 251 queries about restaurants. Out of the 251 queries 13 were redundant and removed; the rest were manually annotated with the *GC*, *GE* address components

[4] http://www.cs.utexas.edu/users/ml/nldata/restquery.html

(street, city, administrative area, country), and also the relation among the *GC* and *GE* as well as the *GE* relations and the concept properties (adjective, name) relations.

The experiment is designed to measure the capability. of the proposed system to extract the geo spatial concepts "*concepts*", the geo spatial entities components "*geoEntities*", the concepts *properties*, and also the *relations* (*cc*, *ce*, *ee*, and *cp*) among aforementioned parts.

The correctness of the system was measured based on *Recall* and *Precision*. *Recall* is defined as the ratio between the numbers of correctly extracted parts by the system (true positive "TP") to the total number of manually tagged parts in the dataset (TP and false negative "FN" which been miss extracted). *Precision* is the ratio between the numbers of correctly extracted parts (TP) to the total number of the extracted parts using the system (TP and false positive "FP" which been extra/wrongly extracted). Figure 4 reports the results of running the system over the RestQueries. The *Precision* and the *Recall* were recorded for each extracted part. The experiment data and the result are available at ([5]).



	Over all				Concept Properties	
Extracted Parts	Concepts	GeoEntities	Properties	Relations	Adjectives	Names
TP	196	290	291	545	238	53
FP	0	4	130	96	10	120
FN	0	7	58	71	53	5
Precision	**100%**	**98.6%**	**69.1%**	**85.0%**	**96.0%**	**30.6%**
Recall	**100%**	**97.6%**	**83.4%**	**88.5%**	**81.8%**	**91.4%**

	Relations				GeoEntities			
Extracted Parts	ce	ee	cp	cc	Street	City	Adm. A.	Country
TP	245	53	247	0	46	167	56	0
FP	4	0	92	0	0	9	0	13
FN	7	13	51	0	20	0	8	0
Precision	**98.4%**	**100%**	**72.9%**	**-**	**100%**	**94.9%**	**100%**	**0.0**
Recall	**97.2%**	**80.3%**	**82.9%**	**-**	**69.7%**	**100%**	**87.5%**	**-**

Fig. 4. Results of running the RestQueries dataset

The system was able to extract 297 concepts, then 101 out of these concepts have been filtered in (*QI step2)*, for example, "mountain view" is *GE*. Thus 100%, 196 concepts, have been all correctly extracted. 91.4% of the *GE* address components were correctly extracted. The reason behind the incorrect and the missed extraction is due to syntax tree, and also the *geo validation* process. For example, *"what are some good places for ice cream on blanding ave in alameda"*, "blanding ave" was tagged with verb phrase which cause the system to miss such entity. And the *geo validation* process was unable to recognize "fairgrounds dr", in the query *"where is a good american restaurant on fairground dr in sunyvale"*, as a street in the city "sunnyvale". 69.1% of the concepts properties was correctly extracted, again the syntax tree affect the extraction process. For instance, the adjective properties e.g. *italian*, was tagged as nouns, which caused 37 missed adjectives as well as 37 extra entity name. Furthermore, the syntax tree split some *NPs* into *NP* and *VP* which cause the system again to wrongly extract part of this name, which also will be considered as an extra extraction , for instance, the nouns "ice, and ave" were extra extracted and the name "ice cream" and the street "balnding ave" was missed as well as their relations. The overall correct extracted relations were 85%. However, the relation

[5] http://home.dei.polimi.it/abuhelou/data.htm.

extraction was directly affected by the abovementioned extracted parts, the main factor was the property extraction.

An important consideration is that, the queries in this dataset mainly consist of only one geo-spatial concept. As a next step creation of a dataset consisting of more complex queries, having multiple geo-spatial concepts, attributes and relations is anticipated to test the approach more effectively.

5 Conclusions

This paper presents an approach for understanding natural language queries using geo-localizations by splitting long queries into sub-queries and understanding the role of each word in each sentence, specifically by extracting objects with their properties and identifying the geographic relationship between them. The precision and recall of the method are sufficiently high to warrant its use for mobile queries, where geo-references are present in the majority of search queries. Future plans include improving and extending the method, by addressing not only geo-localized queries, but also general compound queries; we aim again at combining the syntactic method for query decomposition to other semantic methods and heuristics, using general-purpose ontological knowledge. In this way, it will be possible to understand if the method “scales” to arbitrary query decomposition, or instead its good performance descends from the extensive use of geo-localizations concepts.

Acknowledgements. This research is part of the Search Computing (SeCo) project, funded by the European Research Council (ERC), under the 2008 Call for "IDEAS Advanced Grants", a program dedicated to the support of frontier research.

References

1. S. Bergsma and Q. I. Wang. Learning noun phrase query segmentation. In EMNLP-CoNLL ’07, pages 819–826, 2007.
2. Ceri, S., Brambilla, M. (Eds.). Search Computing Challenges and Directions. Springer LNCS, Vol. 5950 (2010).
3. Y. Chen and Y.-Q. Zhang. A query substitution – search result refinement approach for long query web searches. In WI-IAT, pages 245-251, 2009.
4. J. Guo, G. Xu, H. Li, and X. Cheng. A unified and discriminative model for query refinement. Proc. ACM SIGIR conference on R&D in IR, pages 379-386, 2008.
5. Klein, D., Manning, C.D.: Fast exact inference with a factored model for natural language parsing. In: Becker, S., Thrun, S., Obermayer, K. (eds.) *Advances in NIPS*. P 3-10. MIT Press (2002).
6. K. M. Risvik, T. Mikolajewski, and P. Boros. Query segmentation for web search. In WWW, 2003.
7. B. Tan and F. Peng. Unsupervised query segmentation using generative language models and Wikipedia. In WWW ’08, pages 347–356. ACM, 2008.

A Scientific Resource Space for Advanced Research Evaluation Scenarios

Cristhian Parra, Muhammad Imran, Daniil Mirylenka,
Florian Daniel, Fabio Casati, and Maurizio Marchese

Department of Information Engineering and Computer Science
University of Trento, Via Sommarive 5, 38123, Trento, Italy
{parra,imran,dmirylenka,daniel,casati,marchese}@disi.unitn.it

Abstract. In this paper, we summarize our experience and first results achieved in the context of advanced research evaluation. Striving for research metrics that effectively allow us to predict real opinions about researchers in a variety of scenarios, we conducted two experiments to understand the respective suitability of common indicators, such as the h-index. We concluded that realistic research evaluation is more complex than assumed by those indicators and, hence, may require the specification of even complex evaluation algorithms. While the reconstruction (or reverse engineering) of those algorithms from publicly available data is one of our research goals, in this paper we show how can we enable users to develop their own algorithms with *Reseval*, our mashup-based research evaluation platform, and how doing so requires dealing with a variety of data management issues that are specific to the domain of research evaluation. Therefore, we also present the main concepts and model of our data access and management solution, the *Scientific Resource Space (SRS)*.

Key words: research evaluation, scientific data access and management, resource space, reputation

1 Introduction

The evaluation of research, i.e the assessment of productivity or measuring and comparing impact, is an important process used to select and promote personnel, assign research grants, measure the results of research projects, and so on. The main instrument the community has been relying on so far is the computation of indicators that are based on bibliographical information, e.g., citations and publication counts, indexes like the h-index [10], etc. Yet, we all know that, for instance, in order to select a new researcher or a new professor to hire we don't just look at the h-indexes of the candidates, rank them in decreasing order, and select the first in the list. In fact, selection processes typically involve more complex decision logics and are, also, partially subjective (e.g., taking into account the opinion of the evaluators). As of today, however, such kind of complex evaluation logics are not supported by existing evaluation tools and, therefore, evaluation still requires significant effort in terms of manual work and interpretation.

Ideally, it should be possible to develop metrics also for those complex evaluation scenarios. In practice, however, doing so turns out to be far from easy, since each evaluation process is different from another and contains a lot of tacit knowledge the evaluators oftentimes are not even aware of themselves. Eliciting this kind of tacit knowledge is one of the first step toward an approach of metrics definition that we call "reverse-engineering". That is, given a set of concrete evaluations or rankings and a set of features describing the researchers involved in the evaluation process (e.g., their h-index, their participation in programme committees, the size of their social network, etc.), what we want to do in the long term is deriving which are the algorithms that allows us to reconstruct the same ranking and, hence, to better predict future assessments.

In the short term (in this paper) – recognizing that more data needs to be collected and analyzed to come up with good results – we however think that it can be already beneficial to allow people to define their own evaluation algorithm, to the best of their knowledge. The approach we follow in doing so is that of implementing a research-evaluation-specific mashup platform, that allows its users to source data from different online data sources, to process, aggregate or filter them, in short, to compose them into a complex evaluation logic.

The advent of the Web has placed much of the necessary resources online and today researchers have access to an overwhelming space of scientific publications thanks to instruments that range from traditional digital libraries (such as SpringerLink or Scopus) to specialized search engines (such as GoogleScholar) and metadata services (such as DBLP). All these sources are, however, heterogeneous and spread over the Web, making it complex to integrate them in a coherent and trustworthy way. There are already tools, such as Harzing's *Publish or Perish* (`http://www.harzing.com/pop.htm`) desktop application for the *Scholar H-Index calculator* (`https://addons.mozilla.org/en-us/firefox/addon/scholar-h-index-calculator/`), that use parts of these data sources and support the computation of simple metrics, which can also be reused and integrated if properly wrapped. Then, today's Web 2.0 enables the early sharing of knowledge through instruments like wikis, blogs, or personal web sites. These kinds of contributions are not peer-reviewed but might still have an impact on the scientific community, depending on the reputation of their authors (think, for instance, of the so-called technology evangelists).

In this paper (i) we report on our first step toward the reverse-engineering of evaluation metrics, i.e., the analysis of how well *existing metrics* may help us reconstruct the outcomes of complex evaluation processes (Section 2); (ii) we propose our idea of *Reseval mashup platform* for the drag-and-drop development of complex metrics (Section 3); and (iii) we describe our approach to deal with the above heterogeneity of sources and functionalities available online, i.e., we describe our *Scientific Resource Space* (Section 4), highlighting the peculiar data management issues that characterize the research evaluation domain. Then we discuss related works and provide an outlook over our future work.

2 On Scientific Reputation and Indicators

Reputation is regarded as the measure of our worth or credibility, according to what others think of us, based on past interactions [17]. Differences among reputation systems are only related to dimensions such as the conceptual model of reference and its granularity, or the sources of information in use [18]. The most basic definition is that reputation is in *the eye of the beholder*, a concept highly based on the experience of the e-commerce and economics domains and that could be different for science or other environments (i.e. social networks like Facebook).

How is this reputation built? Is the reputation of a scientist a good indicator of the excellence of his work?. These and many other questions are becoming relevant as we better realize that single indicators are no longer sufficient to represent scientific impact [2]. Answering them require us to *collect information* (both of reputation and of scientific output), to then analyse them in the search for patterns that shed light on the the nature of reputation phenomena within science. Reseval, powered by the SRS, helped us to cope the challenges of collecting data.

The final goal of this research thread, is to derive the logic of reputation in the mind of researchers, in a way that we can represent it as new metrics for research impact. As a first step towards that goal, we have conducted experiments on the relation between bibliometric indicators and perceived reputation. In the following, we describe these experiments and report on the preliminary results we have obtained from them.

2.1 Experiments

To study whether there is or not a relation between bibliometric indicators and perceived reputation, we needed to find sources of reputation information for a set of researchers and then compute bibliometric indicators for the same set of people. Reseval provide the indicators, but for the reputation information we followed two different approaches: (i) A **survey** asking about research impact and deployed in several conferences of Computer Science. (ii) Crawling results from **research position contests** in Italy and France, produced by selection committees.

Besides Reseval, the study also included some indicators obtained from ReaderMeter (`http://readermeter.org/`) and a parser for Google Scholar search results. Once all the information was available, correlation analysis was performed using Kendall-tau method comparing rankings resulting from reputation ratings and rankings resulting from bibliometric indicators. Only in the case of Italian research contests the analysis was different due to the fact that reputation rankings obtained from this source were only pairs of one selected candidate and one candidate put on a waiting list.

Reputation Survey. The *Liquidpub scientific reputation survey* was designed to be deployed in several conferences with a set of candidates relevant for that conference. Each survey consisted on **a sample of 40 candidates** taken from Jens Palsberg's top h-index researchers list (`http://www.cs.ucla.edu/~palsberg/h-number.html`). Half of the sample was computed according to a measure of affinity to the target conference, based on the distance within co-authorship networks of evaluated researchers with respect to others that published in the same conference. In total, 8 surveys were implemented and deployed in conferences such as *BPM (Business Process Management)*, *ICWE (Web Engineering)* and *VLDB (Very Large Databases)*, getting a total of 77 answers in a period of 3 months of being online (`http://reseval.org/survey/`)

Research Contests. The second approach for getting reputation information consisted of getting the results of contests for research position for Italy and France. In the case of Italy, available data at MIUR site was from 2008 and included, for each contest, the pair of selected candidates where one was the winner of the contests and the other was the second place. For **208 contests pairs** where both candidates had at least one recorded citation, we later calculated in what percentage of the times bibliometric indicators succeeded on predicting the first place of the contest. In the case of France, CNRS data included a list of more than **1000 researchers** participating in differences contests whose result were published in the form of a ranking of 2 or more people.

2.2 Preliminary Results

Reputation Survey. Analysis of reputation ratings in the survey compared to bibliometric indicators showed, for all conferences, a stable pattern of correlation coefficients below the threshold for considering them significant. For this to happen, the correlation coefficient has to be **greater than 0.5** (positive correlation) or **less than -0.5** (negative correlation). Figure 1 shows the value of these coefficients for each metric and source, based on the aggregated results from all the surveys.

Research Contests. Correlation analysis of CNRS rankings shows the same pattern of no-correlation we encountered in the surveys. Figure 2 shows a summary of theses coefficients, all near of zero. This being said, we still need to extend this dataset with the names of researchers that were eliminated on early phases of the selection process of CNRS, organized in three stages.

Table 1. Italian Contest results and bibliometric indicators' performance

	H-Index	Citation count	Cited publications
W < S	47.1% (98)	56.2% (117)	50.5% (105)
W > S	38.9% (81)	39.4% (98)	47.6% (99)
W = S	13.9% (29)	4.33% (9)	1.92% (4)

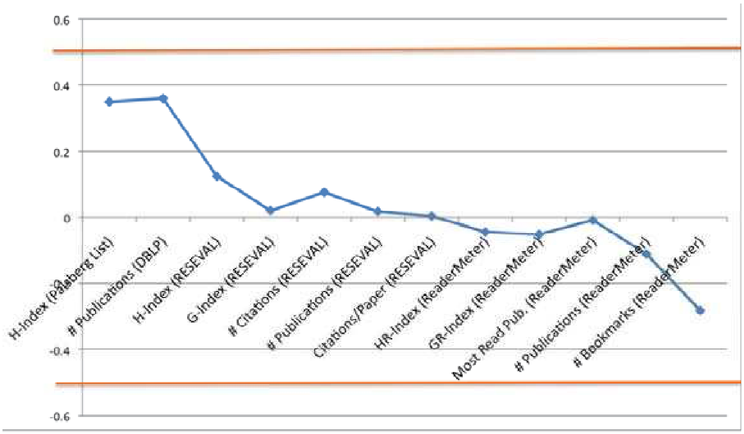


Fig. 1. Correlation coefficients between reputation and bibliometric indicators rankings

In the case of the Italian research contests, Table 1 shows percentage of cases in which the winner has a lower indicator than the second place (W $<$ S), the winner has a better indicator than the second place (W $>$ S) and finally where indicators are the same (W $=$ S). We report only on those indicators that had the better performance, which are the h-index, the total citation count and the number of cited publications. As the table shows, no indicator have a performance better than **50%**.

3 Mashing Up Complex Evaluation Metrics

The experiments above show that research evaluation in practice is much more than just looking at one single indicator or metric. In fact, a good evaluation rather consists in a set of evaluation steps, the application of multiple metrics or data processing functions, comparisons, and similar, a scenario that naturally lends itself to be considered a composition problem. Yet, we are not in the presence of a traditional web service composition, but more in the presence of a *mashup* scenario, in that comparing evaluation results will also require the integration of user interfaces (UIs), e.g., charts and diagrams, in addition to services and data.

With the Reseval project, we aim at providing a mashup platform for research evaluation, taking advantage of both a service-oriented and a data-oriented approach. Reseval[1] is a preliminary research evaluation platform that is currently being developed. An example of a mashup is shown in Figure 3, which describes the evaluation algorithm adopted by the central administration of the University of Trento (UniTN), used for internally distributing resources to departments. In

[1] http://reseval.org/

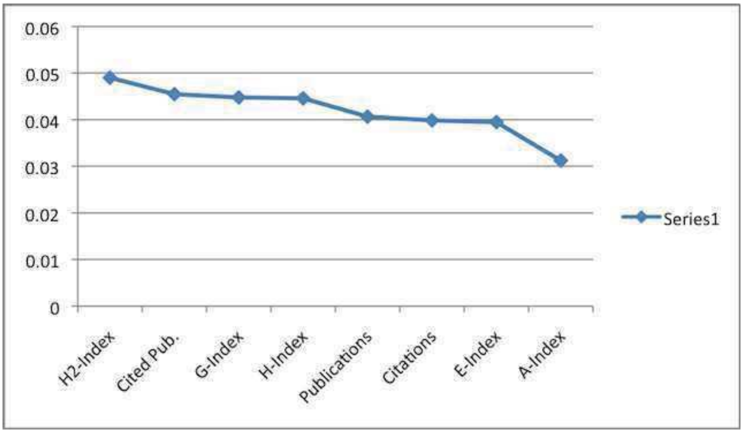


Fig. 2. Correlation coefficients between reputation and bibliometric indicators for Research Contests from CNRS

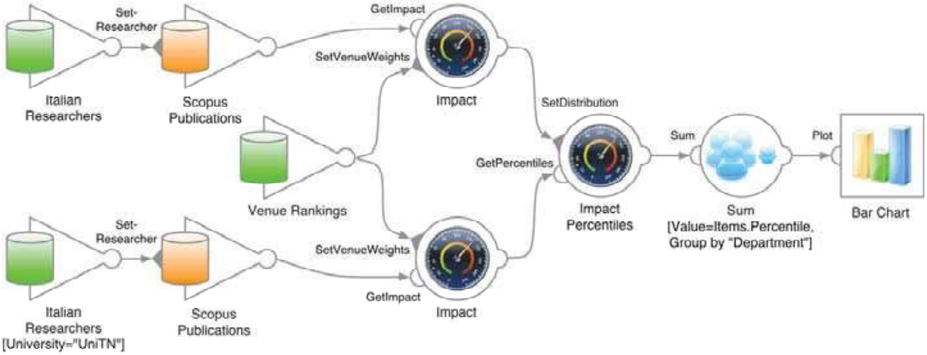


Fig. 3. A complex research evaluation logic mashed up in Reseval

essence, the algorithm computes how well each UniTN researcher performs (in terms of publications) in his disciplinary sector at Italian level, groups performances by department, and plots the final result. The comparison is based on bibliometric indicators.

The requirement we extract from this domain-specific scenario is that we need to empower people involved in the evaluation process (that is, the average faculty member in any academic discipline, as well as the staff member in charge of research evaluation) so that they are able to define and compare relatively complex evaluation processes, taking and processing data in various ways from different sources, visually analyze and understand the results.

Even though there are readily available applications for assessing research impact, all the currently available solutions/tools lack in our view some key features, namely: (1) completeness of data, (2) flexible and personalized metrics, (3) languages to support the user in defining sources, queries, and metrics, and (4) data processing options. Data completeness is indeed a main issue in the process of evaluating research people. In fact, some sources do not (completely) cover some disciplines; for instance, Web of Science is not good for Computer Science, while it is very important to compute citations received by all the documents published by a given author. As we will see, We tackle this issue by leveraging on an open, resource-oriented, Scientific Resource Space System (SRS) that is able to provide homogeneous programmatic access to heterogeneous resources and web services, regardless of how they are implemented, as long as they are web accessible.

We believe that the personalization of the evaluation processes is a key element for the correct use and practical success of the various evaluation indexes. Moreover, people involved in such evaluation process most of the time are not IT experts, capable of building proper software for crawling data sources, automatically parsing relevant information, merging data and computing the needed personalized metrics. Therefore, in order to empower the interested persons an appropriate and possibly easy-to-use IT platform need to be designed, implemented and tested.

Enabling users to develop their own applications or compose simple mashups or queries means simplifying current development practices. Some mashup approaches heavily rely on connections between components (this is the case of Yahoo! Pipes[2] and IBM Damia, for instance), and therefore are inherently imperative; other solutions completely disregard this aspect and only focus on the components and their pre- and post-conditions for automatically matching them, according to a declarative philosophy.

4 The Scientific Resource Space (SRS)

In order to support the composition paradigm proposed by Reseval and to compute quality indicators, we need to access the data we need. To this end, we have built upon the ideas of *dataspaces*, which extend concepts from traditional database management toward heterogeneous data sources [6][9]. The Scientific Resource Space (SRS) follows the intuition that every single piece of knowledge in the vast scientific arena (available online) can be treated as a resource uniquely identified by a URI. In this section, we present our design of such an SRS and report on the current status of its implementation.

4.1 Basic concepts

Managing a space of resources means bringing together inside one homogeneous environment a variety of heterogeneous kinds of resources and providing suitable means to access and use resources and to define and maintain all necessary relationships among the resources. In short, a *resource* can be any artifact we can refer to by a URI and that is accessible over the Web. This notion is very general and captures the requirement of supporting any arbitrary information

[2] http://pipes.yahoo.com/pipes/

such as simple web pages, online documents, web services, feeds, and so on. That is, resources might be simple sources of data or content, but they might also be as complex as SOAP or RESTful web services with their very own interaction logic.

A *resource space* can then be defined as a set of resources and relationships, where the set of resources limits the space to a manageable number of resources, and the relationships express how the resources in the space are interrelated. Theoretically, the biggest resource space with our definition of resource is the Web itself, but, of course, we do not aim at providing a new way of managing the Web. Instead, we think that only by setting suitable boundaries for the resources to be considered, i.e., by limiting the resource space, it is also possible to provide value-adding, novel functionalities that justify the development of a dedicated management system of a SRS.

So far, the concepts are general and prone to be applied on any domain. Our focus, however, is scoped to research evaluation in science. For this, we limit the concept of a resource to include only those artifacts that identify a *scientific resource* such as papers, researchers, journals, conferences, datasets, and so on.

Our implemented SRS has modeled both traditional scientific artifacts (papers, journals, conferences) in and other non-traditional (research blogs, datasets, experiments) in terms of their specific relationships (e.g. co-authorship, citation, less traditional but relevant nowadays' bookmarking, etc.) and possible attributes (e.g. title of a paper, name of an author, volume of a conference, etc.). Regarding the sources of metadata about scientific artefacts, we consider traditional digital libraries and scholarly search engines, as well as social networking services for scientists and even general-purpose social sites.

4.2 High Level Architecture and Implementation

The central component of SRS is the Metadata Warehouse (Figure 4), whose implementation largely follows the traditional ETL (Extract Transform Load) process. The Adapter Layer encapsulates the differences between the data sources. Each adapter is responsible for getting metadata according to the protocols and APIs provided by the source and transforming it into the model of Scientific Resource Space. This task is performed in a few steps. First, scientific metadata is gathered from a source and stored into preliminary tables. The metadata is then loaded into the staging and joined with metadata from other sources. At this stage, metadata elements from each source are preliminary merged based on the identifiers provided by source, ensuring that we introduce no duplicates at the source level. During the cleaning phase the staging area is analyzed to discover (entity matching) and merge entities duplicated across different sources. The algorithms of such matching may vary from simple and intuitive ones, such as comparing titles of the scientific papers, to potentially sophisticated ones like analyzing the co-authorship graphs of scientists. After being cleaned, the metadata is finally loaded into the target database, where it is made available for the applications. This loading is performed by computing and making the changes with respect to the current state of the target database.

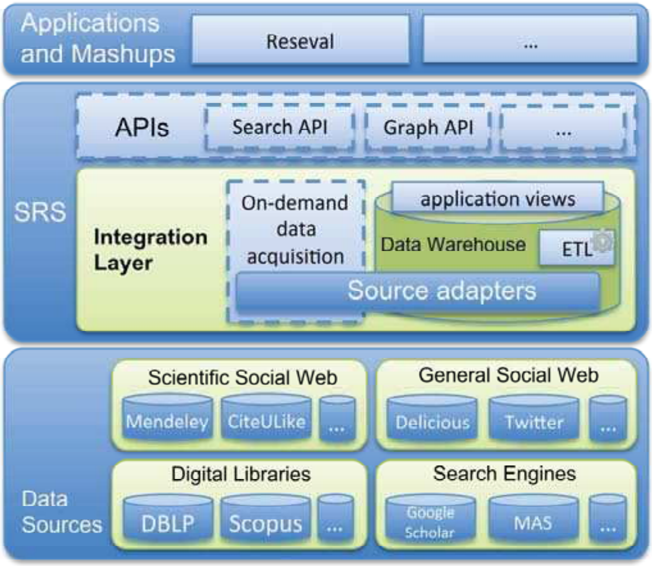


Fig. 4. High-level architecture of Scientific Resource Space (SRS)

The applications built on top of SRS focus on different aspects of scientific resource metadata. In order to provide useful functionality with reasonable performance, they require efficient access to their own presentation of the scientific resource space. For instance, Reseval works with different research metrics including publication and citation-based ones. The numbers of citations and self-citations for papers and authors are the primary units of data for Reseval, and are accessed frequently. For performance reasons, these numbers can not be calculated dynamically and have to be precomputed. SRS addresses this problem by creating the application-specific views which contain all the data needed by the application in a suitable format, and are updated at the final stage of the ETL process. These specific views, as well as general functionality of SRS can later be exposed through the APIs such as search API, or API for navigating the scientific resource space.

In order to enable source-dependent requests, SRS propagates the information about the sources of metadata elements through all the stages of the process to the target database and the application-specific views. At any time for any metadata element it is possible to learn which source or sources it originates from. This property of SRS allows Reseval to compute metrics with respect to any source or combination of them.

Apart from warehousing of metadata, SRS is also looking to support on-demand data acquisition from sources. This approach potentially allows our services to employ sources that expose only search interface, and also to provide personalization in cases when sources rely on user profiles.

5 Related Work

Bibliometrics. Bibliometric indicators have become a standard and popular way to assess research impact in the last few years. All significant indicators heavily rely on publication and citation statistics and other, more sophisticated bibliometric techniques. In particular, the concept of citation [8, 7] became a widely used measure of the impact for scientific publications, although problems with citation analysis as a reliable method of measurement and evaluation have been acknowledged throughout the literature [3]. Indeed, not always a paper is cited because of its merits, but also for some other reasons, as flaws, drawbacks or mistakes. A number of other indices have been proposed to balance the shortcomings of citations count and to "tune" it so that it could reflect the real impact of a research work in a more reliable way. Scientometrics was then introduced as a science for analyzing and measuring quantitatively science itself [4].

In the last decade a number of new metrics were introduced. Although these metrics are also based on citation analysis but they gained popularity over simple citation indexes. For instance h-index [10] was proposed by Jorge Hirsch, as a more comprehensive metric to access the scientific productivity and the scientific impact of an individual researcher. Focusing on a indicator which should indicate quality of a researcher, should consider the performance of top cited paper. Such indicator g-index is proposed by Egghe [5]. To overcome some limitations of both the h-index and the g-index, a new index has been proposed in [1] with the aim to combine the good properties of both indices and to minimize the disadvantages. An index is proposed in [13], is called AR-index, which not only takes into account citations of a researcher but also the publication age. The performance changes in researchers career which comes over the time were ignored previously, thus AR-index can increase or decrease over time.

Reputation and complex metrics. Michèle Lamont's book [16] holds a complete analysis on how evaluation is performed by professors. In the book, she analyses the hard details of peer reviews and 12 panels of experts in the humanities ans social science, extrapolating subjective criteria for decision-making in each different discipline, giving an interesting overview of possible **features** that influence reputation of researchers. The Altmetrics Initiative [11] goes one step further and aims at using social interactions for proposing new metrics of research impact better related to the reputation of researchers.

In the line of analyzing scientific promotion and its relationship with bibliometric indicators, [12][15][14] are some works that show results on how these indicators are related to scientific promotion or how they behave in some particular communities (e.g. Greek Departments of Computer Science). They are related to the experiments we have done (or plan to do) on the approach. However, none of them have tried to compare standard bibliometric indicators with direct reputation, which is the approach we want to use.

Information sources for research evaluation. Until recently researchers had essentially only one source for looking bibliometric type of information: the

Web of Science[3] an on-line commercial database from Thomson Scientific. Starting from the late 90's, many other competitors emerged like Citeseer[4], Scopus[5], Google Scholar[6] and Microsoft Academic[7], with the purpose of giving users a simple way to broadly search the scholarly literature.

Based on the existing sources, new tools are beginning to be available to support people in the research impact analysis. A useful tool is Publish or Perish[8], a desktop based software program that uses only Google Scholar to retrieves the citation data, and then analyzes it to generate the citations based metrics. A different approach is provided by Scholarometer: a social tool which is used in citation analysis and also for evaluation of the impact of an author's publications. It is a browser free add-on for Firefox that provides a smart interface for Google Scholar and requires users to tag their queries with one or more discipline names. Information sources and tools based on these sources are becoming available but they still have many shortcomings. For example they differ in data coverage, data quality. Moreover, these tools are data-source specific and can not be extended to use other data-source. Moreover personalization of metrics is still missing.

6 Conclusion and Future Work

In this work, we have summarized our experience on the research and development of new means for scientific research evaluation, highlighting its requirements in terms of domain-specific data management. As a first step, we approach the problem with *Reseval*, a mashup platform for the composition of complex evaluation metrics. We solve Reseval's data and metadata integration challenges by the means of a dedicated layer, our *Scientific Resource Space*. The resulting integrated infrastructure, has proven to be a powerful instrument also to drive our investigation on the nature of reputation and the reverse-engineering of evaluation metrics we presented in this work.

Yet, as this paper shows, or work is far from being done and our aim is to increase the coverage of our SRS and to improve its integration with other applications that go beyond Reseval. Our work on the reverse-engineering of reputation will benefit from this integration, and we will drive new research threads that are not only in the scope of research evaluation.

We plan to have a demo of the integrated tool ready for demonstration at the time of the conference.

[3] http://scientific.thomson.com/products/wos/
[4] http://citeseer.ist.psu.edu/
[5] http://www.scopus.com/home.url
[6] http://scholar.google.com/
[7] http://academic.research.microsoft.com/
[8] http://www.harzing.com/pop.htm

References

1. S. Alonso, F. Cabrerizo, E. Herrera-Viedma, and F. Herrera. hg-index: A new index to characterize the scientific output of researchers based on the h-and g-indices. *Scientometrics*, 82(2):391–400, 2010.
2. J. Bollen, H. Van de Sompel, A. Hagberg, and R. Chute. A principal component analysis of 39 scientific impact measures. *PloS one*, 4(6):e6022, 2009.
3. A. Chapman. Assessing research: citation count shortcomings. *The Psychologist*, 2:336–44, 1989.
4. D. de Solla Price. *Little science, big science–and beyond.* Columbia University Press New York, 1986.
5. L. Egghe. Theory and practice of the g-index. *Scientometrics*, 69(1):131–152, 2006.
6. M. Franklin, A. Halevy, and D. Maier. From databases to dataspaces: a new abstraction for information management. *ACM Sigmod Record*, 34(4):27–33, 2005.
7. E. Garfield and R. Merton. *Citation indexing: Its theory and application in science, technology, and humanities*, volume 8. Wiley New York, 1979.
8. E. Garfield and A. Welljams-Dorof. Of Nobel class: A citation perspective on high impact research authors. *Theoretical Medicine and Bioethics*, 13(2):117–135, 1992.
9. A. Halevy, M. Franklin, and D. Maier. Principles of dataspace systems. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–9. ACM, 2006.
10. J. E. Hirsch. An index to quantify an individual's scientific research output. *Proceedings of the National Academy of Sciences*, 102(46):16569–16572, 2005.
11. P. G. Jason Priem, Dario Taraborelli and C. Neylon. Alt-metrics: A manifesto. 2010.
12. P. Jensen, J. Rouquier, and Y. Croissant. Testing bibliometric indicators by their prediction of scientists promotions. *Scientometrics*, 78(3):467–479, 2009.
13. B. Jin. The AR-index: complementing the h-index. *ISSI Newsletter*, 3(1):6, 2007.
14. D. Katsaros, V. Matsoukas, and Y. Manolopoulos. Evaluating Greek Departments of Computer Science/Engineering using Bibliometric Indices.
15. J. Kulasegarah and J. Fenton. Comparison of the h index with standard bibliometric indicators to rank influential otolaryngologists in Europe and North America. *European Archives of Oto-Rhino-Laryngology*, 267(3):455–458, 2010.
16. M. Lamont. *How professors think: Inside the curious world of academic judgment.* Harvard Univ Pr, 2009.
17. G. Origgi and J. Simon. On the Epistemic Value of Reputation. In *Paradigms and conceptual systems in knowledge organization: Proceedings of the 11th International ISKO conference, 23–26 February 2010, Rome, Italy*, pages 293–300, 2010.
18. J. Sabater and C. Sierra. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24(1):33–60, 2005.

A System to Support Teaching and Learning Relational Database Query Languages and Query Processing

A. Albano, C. Valisena

University of Pisa, Department of Informatics,
Largo B. Pontecorvo 3, 56127 Pisa, Italy

Abstract. The importance of relational algebra in a database course is widely recognized to facilitate teaching and learning of SQL. From our experience we have also found it very useful for the students to understand the basics of query processing in terms of execution plans. However currently there are no specific tools to make the process of learning relational algebra and execution plans an interesting and stimulating activity. The features of the JRS (Java Relational System) graphical editors of query plans are presented. The graphical editors are used to define and execute queries on a database represented by two kinds of trees: A *logical plan* of relational algebra, and a *physical plan* that describes an algorithm to execute a query using the physical operators of the relational DBMS developed in Java as a teaching tool.

1 Introduction

How to use a relational database system is one of the main topics covered in an undergraduate computer science course, and it is also very common in several other curricula. A database course usually includes conceptual modeling, relational data model and relational database design, relational algebra, SQL, how to build database applications, and the basics of query processing in order to understand how to make physical design decisions.

There is almost universal agreement that knowledge of relational algebra is fundamental to teach and learn SQL as a query language. Relational algebra is based on a small number of operators which take one or two relations as operands to yield a relation as result. A query is just an expression involving these operators. To make a relational algebra expression more readable, it is usually represented as an expression tree of relational algebra operators, called a *logical plan.*

More advanced database courses are system-oriented to deal with data structures to organize tables and indexes, external sorting, physical algebra, transactions and concurrency management, and query optimization techniques to generate physical plans. A *physical plan* is an algorithm to execute a query using different evaluation methods, called *physical operators.* Often the physical operators are particular implementations of the operators of relational algebra.

They differ in their basic strategy and have significantly different costs. However, there are also physical operators for other tasks that do not involve an operator of relational algebra. The result of the evaluation of a physical plan is in general a multiset of records, which is the answer to the query. To make a physical plan more readable, it is usually given as a tree of physical operators.

There are several teaching and learning tools for relational query languages. The focus below is on those that support logical and physical relational algebra.

- ACME is an e-learning framework that supports the automatic correction of problems related to the design of ER diagrams, relational database schemas, normalization, relational algebra and SQL queries. The user defines relational algebra queries using a textual notation [4].
- RAT, Leap RDBMS and Relational are examples of tools for defining and executing textual relational algebra queries [3].
- iDFQL [1] and RALT [2] are examples of interactive systems that enable a logical plan tree to be defined using an interactive graphical interface with a data flow approach. RALT is the only learning tool for relational algebra with the interesting feature of *data lineage* to allow users to track how particular data are derived.

All the above proposals are interesting tools for teaching and learning relational algebra but not physical algebra. Tools such as TOAD (Tool for Oracle Application Developers), go some way in this direction, in fact TOAD shows the Oracle physical plan of an SQL query, how the plan changes by rewriting the query, and a comparison of different query execution alternatives for the same problem. Nevertheless even TOAD-like tools are not really intended specifically for physical algebra.

The proposed interactive JRS environment fills this gap by providing a unique environment with the possibility of practicing with the following different methods in order to formulate relational database queries, and to compare the query results just by graphical interaction using a relational DBMS implemented in Java and designed for educational use which supports a large subset of SQL-92:

- SQL, with the possibility to analyze the query plan produced by the optimizer. The optimizer by default generates left-deep access plans, and uses a greedy optimization technique. A graphical interface allows the user to investigate the effect of other alternatives such as:
  - changing the access plans structure from left-deep to general;
  - changing the optimization level from greedy to limited uniform cost or uniform cost;
  - excluding the use of certain physical operators to generate access plans, and to exploit the impact of certain operators on the cost of a physical plan.
- Relational algebra, with the possibility to define and execute a graphical logical query plan step-by-step, and to see how it can be translated into SQL.
- Physical algebra, with the possibility to define and execute a graphical physical plan step-by-step.

The paper is structured as follows. Section 2, describes the features of two graphical editors for logical and physical plans. In Section 3, we present examples using the graphical editors, and in Section 4, conclusions and future work are presented.

2 Editor Window Areas

Once a database has been selected, the two graphical editors for logical and physical plans are activated with the **Logical Plan** and **Physical Plan** buttons from the main JRS window. They have a similar interface to define the nodes of a tree, the arcs between nodes, and to operate on a tree. The differences are in the types of nodes available and how plans are executed.

When activated, a graphical editor displays a window divided into three main areas (Figure 1):

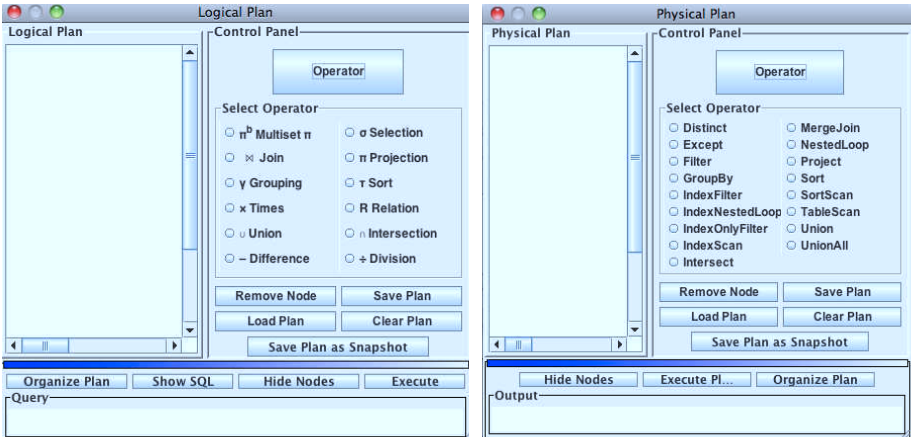


Fig. 1. The Logical and Physical Plan Editors

- **Control Panel**: An area that contains the buttons to add or remove a node, and to save, load or delete a plan.
- **Logical/Physical Plan Area**: An area to define a tree representation of a plan.

  The nodes of a logical plan are the basic operators of the relational algebra (π, σ, γ, $\times$, $\bowtie$, $\cup$, $\cap$, $-$, and $\div$) and, in order to have a resulting a set of records, they require that the relations of the leaves of the tree are defined with at least one key. Instead, the result of the operators π^b and τ is a multiset of records, which are provided to define a logical tree of common SQL queries. The graphical editor allows the use of the operators π^b or τ as the root of an expression tree only. When both are used, the root must be τ.

The nodes of a physical plan are the JRS operators used by the query optimizer to generate access plans.

– **Query/Output**: An area that contains the result of a plan execution, which depends on the type of tree. The area is called **Query** in the case of a logical plan, and **Output** in the case of a physical plane. The area shows the result of evaluating a plan or a subplan rooted in the node selected.
  Above the result area there is a *blue bar*, to resize the area with the mouse, and the following buttons:
  - **Hide/Show Nodes**, to hide the border of the nodes in order to view the plan as plain text.
  - **Execute Plan**, to execute the plan *rooted in the selected node.* This feature enables to analyze the result of a plan step-by-step from the leaves. The result is printed in the output window and it is shown also in a separate window provided by JRS.
  - **Organize Plan**, to redesign the plan tree automatically.
  - **Show SQL**, to see in the **Query** area the translation of a logical plan into SQL which is then used to execute it.

Since the graphical editors are integrated with a relational system, the definition and execution of a plan is simplified by the fact that the graphical interface performs the following task:

1. *Syntax Checking*: The definition of a node is checked for syntax errors.
2. *Semantics Checking*: The definition of a node is checked for semantic errors including type mismatches. It is not possible to use invalid attribute and relation names with the menu provided.
3. *Query Evaluation*: A logical plan is translated into SQL and evaluated using the JRS Relational Engine, while a physical plan is evaluated using the JRS Storage Engine.

3 Examples Using the Graphical Plan Editor

Examples are given using the database schema in Figure 2 and the following query to retrieve the category of products sold singly more than once, and the total quantity sold:

```
SELECT     Category, SUM(Qty) AS TotalQty
FROM       InvoiceLines, Products
WHERE      FkProduct = PkProduct AND Qty = 1
GROUP BY   FkProduct, Category
HAVING     COUNT(*) > 1;
```

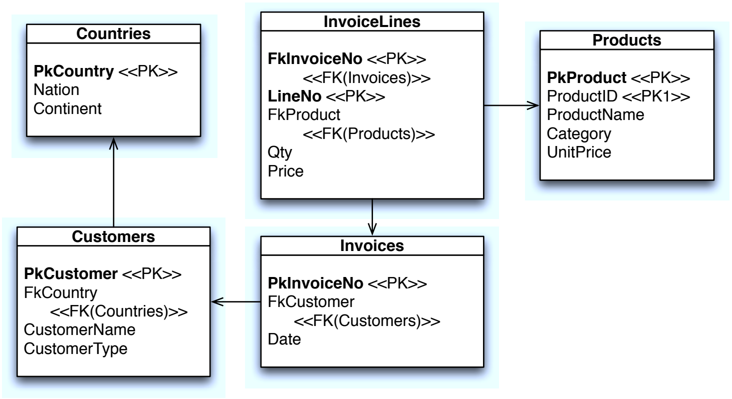
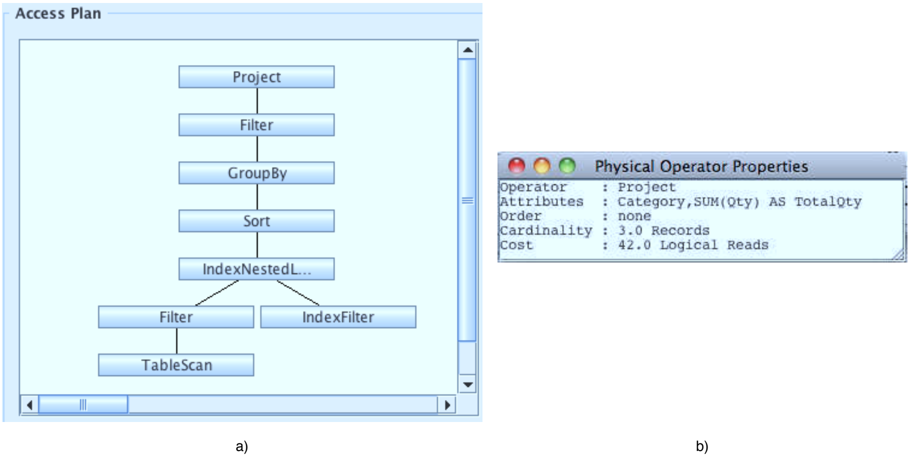


Fig. 2. The Relational Database Schema

Let us first select the option **Show Access Plan**, located in the **Options** menu, and then execute the query. We then get the query result and the *physical query plan* generated by the cost-based JRS query optimizer to execute the query. The physical query plan is represented by an *iterator tree* in a separate window (Figure 3a). The JRS *cost-based query optimizer* estimates the costs of alternative query plans and chooses an efficient final plan. This is done using the metadata available on the database, such as the size of each relation, the number of different values for an attribute, and the existence of certain indexes.



```
Operator     : Project
Attributes   : Category,SUM(Qty) AS TotalQty
Order        : none
Cardinality  : 3.0 Records
Cost         : 42.0 Logical Reads
```

a) b)

Fig. 3. The Optimized Physical Query Plan

Clicking on a node of the plan produces a **Physical Operator Properties** window with information regarding the operator involved, the estimated number of records produced by the operator, and the estimated cost of the operation (Figure 3b).

As usually happens in relational DBMSs, the standard way to evaluate a query with **Group by** is to first retrieve the sorted record required by the operator, and then to execute it in order to produce the final result.

A Logical Plan

Let us define the query using the relational algebra, as shown in Figure 4.

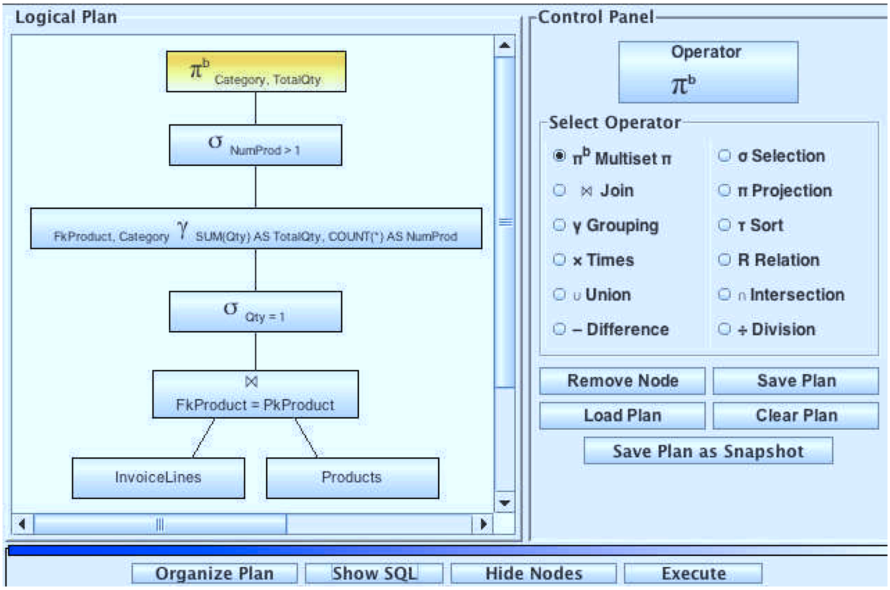


Fig. 4. A Logical Query Plan

Each node of a logical or physical plan has a different contextual menu, which is dynamically created taking into account the attributes of the operands and the schema of the database. Each choice of parameters entails an update of the node label to reflect the choices made. In creating a tree, the node parameters must be specified from the leaves to the root. A contextual menu becomes active, with a right click on the node, but only when the node has all the required operands specified.

By clicking on **Hide Nodes** the logical plan is shown in the traditional form of an algebraic *expression tree* (Figure 5).

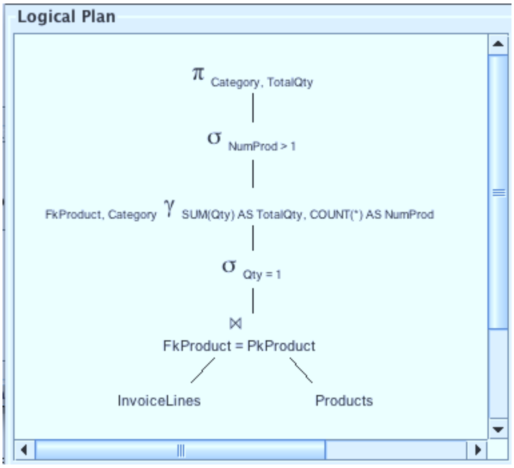


Fig. 5. A Logical Query Plan

Double clicking on a node opens a **Logical Plan Node Information** window with the following information:

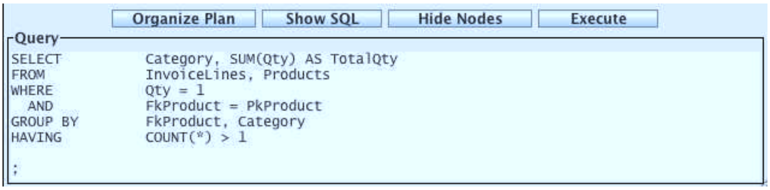
- **Operator**: The node operation.
- **Relation**: The relation name of a leaf node.
- **Condition**: The condition for selection and join operators.
- **Result Type**: The operator result type is a set denoted $\{(A_1\ T_1,\ \ldots,\ A_n\ T_n)\}$, or a multiset, or a sorted set denoted $\{\{(A_1\,T_1, \ldots, A_n\,T_n)\}\}$.
- **Order**: The order of records in the query result.

Any plan node can be selected by clicking. Users can then proceed as follows:

- Clicking on **Execute** produces the query result in the query result window.
- Clicking on **Show SQL** opens a query window that displays the SQL query generated by the Logical Editor, for the subtree of the selected plan node. Figure 6 shows the query generated when the selected node is the plan root, which in this case is the same query used to get the physical plan in Figure 3. The SQL query generated for a logical plan is generally not a single **SELECT**, but a **SELECT** that uses temporary views, because JRS does not allow the use of a subquery in a **FROM** clause. Moreover, in order to avoid the use of views, in the current implementation, the algorithm for generating the SQL query to execute a logical plan does not exploit rewriting rules.

Another Logical Plan

Let us try a different logical plan on the basis of the following result:



```
SELECT        Category, SUM(Qty) AS TotalQty
FROM          InvoiceLines, Products
WHERE         Qty = 1
  AND         FkProduct = PkProduct
GROUP BY      FkProduct, Category
HAVING        COUNT(*) > 1

;
```

Fig. 6. The SQL Query to Execute the Logical Plan in Figure 4

Proposition 1. *Let $\alpha(X)$ be the set of columns in X and $R \underset{C_j}{\bowtie} S$ an equi-join using the primary key p_k of S and the foreign key f_k of R. R has the* invariant grouping *property*

$$ {}_{A}\gamma_{F}(R \underset{C_j}{\bowtie} S) \equiv \pi^{b}_{A\cup F}(({}_{A\cup\alpha(C_j)-\alpha(S)}\gamma_{F}(R)) \underset{C_j}{\bowtie} S) $$

if the following conditions are true:

1. *$A \rightarrow f_k$, with A the grouping columns in $R \underset{C_j}{\bowtie} S$.*
2. *Each aggregate function in F uses only columns from R.*

This property of doing the group-by before a join is called *invariant grouping* since the operator can be brought forward by modifying the grouping attributes only, but the transformation may need an additional projection in order to produce the final result. In general, with a big table R the performance of a join query with grouping and aggregation is improved by doing the group-by before the join.

If there are selections on R, the operator γ is done before a join on the selections on R (Figure 7).

Since the query optimizer does not consider the possibility of doing the group-by before the join, the Logical Editor generates the following SQL code to execute the logical plan:

```
CREATE VIEW LTV1 AS
SELECT      FkProduct, SUM(Qty) AS TotalQty, COUNT(*) AS NumProd
FROM        InvoiceLines
WHERE       Qty = 1
GROUP BY    FkProduct
HAVING      COUNT(*) > 1;

SELECT      Category, TotalQty
FROM        LTV1, Products
WHERE       FkProduct = PkProduct;
```

In this case, the generated query uses a view as a left operand of the join, thus forcing the optimizer to generate two separate plans, one for the view and another for query. However it is interesting to check if the query generated by a logical plan that brings forward the group-by before a join is more efficient.

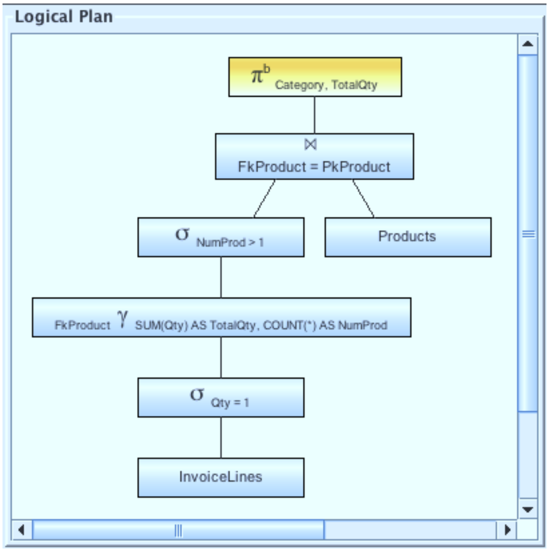

Logical Plan
$\pi^b_{\text{Category, TotalQty}}$
⋈
FkProduct = PkProduct
$\sigma_{\text{NumProd > 1}}$
Products
FkProduct γ SUM(Qty) AS TotalQty, COUNT(*) AS NumProd
$\sigma_{\text{Qty = 1}}$
InvoiceLines

Fig. 7. Another Logical Query Plan

A Physical Plan

Let us now design a physical plan that exploits carrying out the group-by before a join, and uses the operator **IndexNestedLoop** with an index on the **Products** primary key (Figure 8).

Clicking on **Hide Nodes** displays the physical plan in the traditional form of an *iterators tree* (Figure 9).

Double left clicking on a node displays a box with the following information:

- **Operator**: The node operation.
- **Table**: The table name of a leaf node.
- **Index**: The index name, if the operator uses one.
- **Attributes**: The attributes of the index in use.
- **Condition**: The condition for select and join operators.
- **Result Type**: The operator result type, a multiset denoted $\{\!\{(A_1\,T_1, \ldots, A_n\,T_n)\}\!\}$
- **Order**: The order of records in the query result.
- **Cardinality**: The estimated number of records of the plan result.
- **Cost**: The estimated number of pages read from or written to disk to produce the result.

As it happens with a logical plan, any physical plan node can be clicked on in order to execute the subtree with the selected node as root node.

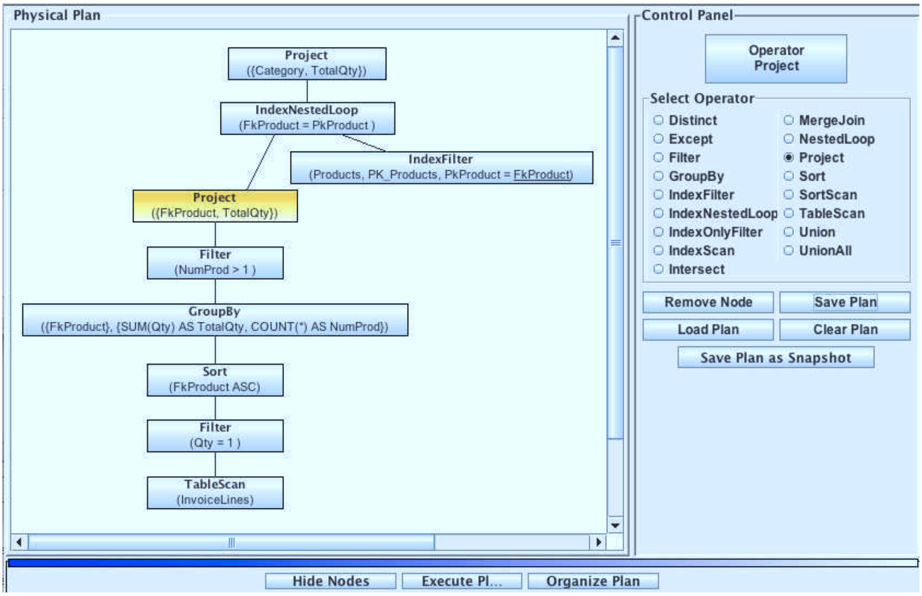


Fig. 8. A Physical Query Plan

4 Conclusions and Future Work

We have presented the JRS relational database system with graphical editors of executable logical and physical plans to support the teaching and learning of query languages and query processing. The main feature of this system is that, unlike other e-learning environments, it has been designed to support both the teacher and student to experiment not only with the SQL language to query a data base and to analyze the query plans generated by the query optimizer, but also to experiment (a) with the execution of a logical plan defined with relational algebra, and (b) with the execution of physical plans defined with the physical operators of the database system. The features of JRS attract student interest but avoid them having to spend their time writing syntactically and semantically correct solutions. Instead the focus is on understanding the concepts of query languages and query processing, and being motivated to experiment with different solutions to solve a problem. In addition, the node information given by the graphical interface is useful for getting the students used to the properties of each operator of a query plan. Thus they learn to give the correct answer to the solution in a written examination that tests the student's ability concerning query processing.

Several system improvements are under consideration. First of all the graphical interface in order to make the interactions simpler. In addition we aim to improve the translation of a logical plan into SQL, to add other join operators, to generalize the relational algebra division operator to express general queries

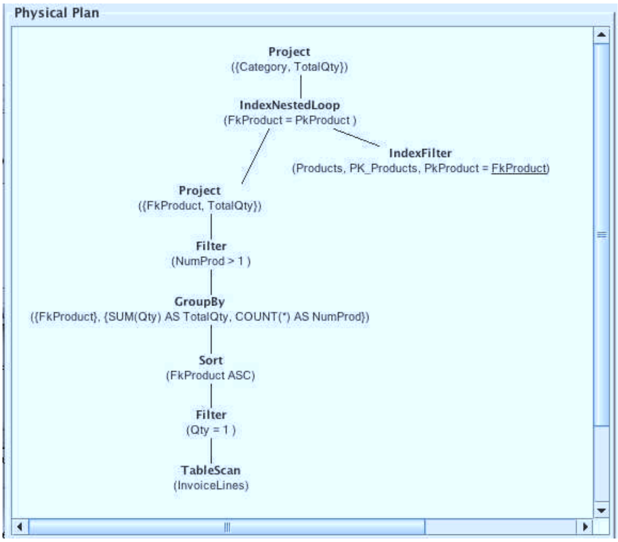


Fig. 9. The Iterators Tree

involving universal quantification, and to provide enhanced functionality and new features on the basis of the most-requested improvements by system users.

Acknowledgments. Thanks to the anonymous referees for their constructive comments.

References

1. Appel A., E. Q. Silva, C. Traina Jr., A. J. M. Traina. iDFQL - A Query-based Tool to Help the Teaching Process of the Relational Algebra. In *World Congress on Engineering and Technology Education, WCETE*, Guararujá, SP, 2004.
2. Mitra P. Relational Algebra Learning Tool. Imperial College, London, 2009.
3. *Relational Algebra*, <en.wikipedia.org/wiki/Relational_algebra>, March 2011.
4. Soler J., I. Boada, F. Prados, J. Poch, R. Fabregat. An Automatic Correction Tool for Relational Algebra Queries. In M., Gervasi and M. Gavrilova (Eds.) *ICCSA 2007*, LNCS, vol. 4706, pp. 861–872, Springer, Berlin Heidelberg, 2007.

Relational Disjunctive Patterns Mining for Discovering Frequent Variants in Process Models

Corrado Loglisci, Michelangelo Ceci, Annalisa Appice, and Donato Malerba

Dipartimento di Informatica, Università degli Studi di Bari
via Orabona, 4 - 70126 Bari - Italy
{loglisci, ceci, appice, malerba}@di.uniba.it

Abstract. The automatic discovery of process models can help to gain insight into various perspectives (e.g., control flow or data perspective) of the process executions traced in an event log. Frequent patterns mining offers a means to build human understandable representations of these process models. This paper describes the application of a multi-relational method of frequent pattern discovery into process mining. Multi-relational data mining is demanded for the variety of activities and actors involved in the process executions traced in an event log which leads to a relational (or structural) representation of the process executions. Peculiarity of this work is in the integration of disjunctive forms into relational patterns discovered from event logs. The introduction of disjunctive forms enables relational patterns to express frequent variants of process models. The effectiveness of using relational patterns with disjunctions to describe process models with variants is assessed on real logs of process executions.

1 Introduction

Workflow management systems are becoming increasingly important in enterprises due to their capabilities of managing activities and actors involved in a business process as well as recording the logs of process executions.

Despite the amount of event logs produced by enterprises, software vendors use this information in order to answer to only simple questions under the assumption that the business process is fixed and known, e.g., the calculation of performance metrics like utilization and flow time. However, in many domains, business processes are evolving and people may have an oversimplified and incorrect view of the actual business processes [10]. In this scenario, process mining techniques play a key role with the extraction of models (or patterns) from event logs where they can provide useful insights in the design of new workflows as well as they permit to collect information exploitable in the workflow optimization.

In the literature, several approaches for mining process models have been proposed. A Markovian approach is described in [2] to investigate the correspondence between the instances of a software engineering process and a model of the process. The process model is represented by a Finite State Machine derived from executions of software development processes and, with the help of distance metrics, it is used to quantitatively measure the discrepancies from new executions. A graph-based perspective is considered in [12] where a variant of Petri nets, called Workflow Nets, has been used to mine and model workflow processes. In these works, a sophisticated algorithm is presented to extract a process model based on binary relations discovered into the workflow logs. Notably, this approach permits to determine which class of workflow models the algorithm is guaranteed to work with. More recently, some authors [5], as we do in this work, have successfully applied pattern discovery techniques in order to identify frequent activities and their relationships. Since high frequency denotes regularity, frequent patterns can provide arguments for process models based on the evidence of regularities in the executions. Indeed, frequent patterns are intended as a means to capture the typical order of execution between activities (control perspective) and, at the same time, they model the possible associations among the properties of the process, activities and actors (data perspective).

The common characteristic of studies reported above is that they discover patterns (or more generally models) that identify the typical order of execution of activities without considering *variants*. However, real-world processes tend to be so complex and less structured that it is difficult to determine patterns to which several process executions comply with [11]. Moreover, the application of patterns that does not consider variants can turn out to be impractical in the workflow management systems which deal with exceptional situations and structural changes during runtime. On the other hand, considering these changes when discovering patterns may lead to the generation of numerous variants which are difficult to maintain even if slightly different one from each other [4, 7].

A deep analysis may reveal that this limitation comes from the fact that traditional frequent pattern discovery algorithms permit to mine conjunctions of the activities present in a set of process executions and does consider disjunctions that can model process variants. This approach poses some limitations to the patterns expressiveness and, in addition, leaves unexplored two potentialities of the pattern discovery: i) discovering interesting patterns when activities are not present in a sufficient number of process executions, and ii) discovering a combination of relationships between activities which are different from the classical conjunctions, such as disjunctions. These two potentialities are not independent each other, since the discovery of patterns including other relationships between activities may lead to discover patterns that otherwise would be discarded. Considering relationships among activities different from the classical conjunction would permit not only to consider variants in the process executions, but also to consider parallel executions of activities.

In this work, we extend our work in [1], where we have investigated the discovery of frequent patters as a means to extract a human interpretable representation of process models. The peculiarity of our previous work is that the frequent pattern discovery is performed in multi-relational data mining in order to take into account the intrinsic relational structure of logs: several activities and/or actors are involved in the same process execution. In particular, the multi-relational approach permits to solve the following problems. First, objects collected in a log belong to different data types (executions, activities and actors) which interact one each other. By resorting to the first-order logic, that is one of the most common relational representation formalism, properties and interaction of executions, activities and actors are modeled by means of first-order logic predicates. Then, reasoning techniques developed in the field of inductive logic programming (ILP) are employed to discover patterns which are relationally defined as conjunctions of atomic formulas built using the data predicates. Second, activities stored in a log are marked with a timestamp which indicates the time of occurrence and implicitly defines a total temporal order over events. Temporal relationships between activities are represented as predicates and these predicates are used to investigate the temporal autocorrelation in the effect of a property of an activity/actor. Third, some user defined domain knowledge (e.g., the definition of the ordering relation between activities) may be available. This knowledge is profitably exploited by inferential mechanisms typical of a theorem prover which are integrated in the relational pattern discovery. However, the main disadvantage of the algorithm presented in [1] is that it cannot deal with process variants that, as stated before, are of fundamental importance.

To cope with this further issue, we propose to derive process models through the discovery of relational patterns with disjunctions. The advantage of integrating disjunctive forms in patterns is two-fold. First, process variants can be identified and represented with the reference model, thus avoiding the explicit maintenance of numerous variants. Second, activities in the patterns can be OR-ed to represent typical $OR - split/OR - join$ in graph-based constructs [12]. The paper is organized as follows. In the next two sections we present related works and background. In Section 3 we present our algorithm to discover disjunctive relational patterns. Experimental results on two real-world databases are commented in Section 4 and conclusions are drawn.

2 Related Work and Background

Recently, the multi-relational or ILP approaches to build business process models from event logs are receiving increasing attention. Goedertier et al. [3] have faced the task of predicting, by means of learned relational classification rules whether, given the state of a process instance, a particular state transition can occur. The representation formalism considered in this work is the Event Calculus, a first-order logic that elegantly captures the time-varying nature of facts. Learning is based on both positive information (possible transitions) and negative information (prohibited transitions). When no negative information is actually available in the logs, it is artificially generated by means of the closed-world assumption.Similarly, Lamma et al. [6] have considered both compliant (positive information) and non compliant (negative information) execution traces and adapt the algorithm ICL to learn constraints among activities expressed as logical formulas. In practice, the main problems of both methods are the reliable provision of negative information and their scalability to huge event logs.

We overcome this issue by learning from positive examples only (as usual in frequent pattern discovery).

Mining variants of a process is not a novel task. Indeed, significant studies have already addressed this task in the literature. Li et al. [7] describe an approach to discover the reference block-structured model which is the best in covering process variants. The model is obtained by first clustering activities in order to form blocks of similar activities, and then merging blocks into larger blocks. The final reference model is thus the model which has minimum distance from the variants, where the distance is measured by the number of change operations at the activity level. The brittleness of this work is the difficulty to create a set of only variants given that, in real event logs, traces of process and variants are stored together. The determination of a set of workflow models, called *disjunctive workflow schema*, is rather the solution proposed by Greco et al [4] to model the relevant variants. A stepwise procedure permits to refine models (workflow schema) created at the previous step, so that the final set is composed of hierarchically organized models. At each step, the executions which support a schema are partitioned into clusters each of which contains executions with the same characteristics. Each cluster thus represents a relevant variant and it is modeled by a refined workflow schema. However, since each variant is mined with a specialized model, the applicability of this method may be compromised in the case a huge number of specialized models is produced.

The background of this work is in [1], where we have used SPADA in order to extract relational frequent patterns from event logs. In SPADA, it is possible to distinguish between reference objects (*ro*) and task-relevant objects (*tro*). The former are data on which patterns are enumerated and contribute to compute the support of a pattern, while the latter contribute to define the former and they can be involved in a pattern. In the logic framework adopted by SPADA, event logs are converted into a deductive database D. Properties and relationships of the process executions (reference objects), activities and actors (task-relevant objects) are represented as ground atoms in the extensional part D_E, while a user defined background knowledge is expressed as a normal logic program which defines the intensional part D_I. An example of ground atoms stored into the extensional database D_E is reported in the followings:
$process(e1)$. $process(e2)$. $activity(e1, a1)$. $activity(e1, a2)$. $activity(e2, a3)$.
$activity(e2, a4)$. $is\ a(a1, workflow)$. $is\ a(a2, complete)$. $is\ a(a3, namemaker)$.
$is\ a(a4, schedule)$. $time(a1, 10)$. $time(a2, 25)$. $time(a3, 22)$. $time(a4, 23)$.
$actor(a1, paul)$. $actor(a2, paul)$. $is\ a(paul, user)$. $actor(a3, paul)$.
$actor(a4, mary)$. $is\ a(mary, admin)$.

These ground atoms describe the process executions $e1$ and $e2$ (reference objects) according to the activities $a1$, $a2$, $a3$, and $a4$ (task-relevant objects) and the actors, $u1$ and $u2$ (task-relevant objects). Differently, an example of a normal logic program stored as intensional database D_I is the following:

$$before(A1, A2) \leftarrow activity(C, A1), activity(C, A2), A1 \neq A2, time(A1, T1), time(A2, T2),$$
$$T1 < T2, not(activity(C, A), A \neq A1, A \neq A2, time(A, T), T1 < T, T < T2)$$

This normal logic program defines the temporal relationship *before* and permits to entail the temporal ground atoms $before(a1, a2)$ and $before(a3, a4)$.

The set of ground atoms (extensionally or intensionally) stored in D is partitioned with respect to the process executions into a number of non-intersecting subsets $D[e]$ (units of analysis) each of which includes ground atoms concerning the activities and actors involved in the process execution e. Then, SPADA is able to discover relational frequent patterns across the units of analysis of D which are associated to the process executions. The discovery process is in charge of a levelwise method, that is tailored as a breadth-first search in the lattice of relational patterns spanned by the θ-subsumption generality order ($\succ_\theta$). In this context, the relational patterns are formulated as $process(P), \mu(P)\ [s]$, where P is a variable to represent a process execution, $process(P)$ is the atom that identifies a process execution P, while $\mu(P)$ is a conjunction of atoms which provides a description of a fragment of the process model underlying the generation of the process execution P, s is the support of P in D. Each atom in $\mu(P)$ represents one of the property of activity, actor or process execution, or relationship between activities, process executions and activities, activities and actors. For example:

$$process(P), complete(P, A), schedule(P, B), delete(P, C)\ before(A, B),$$
$$before(B, C). \qquad [support = 63\%]$$

is a relational pattern which describes a fragment of a process model where the activities complete, schedule and delete are executed in a sequence. The support s estimates the probability $p(process(P) \cup \mu(P))$ on D. This means that $s\%$ of the units of analysis $D[e]$ are covered by $process(P) \cup \mu(P)$. Formally, the unit of analysis $D[e]$ is covered by $process(P) \cup \mu(P)$ if there exists a substitution $\theta = \{P \leftarrow e\} \cdot \theta_1$ such that $[process(P) \cup \mu(P)]\theta \subseteq D[e]$.

3 Discovering Process Models with Variants

The motivation behind the usage of disjunctive forms in patterns is that the set of patterns discovered with traditional approaches, included SPADA, strongly depends on frequency-based thresholds such as the minimum support threshold. This means that when the minimum support is high valued, many interesting patterns are filtered out: conjunctions of atoms, for which the considered statistical measure does not exceed the minimum threshold, are ignored. The introduction of the disjunctive forms would permit to include the atoms which occur in parallel to or in alternative to other atoms. The effect is that of increasing the values of the statistical measures associated to the patterns. For example, let us suppose that the atom $complete(A, D)$ may occur alternatively to the atom $schedule(A, D)$. Then, the relational pattern:

$$process(A), complete(A, B), \langle \mathbf{complete}(\mathbf{A}, \mathbf{D}) \vee \mathbf{schedule}(\mathbf{A}, \mathbf{D})\rangle, before(B, D)$$

might be frequent, although the patterns

$$process(A), complete(A, B),\ complete(A, D),\ before(B, D) \text{ and}$$
$$process(A), complete(A, B), schedule(A, D), before(B, D)$$

might be both infrequent.

This consideration advocates the starting point of our approach, which is that of considering infrequent conjunctive patterns. These patterns are re-evaluated and extended to the disjunctive form by inserting disjunctions which involve atoms already present in the patterns. Disjunctions are created among atoms which are semantically related in the application domain. The semantic relatedness is intended as background knowledge on the predicates used in the atoms and permits us to numerically quantify the dissimilarity or conceptual distance between atoms. It guarantees that meaningful disjunctions are created.

The proposed approach follows a two-stepped procedure. First, it extracts the infrequent conjunctive patterns which can be considered as basis for the disjunctive patterns construction. In particular, patterns whose support is lower than the minimum support threshold, but exceeds a new ad-hoc threshold are selected. The newly defined threshold permits to identify the set of patterns to be extended to the disjunctive form. Second, by following the main intuition reported in [9], background knowledge is accommodated to exploit the information on the dissimilarity among the atoms in the disjunctive pattern generation. This way, disjunctive patterns are computed by iteratively integrating disjunctions into the patterns by means of a pair-wise joining operation. The final result is a set of patterns which may comprise both conjunctions and disjunctions of atoms, whose support is greater than the minimum support threshold.

Working in the relational setting adds additional sources of complexity to the problem of joining patterns due to the *linkedness* property [8]. In the relational representation, atoms of the same pattern are dependent each other due to the presence of variables (differently from the items in the propositional representation [9]). In this work, patterns to be joined should differ in only one atom (but different atoms should be similar) and share the remaining atoms up to a redenomination of variables.

Before formally defining the problem we solve in this paper, we clarify how the deductive database we have described in the previous section changes. In particular, the intensional part D_I of the deductive database D includes the definition of a new kind of domain knowledge that permits to express the dissimilarity among atoms in the form of *Datalog* weighted edges of a graph. An example of the Datalog weighted edge is the following: $schedule - (complete - 0.88)$. It states that the dissimilarity between the predicates *schedule(·,·)* and *complete(·,·)* is 0.88. More generally, it represents an undirected edge e between two vertices v_i and v_j (e.g., *schedule*, *complete*) with weight w_{ij} (e.g., 0.88) and it is denoted as $e(v_i, v_j, w)$. A finite sequence of undirected edges $e_1, e_2, \ldots, e_m$ which links the vertices v_i and v_j is called *path* and denoted as $\rho(v_i, v_j)$. The complete list of such undirected edges represents the background information on the dissimilarity among atoms and allows the algorithm to join patterns by introducing disjunctions (e.g., $\langle schedule(A, W) \vee complete(A, W) \rangle$). The formal statement of the problem of discovering relational frequent patterns with disjunctions can be articulated in two steps.

1. *Given:* the extensional part D_E of the deductive database D and the normal logic programs stored into the intensional part D_I of D, two thresholds $minSup \in [0; 1]$ and $nSup \in [0; 1]$, where the former represents a minimum support value, while the latter represents a maximum support value ($nSup <$

minSup), *Find:* the collection I_R of the relational infrequent patterns whose support is between *nSup* and *minSup*.
2. *Given:* the collection I_R, the Datalog weighted edges stored in the intensional part D_I of the deductive database D and two thresholds *minSup* and $\gamma \in [0;1]$ (γ defines the maximum dissimilarity value among atoms involved into a disjunction), *Find:* relational patterns with disjunctions whose support exceeds *minSup* and whose dissimilarity of atoms involved in the disjunctions does not exceed γ.

The computational solution to these problems is implemented in jSPADA.

3.1 Mining Infrequent Conjunctive Relational Patterns

In jSPADA, the search is based on the level-wise method and implements a two-stepped procedure: i) generation of candidate patterns with k atoms (k-th level) by considering the frequent patterns with $k-1$ atoms ($k-1$-th level); ii) evaluation of the support of patterns with k atoms. So, the patterns whose support does not exceeds *minSup* will be not considered for the next level: the patterns discarded (infrequent) at each level are rather considered for the generation of disjunctions. The collection I_R is thus composed of a subset of infrequent patterns, more precisely those with support greater than or equal to *nSup* (and less than *minSup*).

We observe that, the set of infrequent patterns I_R computed by jSPADA do not contain all possible infrequent relational patterns. This depends on the fact that any k-level infrequent pattern “$P_{k-1} \wedge A$” output by jSPADA has the head P_{k-1} that is a relational pattern frequent at the level $k-1$, and the tail A which is an atom such that the conjunction “$P_{k-1} \wedge A$” is infrequent. This means that infrequent patterns whose head is already infrequent cannot be discovered by jSPADA. On the other hand computing all possible infrequent relational patterns is computationally expensive. As a consequence, the set of patterns with disjunctions discovered from I_R is a necessary approximation of the complete set of relational patterns with disjunctions.

3.2 Extending Relational Patterns with Disjunctions

The generation of disjunctive relational patterns is performed by creating disjunctions among similar atoms in accordance to the weighted edges of the background knowledge: two patterns which present similar atoms are joined to form only one. The implemented algorithm (see Algorithm 1) is composed of two subprocedures: the first one (lines 2-12) creates a graph $\mathcal{G}_\mathcal{D}$ with the patterns of I_R by exploiting the knowledge defined in D_I, while the second one (lines 13-32) joins two patterns (vertices) on the basis of the information (weight) associated to the connecting edge. In particular, for each pair of patterns which have the same length (namely, at the same level of the level-wise search method) it checks whether they differ in only one atom and share the remaining atoms up to a re-denomination of variables (line 3). Let α and β be the two atoms differentiating P from Q (α in P, β in Q), a path ρ which links α to β (or vice-versa) is searched among the weighted edges according to D_I: in the case the sum ω of the weights found in the path is lower than the maximum dissimilarity γ the vertices P and Q are inserted into $\mathcal{G}_\mathcal{D}$ and linked through an edge with weight ω (lines 4-9). Note that when there is more than one path between α and β, then the path with lowest weight is considered. Intuitively, at the end of the first sub-procedure, $\mathcal{G}_\mathcal{D}$ will contain, as vertices, the patterns which meet the condition at the line 3, and it will contain, as edges, the weights associated to the path linking the atoms differentiating the patterns.

Once we have $\mathcal{G}_\mathcal{D}$, a list $\mathcal{L}_\mathcal{D}$ is populated with the vertices and edges of $\mathcal{G}_\mathcal{D}$: an element of $\mathcal{L}_\mathcal{D}$ is a triple $\langle P, Q, \omega \rangle$ composed of a pair of vertices-patterns (P, Q) with their relative weight. Elements in $\mathcal{L}_\mathcal{D}$ are ranked in ascending order with respect to the values of ω so that the pairs of patterns with lower dissimilarity will be joined for first. This guarantees that disjunctions with very similar atoms will be preferred to the others (line 13). For each element of $\mathcal{L}_\mathcal{D}$ whose weight ω is lower than γ the two patterns P and Q are joined to generate a pattern J composed by the conjunction of the same atoms in common to the two patterns P and Q and of the disjunction formed by the two different (but similar) atoms (lines 14-15). This joining procedure permits to have patterns with the same length of the original ones and which occur when at least one of the original patterns occurs. Therefore, if a pattern J is obtained by joining P and Q, it covers a set of units of analysis equal to the union of those of P and Q: the support of J is determined as in line 16 and, generally, it is higher than the support of both P and Q. In the case the support of J exceeds $minSup$, then it can be considered statistically interesting and no further processing is necessary (lines 16-17). Otherwise, J is again considered and inserted into $\mathcal{G}_\mathcal{D}$ as follows. The edges which linked another pattern R of $\mathcal{G}_\mathcal{D}$ to P and Q are modified in order to keep the links from R to J: the weight of the edges between one pattern R and J will be set to the average value of the weights of all the edges which linked R to P and Q (lines 19-27). The modified graph $\mathcal{G}_\mathcal{D}$ contains conjunctive patterns (those of I_R) and patterns with disjunctions (those produced by joining). Thus, $\mathcal{G}_\mathcal{D}$ is re-evaluated to extend disjunctions previously created or to insert into disjunctive patterns other disjunctions obtained with other atoms. The algorithm proceeds iteratively (line 29-30) until no additional disjunction can be performed (namely, when $\mathcal{L}_\mathcal{D}$ is empty or the weights ω are higher than γ). At each iteration, the patterns P and Q are removed from $\mathcal{G}_\mathcal{D}$ (line 32).

An explanatory example is illustrated in Figure 1. Let us consider the background knowledge D_I on the dissimilarity among four atoms and the set I_R containing four infrequent conjunctive relational patterns as illustrated in Figure 1a and γ equal to 0.7. The first sub-procedure of Algorithm 1 analyzes P_1, P_2, P_3, P_4 and discovers that they differ in only one atom, while the other atoms are in common ($process(A), unknown(A, C), before(B, C)$). Then, it creates the graph $\mathcal{G}_\mathcal{D}$ by collocating P_1, P_2 and P_3 in three different vertices and linking them through edges whose weights are taken from the paths ρ in D_I. P_4 is not considered because the dissimilarity between *start* and *resume* in the graph is higher than γ (row (1) in Figure 1b). The second sub-procedure starts

Algorithm 1 Extending Relational Pattern with Disjunctions.

```
input: $I_R, D_I, \gamma, minSup$
output: $\mathcal{J}$    // $\mathcal{J}$ set of disjunctive patterns
for all $(P, Q) \in I_R \times I_R,\ Q \neq P$ do
  if $P.length = Q.length$ and $check\_atoms(P, Q)$ then
    $(\alpha, \beta) := atoms\_diff(P, Q)$    //$\alpha, \beta$ atoms differentiating $P$ and $Q$
    if $\rho(\alpha, \beta) \neq \oslash$ then
      $\omega := \sum_{e(v_i, v_j, w_{ij})\ in\ \rho(\alpha, \beta)} w_{ij}$
      if $\omega \leq \gamma$ then
        $addNode(P, \mathcal{G_D});\ addNode(Q, \mathcal{G_D}); addEdge(P, Q, \omega, \mathcal{G_D});$
      end if
    end if
  end if
end for
$\mathcal{L_D} \leftarrow$ edges of $\mathcal{G_D}$    // list of edges of $\mathcal{G_D}$ ordered in ascending mode w.r.t. $\omega$
while $\mathcal{L_D} \neq \oslash$ and $\forall e(P, Q, \omega) \in \mathcal{G_D}\ \ \omega \leq \gamma$ do
  $J \leftarrow join(P, Q);\ J.support := P.support + Q.support - (P \cap Q).support;$
  if $J.support \geq minSup$ then
    $\mathcal{J} := \mathcal{J} \cup \{J\}$
  else
    for all $R$ such that $\exists\ e(P, R, \omega_1) \in \mathcal{G_D}$ and $\exists\ e(Q, R, \omega_2) \in \mathcal{G_D}$ do
      $addEdge(R, J, (\omega_1 + \omega_2)/2, \mathcal{G_D})$
    end for
    for all $R$ such that $\exists\ e(P, R, \omega_1) \in \mathcal{G_D}$ and $\nexists\ e(Q, R, \omega_2) \in \mathcal{G_D}$ do
      $addEdge(R, J, \omega_1, \mathcal{G_D})$
    end for
    for all $R$ such that $\exists\ e(Q, R, \omega_2) \in \mathcal{G_D}$ and $\nexists\ e(P, R, \omega_1) \in \mathcal{G_D}$ do
      $addEdge(R, J, \omega_2, \mathcal{G_D})$
    end for
    $\mathcal{L_D} \leftarrow$ edges of $\mathcal{G_D}$;  $update\ \mathcal{L_D}$
  end if
  $removeNode(P, \mathcal{G_D});\ removeNode(Q, \mathcal{G_D})$
end while
```

by ordering the weights of the edges: the first disjunction is created by joining P_1 and P_3 given that the dissimilarity value is lower than γ and the lowest (row (2) in Figure 1b). Next, the pattern so created and P_2 are checked for joining. Both have the same length and differ in only one atom. Although the first presents a disjunction and the second presents a "simple" atom, dissimilarity is lower than γ and a new disjunctive pattern is created (row (3) in Figure 1b).

A final consideration concerns the time complexity of Algorithm 1. Let us consider the set I_R partitioned into disjoint subsets $\{I_{Rk}\}_k$ on the basis of the pattern length k. Let n_k be the cardinality of I_{Rk}. For each k, the time complexity of extending relational patterns in I_{Rk} with disjunctions is quadratic in n_k at worst (this is due to the pairwise join operation).

4 Experiments

Experiments are performed by processing event log provided by THINK3 Inc[1]. This dataset describes 353,490 executions of business processes in a company. The period under analysis is from April 7th 2005 to January 10th 2007 for a total of 1,035,119 activities and 103 actors. Activities are classified as *tools* (131),

[1] http://www.think3.com/en/default.aspx

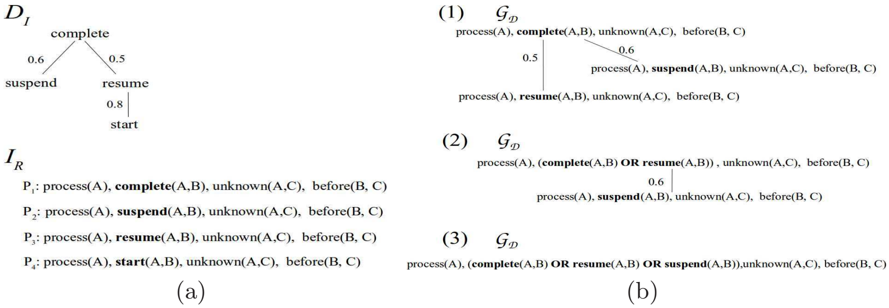

D_I
complete
0.6
0.5
suspend
resume
0.8
start
I_R
P1: process(A), complete(A,B), unknown(A,C), before(B, C)
P2: process(A), suspend(A,B), unknown(A,C), before(B, C)
P3: process(A), resume(A,B), unknown(A,C), before(B, C)
P4: process(A), start(A,B), unknown(A,C), before(B, C)
(a)
(1) $\mathcal{G}_\mathcal{D}$
process(A), complete(A,B), unknown(A,C), before(B, C)
0.6
0.5
process(A), suspend(A,B), unknown(A,C), before(B, C)
process(A), resume(A,B), unknown(A,C), before(B, C)
(2) $\mathcal{G}_\mathcal{D}$
process(A), (complete(A,B) OR resume(A,B)) , unknown(A,C), before(B, C)
0.6
process(A), suspend(A,B), unknown(A,C), before(B, C)
(3) $\mathcal{G}_\mathcal{D}$
process(A), (complete(A,B) OR resume(A,B) OR suspend(A,B)),unknown(A,C), before(B, C)
(b)

Fig. 1. An example of relational pattern extension from disjunctive atoms (γ=0.7).

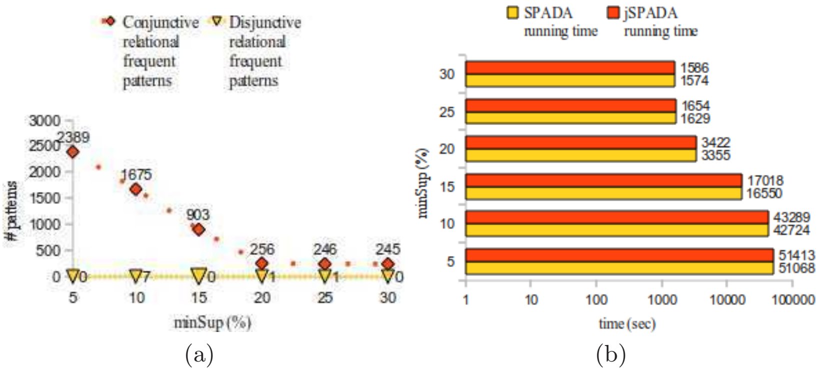


Fig. 2. Learning times and number of patterns discovered by SPADA and jSPADA by varying *minSup* (*nSup*=0.1%, γ=0.6).

workflow (919,052), *namemaker* (106,839), *delete* (2,767), *deleteEnt* (2,354), *prpDelete* (471), *prpSmartDelete* (53), *prpModify* (34) and *cast* (1,430). Actors are classified as *user* (103), *viewer* (3) or *administrator* (2). In this Section, we illustrate the results obtained with a random sample of THINK3 data including 3580 executions. Process instances play the role of reference objects, while activities and actors play the role of task-relevant objects.

The goal of the experiments is to compare the conjunctive patterns discovered by SPADA with those disjunctive discovered by jSPADA in terms of the cardinality of the extracted patterns set and in terms of the learning time by varying threshold values. In the background knowledge D_I, a constant weight is assigned to each pair of activities. This way, the generation of a disjunctive form may equally involve each one of the activities. The weight is computed as the ratio of 1 (maximum dissimilarity value) to the number of distinct activities. In In THINK3, the weight is set 0.16 for each one of the six activities, while it is set to 0.33 for each one of the three actors.

Experiments are performed[2] by tuning the thresholds $minSup$, $nSup$ and γ (see Figure 2.a). As we expected, the number of final patterns (the summation of Conjunctive relational frequent patterns and Disjunctive relational frequent patterns) decreases as $minSup$ increases. Indeed, by enlarging the range $[nSup; minSup)$, the number of infrequent conjunctive patterns and the number of potential disjunctive patterns grow up, while the increase of $minSup$ leads to discover only those really frequent. Differently, narrowing the range $[nSup; minSup)$ leads to a larger set of Conjunctive frequent patterns but also to a smaller set of Conjunctive infrequent patterns which will be used in the pair-wise joining operation, and finally to a reduced set of disjunctive frequent patterns, as in the Figure 2.a when $minSup$=5%.

An interesting consideration is that the order of magnitude of the number of discovered disjunctive patterns is reasonably small (lower than 30). This permits to reach the objective of discovering process models with variants which are not difficult to interpret for the end-user. Another consideration can be found by the analysis of learning times (Figure 2.b) where it is possible to see that SPADA and jSPADA show comparable learning times. A deeper analysis reveals that by increasing $minSup$, the computational cost spent for the only generation of disjunctive patterns is of at least two orders of magnitude smaller than that of SPADA. Indeed, the increase of $minSup$ leads to enlarge the range $[nSup; minSup)$ and this would require longer learning time to process a greater set I_R. Actually, the reason of these time performances is twofold: first, a larger set I_R does not necessarily imply a larger set of disjunctive patterns given that, if the atoms of two patterns are different, no disjunction can be created; second, when $minSup$ has low values (e.g., 10%) the number of iterations in Algorithm 1 is smaller since disjunctive patterns support easily exceeds $minSup$.

A peculiarity of the approach is that it enriches relational patterns discovered by SPADA with additional atoms. For instance, the following pattern is discovered by jSPADA at $minSup$=20% ($nSup$=0.001,γ=0.6 in THINK3):

$$P_1: \ process(A), \langle namemaker(A,B) \vee workflow(A,B)\rangle, user(B,C), \\ loggroup(C, ekm\_j) \qquad [support = 21.9\%]$$

P$_1$ states that activities $namemaker$ and $workflow$ occur one in alternative to the other. This happens in 785 executions out of 3850 executions where the actor is of kind $user$ and the login mode is ekm_j. P$_1$ joins the following two conjunctive patterns discovered by SPADA at $minSup$=5% that could represent two variants of the same model:

$$P_2: process(A), namemaker(A,B), user(B,C), loggroup(C, ekm\_j) \quad [support = 8.1\%]$$

$$P_3: process(A), workflow(A,B), user(B,C), loggroup(C, ekm\_j) \quad [support = 19.8\%]$$

Therefore, the proposed approach permits to unearth information extracted from SPADA at lower computational cost since the parameter $minSup$ strongly affects the learning times of both systems.

[2] Additional results are accessible at http://www.di.uniba.it/~loglisci/jSPADA/.

5 Conclusions

In this paper, we present an approach to discover process models in the form of frequent relational patterns with disjunctions. Patterns describe activities and actors involved in the processes. Disjunctions permit to express $OR-split/OR-join$ constructs such that variants of process models can be identified. Experiments on real event logs empirically prove the effectiveness of the proposed approach. Discovered patterns permit to express variants and can be easily interpreted by end-users. As future work, we plan to investigate the applicability of relational frequent pattern mining in order to also mine loops.

Acknowledgment

This work is in partial fulfillment of the research objectives of the project ATENEO-2010: "Modelli e Metodi Computazionali per la Scoperta di Conoscenza in Dati Spazio-Temporali".

References

1. A. Appice, M. Ceci, A. Turi, and D. Malerba. A parallel, distributed algorithm for relational frequent pattern discovery from very large data sets. *Intelligent Data Analysis*, 15(1):69–88, 2011.
2. J. E. Cook and A. L. Wolf. Software process validation: Quantitatively measuring the correspondence of a process to a model. *ACM Trans. Softw. Eng. Methodol.*, 8(2):147–176, 1999.
3. S. Goedertier, D. Martens, B. Baesens, R. Haesen, and J. Vanthienen. A new approach for discovering business process models from event logs. In *Technical Report*. KBI 0716, 2007.
4. G. Greco, A. Guzzo, L. Pontieri, and D. Sacca'. Discovering expressive process models by clustering log traces. *IEEE Transactions on Knowledge and Data Engineering*, 18:1010–1027, 2006.
5. S.-Y. Hwang, C.-P. Wei, and W.-S. Yang. Discovery of temporal patterns from process instances. *Comput. Ind.*, 53(3):345–364, 2004.
6. E. Lamma, P. Mello, F. Riguzzi, and S. Storari. Applying inductive logic programming to process mining. In *ILP 2007*, pages 132–146, 2007.
7. C. Li, M. Reichert, and A. Wombacher. Discovering reference models by mining process variants using a heuristic approach. In *BPM*, pages 344–362, 2009.
8. J. W. Lloyd. *Foundations of Logic Programming, 2nd Edition*. Springer, 1987.
9. J. F. Roddick and P. Fule. Semgram - integrating semantic graphs into association rule mining. In *AusDM*, pages 129–137, 2007.
10. W. van der Aalst, V. Rubin, H. Verbeek, B. van Dongen, E. Kindler, and C. Gnther. Process mining: a two-step approach to balance between underfitting and overfitting. *Software and Systems Modeling*, 9:87–111, 2010.
11. W. M. P. van der Aalst and A. J. M. M. Weijters. Process mining: a research agenda. *Comput. Ind.*, 53(3):231–244, 2004.
12. W. M. P. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE TKDE*, 16(9):1128–1142, 2004.

A Probabilistic Hierarchical Approach for Pattern Discovery in Collaborative Filtering Data

Extended Abstract

Nicola Barbieri[1][2], Giuseppe Manco[2], and Ettore Ritacco[2]

[1] Department of Electronics, Informatics and Systems - University of Calabria,
via Bucci 41c, 87036 Rende (CS) - Italy
nbarbieri@deis.unical.it,
[2] Institute for High Performance Computing and Networks (ICAR)
Italian National Research Council
via Bucci 41c, 87036 Rende (CS) - Italy
{barbieri,manco,ritacco}@icar.cnr.it

Abstract. This paper presents a hierarchical probabilistic approach to collaborative filtering which allows the discovery and analysis of both global patterns (i.e., tendency of some products of being 'universally appreciated') and local patterns (tendency of users within a community to express a common preference on the same group of items). The core of our approach is a probabilistic co-clustering strategy, arranged in a hierarchical fashion: first, user communities are discovered, and then the information provided by each user community is used to discover topics, grouping items into categories. The experimental evaluation shows that the proposed model achieves a competitive prediction accuracy with respect to the state-of-art collaborative filtering approaches.

1 Introduction and Preliminaries.

Recommender Systems (RS) provide users with personalized suggestions about new products, services, and information. According to the *Collaborative Filtering (CF)* approach, RS are based exclusively on a database of user preferences. Traditional CF approaches try to foresee the preferences of users on previously unseen products, by analyzing and discovering patterns in a high dimensional and extremely sparse preference matrix.
Collaborative filtering data exhibit global patterns (i.e. tendencies of some products of being 'universally' appreciated) as well significative local patterns (i.e tendency of users belonging to a specific community to express similar preference values on the same items). Typically, local patterns can be better detected by means of co-clustering approaches [4, 6, 7]. Unlike traditional CF techniques, which try to discover similarities between users or items using clustering techniques or matrix decomposition methods, co-clustering approaches aim to partition data into homogenous blocks enforcing a simultaneous clustering on both the dimensions of the preference data.

A main weakness of the current approaches to co-clustering is the static structure enforced by fixed row/column blocks where both users and items have to fit. This paper presents a novel probabilistic hierarchical approach which is able to discover both global and local trends in data, allowing different user communities to show different interpretations of item categories.The proposed schema differs from the previously proposed coclustering approaches to CF data because it does not assume the existence of a unique partition on the item-set: each user community is characterized by having its own set of topics involving items and user preferences. Following a hierarchical clustering approach, we (i) determine user communities by gathering together similar users, then (ii) for each user community the clustering phase produces a mixture of topics upon which the item set and the user preferences are accommodated into categories. The hierarchical coclustering model does not enforce any strong assumption on the membership of users and items improving the flexibility of the model itself. Each user participates to different user communities with a certain degree and, given a user community, each item may belong to different item-categories. As a result, the proposed model summarizes the advantages of a flexible probabilistic structure for user profiling and a competitive prediction accuracy on user ratings.

A RS consists of a set of M users $\mathcal{U} = \{u_1, \cdots, u_M\}$, which will be indicated for short as the *user-set*, a set of N items $\mathcal{I} = \{i_1, \cdots, i_N\}$, named *item-set*, and a collection of rating values expressing the preference of one user on a corresponding item. Such collection of preference indicators can be represented as a $M \times N$ rating matrix $\mathbf{R}$, where r_i^u is the rating given by the user u on the item i. Tipically, ratings are integer values within a scale 1 (low interest) to V (strong interest). Even in the case of a very dynamic system, the rating matrix is typically characterized by an exceptional sparsity rate.

We will further denote by $\hat{r}_i^u$ the predicted rating for the pair (u, i). Considering the case of users and products which have provided/received at least one preference value, several evaluation metrics have been proposed to quantify the quality of a prediction algorithm. One of the most referenced methods to measure the performance of a predictor is the Root Mean Squared Error, which emphasizes large errors.

In a probabilistic settings, we adopt random variables R, I, and U denoting a rating, an item and a user respectively. With an abuse of notation, we shall omit the random variable in the specification. For example, $P(r, u, i)$ shall denote the joint probability $P(R = r, U = u, I = i)$.

2 A Hierarchical Co-Clustering Approach for Modeling User Preferences

The starting point in our approach is the observation that different communities can infer different evaluations of the same item. Specific groups of users tend to be co-related according to different subsets of features. However, though semantically-related, two users with (possibly several) differences in their item ratings would hardly be recognized as actually similar by any global model imposing a fixed structure for item categories. Viewed in this perspective, the identification of *local patterns*, i.e. of proper combinations of users and items, would lead to the discovery of natural clusters in the data.

The generative model for the proposed schema can be summarized as follows:

1. For each user $u \in \mathcal{U}$ choose a distribution over user communities $P(c|u)$ and a distribution over local topics $P(d|c, u)$
2. For each item $i \in \mathcal{I}$ and for each user attitude $k = \{1, \cdots, K\}$, choose a distribution $P(d|c_k, i)$
3. For each pair $\langle u, i \rangle$
   (a) Choose a user community $c_k \sim P(c|u)$
   (b) Choose an item category $d_h \sim P(d|c_k, i)P(d|u, c_k)$
   (c) Generate a rating value according to the distribution on rating values conditioned on the community c_k and on the item-topic d_h: $P(r|c_k, d_h)$

Formally, we can assume that the probability of observing the rating value r given the pair $\langle u, i \rangle$ is

$$P(r|u,i) = \sum_{k=1}^{K} P(c_k|u)P(r|u,i,c_k)$$
$$P(r|u,i,c_k) = \sum_{h=1}^{H_k} P(d_h|c_k,i)P(d_h|u,c_k)P(r|c_k,d_h)$$

which, considering the structural dependency between c_k and its inner topics d_h can be simplified (in the notation) as:

$$P(r|u,i) = \sum_{k=1}^{K} P(c_k|u) \sum_{h=1}^{H_k} P(d_h|i)P(d_h|u)P(r|d_h) \tag{1}$$

The hierarchical model for users' ratings consists in a set of K user community models and for each of them a set of H_k topic models which represent "local" preference patterns. The user community level specifies the probabilities $\gamma_{uk} = P(c_k|u)$ with $k = 1, \ldots, K$, which measure how much the ratings given by the user u fit the preference behavior underlined by each of the communities.

Due to the strong coupling between the user community latent variable c and the one corresponding to local patterns d, the exact inference for the model, which would maximize both the user community cohesion and the local topic similarity, is difficult to solve analytically. Hence, we adopt an approximated solution, based on a hard clustering policy for user communities, such that the inference of the parameters can be performed efficiently without compromising the generative semantic and the flexibility of the model.

We devise a hierarchical approach to the estimation of the components involved into Eq. 1.The general scheme of the proposed hierarchical approach can be summarized as follows. Given a rating matrix $\mathbf{R}$:

– Discover K user communities;

- For each of those communities, according to an hard clustering approach,
  - Select a subset of users that belong to the considered community $\mathcal{U}_k = \{u \in \mathcal{U} | p(c_k|u) \geq p(c_j|u), j = 1, \dots, K\}$, and $\mathbf{R}_k$ the corresponding submatrix of $\mathbf{R}$;
  - Generate a set of H_k topic models for $\mathbf{R}_k$.

The probability of observing the rating r for the pair (u, i) can be computed as:

$$P(r|i, u) = \sum_k \gamma_{uk} \cdot \tilde{P}(r|i, u, c_k) \tag{2}$$

where the probabilities γ_{uk} act as mixture weights and the distribution over rating values corresponding to the community c_k is computed taking into account both global and local patterns:

$$\tilde{P}(r|i, u, c_k) = \begin{cases} P(r|i, u, c_k) & \text{if } u \in \mathcal{U}_k \\ P(r|i, c_k) & \text{otherwise} \end{cases} \tag{3}$$

2.1 Modeling User Communities and Local Patterns.

The discovery of the communities is accomplished essentially via a model-fitting procedure based on a maximum-likelihood estimation. In practice, we assume that the rating matrix $\mathbf{R}$ is modelled as a set of user vectors, where each vector is characterized by the preferences of the user. Formally, this means that we can model the probability $p(r, i|u)$ for each triplet $\langle r, i, u\rangle$.

The corresponding probability of observing a user, hence, corresponds to the joint probability of observing all his ratings, that is

$$P(u|\Theta, \mathbf{R}) = \sum_{j=1}^{K} \prod_{i=1}^{N} \prod_{r=1}^{V} \pi_j \left(\alpha_{ij} \cdot \sigma_{rij}\right)^{\delta(u,i,r)}$$

where a single community c_j is characterized by the parameters $\pi_j = P(c_j)$, $\alpha_{ij} = P(i|c_j)$, $\sigma_{rij} = P(r|c_j, i)$ and $\delta(u, i, r)$ is an indicator function which is equal to 1 if $r_i^u = r$, zero otherwise. This modeling allows us to adopt a maximum likelihood approach to the estimation of the Θ parameters characterizing the $P(i|\Theta)$ and $P(r|i, \Theta)$.

Estimating the parameters by means of an EM procedure yields the following equations:

E-Step:

$$\gamma_{uj} = P(c_j|u) = \frac{P(u|c_j) \cdot \pi_j}{\sum_{j'=1}^{K} P(u|c_{j'}) \cdot \pi_{j'}}$$

M-Step:

$$\pi_j = \frac{\sum_{u=1}^{M} \gamma_{uj}}{M}$$

$$\alpha_{ij} = \frac{\sum_{u=1}^{M} \gamma_{uj} \sum_{r=1}^{V} \delta(u, i, r)}{\sum_{u=1}^{M} \gamma_{uj} \sum_{i'=1}^{N} \sum_{r=1}^{V} \delta(u, i', r)} \qquad \sigma_{rij} = \frac{\sum_{u=1}^{M} \gamma_{uj} \cdot \delta(u, i, r)}{\sum_{u=1}^{M} \sum_{r'=1}^{V} \gamma_{uj} \cdot \delta(u, i, r')}$$

A further advantage of the above formalization is the possibility of exploiting the above model for prediction purposes. A prediction function in fact can be defined as

$$\hat{r_i^u} = E[R|u,i] = \sum_{r=1}^{V} r \cdot \sum_k \sigma_{rik} \cdot \gamma_{uk} \tag{4}$$

We shall see in the following that the resulting baseline function is even competitive with state-of-the art approaches.

The above formalization also allows an alternative gaussian model: $P(r|i,c_j) = N(r;\mu_{ij},\sigma_{ij})$ where μ_{ij} and σ_{ij} are, respectively, the average and the variance of ratings assigned to the item i in the user community j, that can be estimated employing an EM approach. This modeling allows as to employ a Z-score normalization as proposed in [5].

The approach to the discovery of local community patterns is based again on a EM procedure which aims at maximizing the likelihood of the $\mathbf{R}_k = \{\langle r,u,i\rangle | p(c_k|u) \geq p(c_j|u), j = 1,\ldots,K\}$ rating matrix associated to a community model c_k. The EM algorithm can hence be defined in terms of the following formulas:

- **E-Step:**

$$\psi_k(h;r,i,u) = \frac{P(r|d_h)P_k(d_h|i)P_k(d_h|u)}{\sum_j P(r|d_j)P_k(d_j|i)P_k(d_j|u)}$$

- **M-Step:**

$$P_k(d_h|i) = \frac{\sum_u^M \sum_r^V \psi_k(h;r,i,u)}{\sum_{h'} \sum_u^M \sum_r^V \psi_k(h';r,i,u)} \qquad P_k(d_h|u) = \frac{\sum_i^N \sum_r^V \psi_k(h;r,i,u)}{\sum_{h'} \sum_i^N \sum_r^V \psi_k(h';r,i,u)}$$

where $P(r|d_h) = N(r;\mu_{d_h},\sigma_{d_h})$ and mean and variance parameters are estimated as:

$$\mu_{d_h} = \frac{\sum_{\langle u,i,r\rangle} \psi_k(h;r,i,u)\cdot r}{\sum_{\langle u,i,r\rangle} \psi_k(h;r,i,u)} \qquad \sigma_{d_h} = \frac{\sum_{\langle u,i,r\rangle} \psi_k(h;r,i,u)\,(r-\mu_{d_h})^2}{\sum_{\langle u,i,r\rangle} \psi_k(h;r,i,u)}$$

The estimation of the correct number of clusters is accomplished by resorting to a Cross-Validation approach based on a penalized Log-Likelihood principle. Given a set D of observations we aim at finding the model parameters Θ maximizing the probability $P(\Theta|D)$ according to two opposing requirements: the fitting of the data and the complexity of the model (number of parameters).

3 Evaluation

We evaluate the effectiveness of the proposed technique considering both the prediction capabilities of the User Community Model, adopted in the first stage in discovering communities, and of the hierarchical model on two popular benchmark datasets (Netflix and Movielens). We exploited a subsample of the whole

	Nextflix		MovieLens	
	Training Set	Test Set	Training Set	Test Set
Users	435,656	389,305	6,040	6,040
Items	2,961	2,961	3,706	3,308
Ratings	5,714,427	3,773,781	800,168	200,041
Avg ratings (user)	13.12	9.69	132,47	33,119
Avg ratings (item)	1929.90	1274.50	215.91	60.47
Sparseness Coeff	0,9956		0,9643	

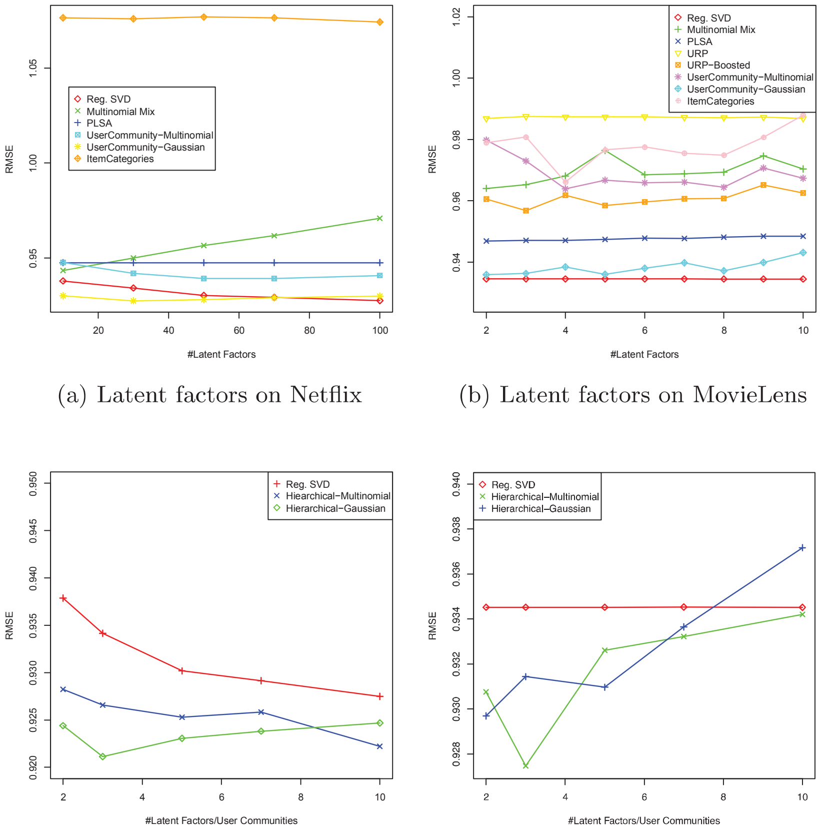
Table 1. Summary of the Data used for validation.

Netflix dataset, and partition the data into training and test set. Table 1 summarizes relevant information about these datasets.

We compare our approach with most algorithms; in particular, we directly implemented the Reg. SVD [3], G-PLSA [5], FMM [6], Multinomial Mixture Model [10] and URP [9]. Other algorithms not listed here will be discussed separately in the end of the section.

In a first set of experiments, we evaluate the performance achieved by the User Community Models, considering both the Multinomial and the Gaussian version and performed a suite of experiments varying the number of user communities and compared the obtained RMSE values with the ones achieved by the Gaussian pLSA algorithm on the same data. Performance results of the two User Communities Models and G-PLSA are shown in Figures 1(a) and 1(b). Considering Netflix, the multinomial User Community approach and the G-PLSA do not produce a significant improvement over the Cinematch base, which is close to 0.95; for both these models the best RMSE values is achieved by considering 150 user communities. The average RMSE for the G-PLSA model is 0.9474 and only minor improvements on this result are observed varying the number of clusters. The gaussian User Community version outperforms both the multinomial model and G-PLSA, achieving the best RMSE value of 0.9280 when 30 user communities are employed. We were not able to extensively report on FMM and URP on Netflix, due essentially to the high computational resources needed by these models. For URP, the only model we were able to compute required 17,340secs and did not exhibit significant results. We were able, however, to thoroughly experiment on MovieLens with these models as well. Surprisingly, the multinomial User Community model has a significant worsening on MovieLens. While the Gaussian model is still competitive, probably due to the z-score normalization, the multinomial model seems to suffer more the skewness of the dataset.

The hierarchical schema allows us to obtain more refined results. This approach has been evaluated by considering both the multinomial and the gaussian version on the first layer clustering, and adopting the procedure for the dynamic estimation of the number of topics. Fig. 1(d) and Fig. 1(c) show the performances achieved by the two version and the ones achieved by a natural competitor based on latent factors: the regularized SVD. In both the cases, the hierarchical approach produces a significant improvement over the first clustering layer, outperforming the SVD model. On Netflix, hierarchical approach produces



(a) Latent factors on Netflix

(b) Latent factors on MovieLens

(c) Hierarchical model on Netflix

(d) Hieararchical model on Movielens

Fig. 1. Performance results.

RMSE values 0.9222 (multinomial model) and 0.9211 (gaussian model), while the best result achieved by the SVD model is 0.9275. This situation is also reflected in MovieLens where the Reg. SVD produced 0.9345. Again, it's a surprise to see that in this case the multinomial hierarchical approach (0.9274) outperforms the Gaussian hierarchical (0.9296). This result is even more surprising, if we consider that the multinomial user communities didn't perform very well in the first level. It seems that the adoption of specific item categories boosts the performance significantly.

The hierarchical model results competitive with some among the most popular and effective approaches for making recommendations, such as *PMF* [12], *Bi-LDA* [11] and *SVD++* [8]: the first one is reported to achieve on a sample of 1M ratings of Netflix data an RMSE equals to 0.9253; the latter achieves 0.9333 on the overall Netflix dataset. As far as the *SVD++* is concerned, although it achieves a 0.904 RMSE value on the considered dataset, the problem with such an approach is that it takes advantage of implicit information contained in the test-set. The model also compares with *fLDA* [2] and the *Regression-based latent factor models* [1], which integrate user/item features and on a 75% − 25% split of the MovieLens-1M achieve 0.9381 and 0.9258 RMSE values.

4 Conclusions and Future Works

In this work we proposed a probabilistic model for the discovery of both global and local patterns from users' preference data, which is based on a hierarchical approach. In the first phase we detect user communities, groups of users who tend to follow the same preference pattern, considering both rating assignments and purchased items; the information provided by each user community is then refined by applying 'local' topic analysis techniques.
Experimental evaluations on real data proved that both the User Community model (for the discovery of global patterns) and the hierarchical topic detection model exhibit prediction capabilities comparable to state-of-the art approaches.

The proposed approach is suitable for further investigations in several respects. Foremost, the proposed strategy can be combined with temporal/sequential information in order to better model user changes in preferences. Also, the proposed approach allows suitable integration of prior modeling and Bayesian estimation, for the "cold-start" issues.

References

1. Agarwal, D., Chen, B.C.: Regression-based latent factor models. In: KDD. pp. 19–28 (2009)
2. Agarwal, D., Chen, B.C.: flda: matrix factorization through latent dirichlet allocation. In: WSDM. pp. 91–100 (2010)
3. Arkadiusz, P.: Improving regularized singular value decomposition for collaborative filtering. In: SIGKDD. pp. 39–42 (2007)
4. George, T., Merugu, S.: A scalable collaborative filtering framework based on coclustering. In: ICDM. pp. 625–628 (2005)
5. Hofmann, T.: Collaborative filtering via gaussian probabilistic latent semantic analysis. In: SIGIR. pp. 259–266 (2003)
6. Jin, R., Si, L., Zhai, C.: A study of mixture models for collaborative filtering. Information Retrieval 9(3), 357–382 (2006)
7. Khoshneshin, M., Street, W.N.: Incremental collaborative filtering via evolutionary co-clustering. In: RecSys. pp. 325–328 (2010)
8. Koren, Y.: Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: KDD. pp. 426–434 (2008)
9. Marlin, B.: Modeling user rating profiles for collaborative filtering. In: NIPS (2003)
10. Marlin, B.: Collaborative Filtering: A Machine Learning Perspective. Master's thesis, Department of Computer Science, University of Toronto (2004)
11. Porteous, I., Bart, E., Welling, M.: Multi-hdp: a non parametric bayesian model for tensor factorization. In: AAAI. pp. 1487–1490 (2008)
12. Salakhutdinov, R., Mnih, A.: Probabilistic matrix factorization. In: NIPS, pp. 1257–1264 (2008)

Visualizing Information in Data Warehouses Reports

Michele Risi[1], Maria I. Sessa[1], Genoveffa Tortora[1], Maurizio Tucci[1],

[1] University of Salerno, via Ponte don Melillo,
84084 Fisciano, Salerno, Italy
{mrisi, misessa, tortora, mautuc}@unisa.it

Abstract. The display of information extracted from a data warehouse is an important aspect of human-machine interaction. Visualization tools play a central role in contexts where information must be represented by preserving both the accuracy of data, and the complexity of relationships between data. Much attention has been paid to the problem of effective visualization of data in individual reports that usually are viewed through different types of standard graph (Histogram, Pie, etc.) or in a tabular form. However, this kind of representation provides separate information items, but gives no support to visualize their relationships, which are the basis for most decision processes. This paper presents a methodology for information visualization by exploiting a visual language, called CoDe (Complexity Design), which allows users to manage the complexity of information to represent. In particular, CoDe provides an extensible set of graphical patterns, which allows to compose basic patterns to form complex ones; and allows the designer to specify the semantics of graphs by creating a logical link with the data they represent. The language visually represents both the items of information and their interrelationships at different levels of abstraction, keeping consistency between visually displayed items and the information contained in reports extracted from a data warehouse in tabular form by using the OLAP operations.

Keywords: Data Warehouses, Data Mining , Knowledge Representation, User Interfaces.

1 Introduction

Representation of scientific data or information obtained by scientific techniques of observation and processing, has been widely investigated and considerable studies have been aimed at graphic representation of data consistent with each other.

Often the representation of this information has been confined to specialist areas and each area has developed its own technical communications professionals, not always accessible to the general public. Less attention has been devoted to the visualization of complex scientific information or on scientific data variously interconnected with each other and heterogeneous.

Recently, visualization techniques of scientific information, are getting more attention, partly as a result of increased interest in the techniques of human-machine interaction that can access the input and output in an increasingly simple ways for human use. On the other hand, the enormous development of interaction and communication, which is an emerging characteristic of our times, requires a greater ability to acquire information quickly no longer isolated, but variously interrelated with each other and to build cognitive maps that summarize the semantics in order to develop decision-making. In particular, a wide set of tools are available to represent the information extracted from a data warehouse, visualizing tabular reports by means of standard graph as Histogram, Pie, etc. However, this kind of representation allows to represent information in separated reports, but no support is given to the visualization of their relationships. Thus, a cognitive map which summarizes the semantics of these relations, without losing details, could be a useful support for decision processes.

We propose an approach to the design of a user interface that allows to visualize relations between different reports in Excel format, by exploiting a graphic language named CoDe (Complexity Design) [5]. The representation obtained exploiting the visual language CoDe can be considered as a sort of high-level cognitive map of the complex information underlying the representation of the ground data. The choice of the final visualization shape (in terms of standard graphs) is left to the user interface, which is in charge of providing the necessary implementation constructs.

In particular, CoDe is composed by an extensible set of graphical patterns as lexical elements, which provides two families of operators: a first set of operators is supplied to compose basic visualization patterns to form complex ones; a second set of operators allow the designer to specify the semantics of graphs by creating a logical link with the data they represent. In fact, the information extracted from a data warehouse in tabular form using the OLAP operations, are visually represented by different graphs that are opportunely aggregated in order to simultaneously display the quantitative information of data and their interrelationships.

In Section 2, we discuss the related work. In Section 3 we describe the proposed methodology, whereas in Section 4 we illustrate the syntactic and semantic of the visual language CoDe. Finally, Section 5 will describe a modeling example and finally section 6 will conclude the paper with final remarks and future works.

2 Related Work

Several techniques and tools have been proposed in the literature to relate and display information extracted from a data warehouse, but to the best of our knowledge, no one allows to define graphically the semantic relationships between the data and consider visualizations that can involve more than one type of graph. In particular, these approaches usually provide standard graphical tools and do not allow to compose, aggregate and change at will the different visualizations.

Ma *et al.* [1][11] describe the design and the implementation of meteorological data warehouse by using Microsoft SQL Server. In particular, the proposed systems create process of meteorological data warehousing based on SQL Server Analysis Services and use SQL Server Reporting Services to design, execute, and manage multidimensional data reports. Moreover, the generating data reports are represented as folding tabular form related to a two-dimensional maps available through the browser, but the users cannot choice the type of visualization.

Hsu *et al.* [7] apply a clustering analysis on OLAP reports to automatically discover the grouping knowledge between different OLAP reports. In particular, the proposed approach highlights this knowledge by using a dendrogram/icicle plot representation and histograms. The data interrelationship is computed by using a data mining approach for finding the hidden rules inside the data. Respect our approach, the different reports cannot be composed and changed by the users.

Su *et al.* [12] describe a technical framework to produce a report system based on a three-layer calculating architecture which implements a metadata mapping, an ETL module and a data warehouse. In particular, the proposed technique is able to connect with several different data sources and uses a statistical calculation process to integrate them. Moreover, it allows to define and extend different report models, that are in tabular form or represented graphically. As the previous work, no multiple reports can be considered.

Wojciechowski *et al.* [14] present a web based data warehouse application designed and developed for data entry, data management as well as reporting purposes. The application was build using Oracle Application Express (OAE) technology. The proposed application implements two ways of reporting, report in a tubular form and report in a chart format.

In [8] is presented a data warehouse application that is able to generate summary reports as input data files for a data mining system to predict future student enrollment. The courseware is designed to build the data warehouse systematically using an incremental approach, and the aggregation to relate and report as chart the information data.

Bin *et al.* [2] discuss a novel and efficient information integration technology aiming at information isolated island, based on data warehouse theory and other existing systems. In particular, the approach realizes data analysis on the basis of integration through physical views and self-definition sampling strategies. The output is represented in tabular form.

Anfurrutia *et al.* [1] present a product-line approach to database reporting based on the predictability and similarity among reports. In particular, the approach exploits a feature model that provides an abstract and concise syntax for expressing commonality and variability in a database reporting. The data reports are provided textually. In fact, the data are extracted and manipulated as XML files, and the page layout and the style format of data are defined by an XSL formatter.

Other different techniques are based on the reporting functions (e.g. [6][9]). Reporting functions represent a valid tool in an analytical environment, needed to exploit the benefit of integrated data usually gathered in a data warehouse. Moreover, they extend the classical grouping and aggregation functions providing a column-based ordering, partitioning, and windowing mechanism. In particular, Lehner *et al.* [9] discuss the problem of deriving reporting function queries from materialized reporting function views. Moreover, in [6] they introduce materialized reporting function views and show how to rewrite queries with reporting functions as well as aggregation queries to this kind of materialized view. The data provided as output of the queries are represented as folding tabular or grid forms.

3 The proposed methodology

In this paper we focus on the design of a user interface that allows to compose and visualize information and data relations contained in reports extracted from a data warehouse in tabular form by using OLAP operations. The user interface modeling is performed by exploiting a graphic language named CoDe. In particular, CoDe allows to visually represents both the items of information and their interrelationships at different levels of abstraction, keeping consistency between items and the information contained in reports.

The representation obtained exploiting the visual language CoDe can be considered as a sort of high-level cognitive map of the complex information underlying the representation of the ground data. The choice of the final visualization shape (in terms of standard graphs) is left to the user interface, which is in charge of providing the necessary implementation constructs. In particular, these constructs allow to integrate and display different kind of visualization such as Histogram, Pie, etc. by scaling the quantitative information data between related items and preserving the relationship and the ratio between data extracted from a data warehouse.

In the following, we illustrate the proposed methodology process that is in charge to define the data and metadata extraction, to model the cognitive map generating the user interface, and finally to visualize the report.

3.1 The methodology process

The methodology process allows to extract data information from a data warehouse, model the cognitive map and display the final report as a single image. In particular, the process is composed of three subsequent phases: OLAP Operation, CoDe Modeling and Report Visualization, as shown in Fig. 1, where the rounded rectangles represent process phases, whilst the rectangles represent the intermediate artifacts produced at the end of each phase.

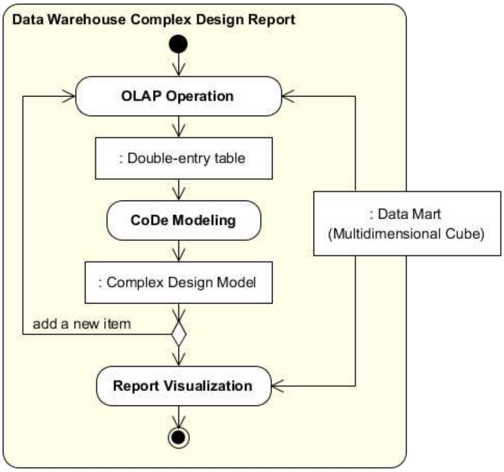


Fig. 1. The user interface development process.

The first phase, *OLAP Operation*, is in charge to dynamically generate reports from a data mart, represented as a multidimensional cube. In particular, these report can be organized as double-entry table, as represented in Fig. 2.

title_name		
C_1	…	C_n
$value_1$	…	$value_n$

Fig. 2. A double-entry table.

Following [3], we define graph the visualization of the knowledge carried out by a double-entry table. In particular, a graph, and the associated double-entry table, can be considered as a knowledge item. In fact, any graph must provide some ways to recognize the variational concepts involved in the represented information, given by the patterns in the corresponding double-entry table. In particular, title_name denotes the visualized information as a single consistent item, and components C_1, ..., C_n (represent concepts, objects and/or categories) with their associated values, that visualize specific information in the graph.

To provide a systematically approach for representing each OLAP report as a double-entry table, we define a pattern describing the OLAP operations to be performed. In particular, each consistent item in CoDe is defined applying this pattern to a specific multidimensional cube to properly extract item data with their interrelationships. It is worth noting that the OLAP operations pattern is executed virtually on the data mart by considering only the metadata information. The concrete execution of OLAP operations to compute data will be successively performed in the visualization phase.

The construction of the double-entry table from the multidimensional cube is based on: the set of measuring attributes (i.e. the components) for the report; the structure of the report that is divided into one horizontal axis and one or more vertical axes (i.e. the dimensions of the cube); and slicing/dicing/pivoting/rolling/drilling dimensional operators. These operators are data summarization/aggregation tools that help simplify multidimensional data analysis of a data warehouse. In particular, OLAP operators are useful in generating selection or aggregate values based on the combination of the selected cube dimensions.

The pattern used to compute a double-entry tables starting from the multidimensional cube is shown in Table 1.

Table 1. Pattern describing the OLAP operations to create the double-entry tables.

Graphical Pattern	**OLAP operations pattern**
Consistent item	*[pivoting][rolling(h) | drilling(h)][dicing(h)] [rolling(v)]*[slicing(v)]**

The label in the brackets represent a single OLAP operation, whilst the label *h* in the parenthesis indicates to perform the operation on the horizontal axis and the label *v* indicates to apply the operator on zero, one or more vertical axes (the multiplicity is expressed by the asterisk symbol). A careful reader can note that the same double-entry table can be computed by permuting the order of the operations in the defined pattern. As an example, Fig. 3 shows the application of the pattern to compute a consistent item on the multidimensional cube.

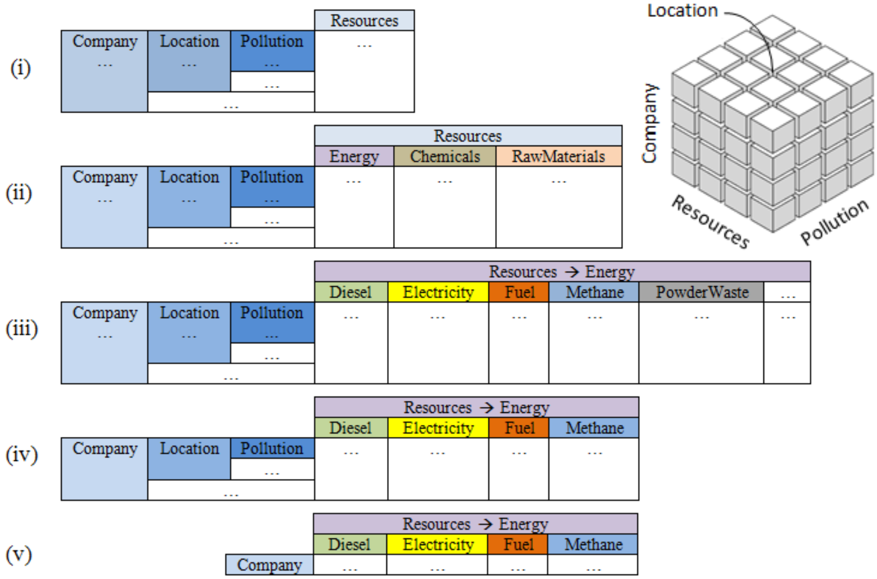


Fig. 3. OLAP operations on a multidimensional cube.

In particular, we start from the initial multidimensional cube as shown in the right hand part of Fig. 3. The multidimensional cube provides information about the production of companies located in Italy with respect to the resources employed and the produced pollution.

The first OLAP operation performed is *pivoting* which is used to rotate the cube in order to place the *Resources* dimension on the horizontal axis (see step (i)). The remaining dimensions of the cube are considered on the vertical axes. Then, a *drilling* operation is performed on the horizontal axis (i.e. Resources), in order to increase the details of the considered dimension (see steps (ii) and (iii)). Successively, a *dicing* operation on the horizontal axis is performed to select a subset of resources and exclude the other attributes including the *PowderWaste* attribute; and a *rolling* operation on the vertical axes (i.e. *Company*, *Location* and *Pollution*) to aggregate data and reduce the details (see step (iv)). Finally, a *slicing* operation on the vertical axes is performed to reduce the dimensions to *Company* (see step (v)) and produce the double-entry table.

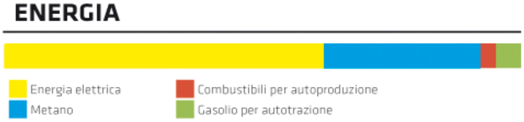
The *Code Modeling* phase takes as input the double-entry tables provided by the OLAP operation phase and allows the designer to edit the cognitive map exploiting the notation of the CoDe visual language. The output is a description of the Complex Design Model that is composed by a CoDe diagram and a lattice diagram describing the OLAP patterns needed to compute the data used in the model. The visual language CoDe is described in the next section and further details can be found in [5].

The first two step of the process can be repeated whenever the designer add a consistent item to the model. Finally, the *Report Visualization* phase, is in charge to display the Complex Design Model, by extracting data from the multidimensional cube through the OLAP operator patterns described in the lattice diagram. OLAP processing could be slow, thus the use of a lattice diagram allows to improve the performance reducing the total number of OLAP operation performed.

Then, the Report Visualization phase integrates, merges and displays different kind of graphs by scaling the quantitative information data between related items by preserving the relationship and the ratio between data extracted from the multidimensional cube. During this phase, the designer can place the graphs in specific locations of the drawing area and add some information labels. As an example, Fig. 4(a) shows the execution results of the OLAP operation pattern whereas Fig. 4(b) displays the correspondent Bar-Chart graph.

Products → Energy (Company)			
Diesel	Electricity	Fuel	Methane
762	5715	423	3768

(a)



(b)

Fig. 4. An double-entry table computed with an OLAP operation pattern and the final displayed graph.

4 The visual language CoDe

The formal definition of the graphic language CoDe is based on the idea that a visual representation of complex knowledge should be considered as a statement of a formal language in the first order Logic paradigm [5]. The formal definition of the language specifies both syntactic and semantics rules [12]. Syntactic rules are used to construct well formed visual representations, while semantic rules provide interpretations based on the related information. Thus, by an ontological point of view, a graphic visualization can be considered as made by terms, a sort of knowledge items represented by means of suitable graph types, and relations between these terms, that point out relationships and state semantics links to the data they represent. According with this paradigm, the CoDe language describes the architectural structure of the visualization at a meta-level, by means of terms and relations, which provide an abstract representation of the information that must be represented.

On the other hand, the same data and relationships can be represented in several different ways, depending essentially on an aesthetic choice of different possible graph types to represent knowledge items, or different styles in rendering the relationships. However, if the represented knowledge items, and the architectural relationships are the same, by translating these different visualizations in CoDe language, the equivalent meaning should be obtained as the same expression in this graphic meta-language of abstraction. The aim is to focus on the abstraction of knowledge items, and on the relations between them, provided by the architectural structure of the visualization, and not on the specific shapes selected for the corresponding graphs or relationship links. Indeed, although the shape of an Histogram, Pie, etc., can depend on different aesthetic factors, the meaning carried out is always the same. Thus, in order to define terms, a starting set of basic elements, representing items of information, must be considered. As an example, if an item of knowledge is a distribution of frequencies, it can be suitably represented by a basic standard like an Histogram. To increase the information carried out by a graph, additional visual elements can be defined, obtained using composition or transformation functions applied to the elements in the starting set. These functions are, in some sense, constructors of graphs that represent more complex information. Both basic and complex graphs are named terms, and are the elements of the domain of the language, as shown in Fig. 5. A semantic interpretation of a term is obtained by stating links to the represented data, according to the intended semantics of the graph types.

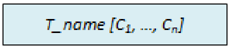


Fig. 5. A term of the CoDe language.

CoDe considers visualizations that can involve more than one graph. Indeed, by means of suitable functions, graphs can be composed or modified in order to visualize more complex knowledge. In the implementation process of ground data visualization, the starting set of terms is composed by standard-graphs (Histogram, Pie, Area, Bubble, Line, Radar, Bar-Chart, etc.), that can be exploited to visualize a double-entry table. However, any standard-graph has a fixed structural rule that state, by proportional magnitude, the semantics correspondence between the value of a component and its visualization on the plane. Moreover, the final implementation of a standard-graph depends by other visual parameters as: texture, color, orientation, shape, labels, etc.

CoDe also introduces some functions to compose graphs in order to represent more complex information. The visualizations carried out by applying these functions are themselves terms of the language. In general, a function is graphically visualized in CoDe as an oriented arrow connecting the involved terms. A label shows the symbol denoting the function itself. In the sequel we introduce definitions of some functions.

4.1 CoDe functions

The NEST function involves two terms such that one component of the first term is the title of the second one. The semantics of the NEST_i function is that the i-th component C_i in the first double-entry table can be described, at a deeper level, by means of the second table having C_i as title, as shown in Fig. 6.

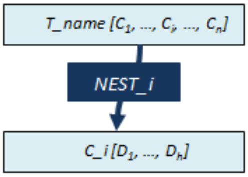
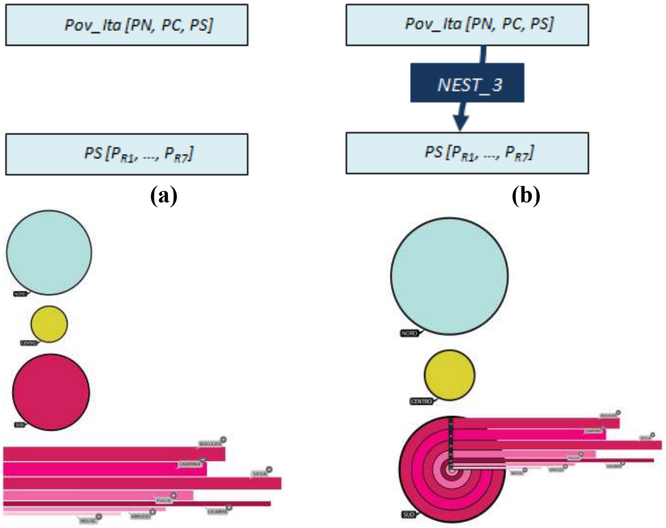


Fig. 6. The NEST function.

The considered tables can be visualized by the same or by different standard-graphs. For instance, let us consider as given terms shown in Fig. 7(a) representing two terms that provide, separately, the frequency distribution of poor people in the North, Center and South of the Italy, and frequency distribution of poor people in the

7 regions of South. Moreover, Fig. 7(c) shows the respective visualizations by means of standard-graphs Bubble and Histogram. For instance, by applying the NEST_3 function, we can construct a more complex information item which organizes, in a single structure, information given by the two separated terms. The new term, that represents this more complex information, is visualized in the Fig. 7(b). Finally, the Fig. 7(d) shows a visualization of the more complex term, obtained by suitably overlapping the two standard graphs Bubble and Histogram.

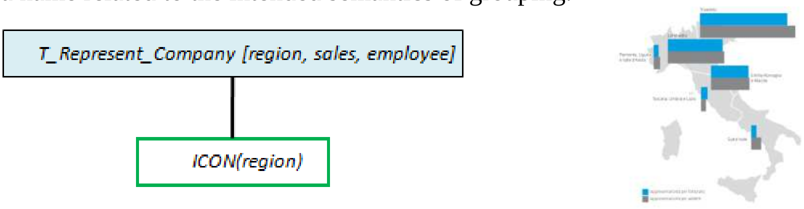


(a) (b)

(c) (d)

Fig. 7. A complex term obtained by using the NEST_3 function.

The aggregation function involves terms that have the same components. They are considered as a single term, but preserving their distinct identities. In CoDe, a term obtained by aggregation function is denoted by a dashed blue line including both the involved terms and the AGGR_name symbol, which denotes the applied function and a name related to the intended semantics of grouping.



(a) (b)

Fig. 8. The ICON operator.

Finally, another kind of functions that are exploited by the visualization interface are visualization operators, that involve only one term and do not concern its structure. Indeed, they only modify parameters of visualization of the involved term, to improve its aesthetic impact. As an example, the i-th component of a distribution of frequency values could be visualized by exploiting a metaphoric ICON, or, a metaphoric COLOR intensity that provides a proportional visualization of the related value. Moreover, a metaphoric TIME representation can be considered, when a time component is involved. Since CoDe manages these operators as external functions concerning facilities of the visualization interface, a different symbol will be used to provide the related graphical representation, as shown in Fig. 8(a).

As an example of the ICON operator, to visualize the representativeness of company employees and sales on the different regions of Italy; instead of using names of Italian regions to denote components, the related geographic shapes can be considered, as depicted in Fig. 8(b).

4.3 *Relations*

CoDe defines two kinds of relations: link, which states the existence of a logical relationship between terms, and set, which asserts the knowledge expressed by the data represented by a single term. A label can describe the intended meaning of the relation. The symbol used in CoDe to denote a link relation is a directed pink arrow, which connects the component terms. A label denotes the intended meaning of the relation, as shown in Fig. 9.

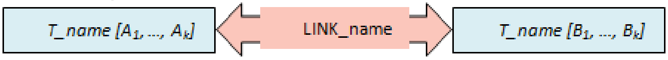


Fig. 9. The LINK relation.

For the monadic relation set, the symbol is a pink rounded rectangle containing the asserted term. It is useful to stress the difference between function and relation concepts. A function is a fixed construction tool, that carries out a new term starting from the involved terms. A relation is a predicative tool, expressing the existence or not of a fixed link between terms, then it provides a truth value, true or false, not a new term. Thus, a term is the abstraction of an objective data belonging to the complex system that must be visualized, whereas a relation is the abstraction of a logical relationships on data (in general, true or false). Relation visualized are generally intended as stating true statement. In order to state a relation between terms there is no constraint on their structure and the chosen relations depend from the subjective view of the visualization designer. In other words, the same data can be used with different relations to emphasize different informative needs. As a consequence, if the designer makes a wrong choice of the relationships to represent, the information carried out by the related visualization is useless or confusing.

Finally, let us observe that the CoDe language is a tool for the description of the architectural structure of visualization at different levels of abstraction. Starting from a more abstract level, a final visualization can be obtained through several refinements.

5 A modeling example

A prototype modeler of CoDe language was developed using Eclipse GEF/GMF [16][17]. Moreover, the Report Visualization exploits the powerful drawing tools provided by the Adobe Illustrator application [18], by generating textual scripts that

Illustrator interprets to draw the cognitive maps. An example of the described methodology application is the visualization of complex information realized by the DensityDesign group [15] regarding the multidimensional cube of Fig. 3, that is shown in Fig. 10. Moreover Fig. 11 shows the underling meta-information graphically represented in CoDe.

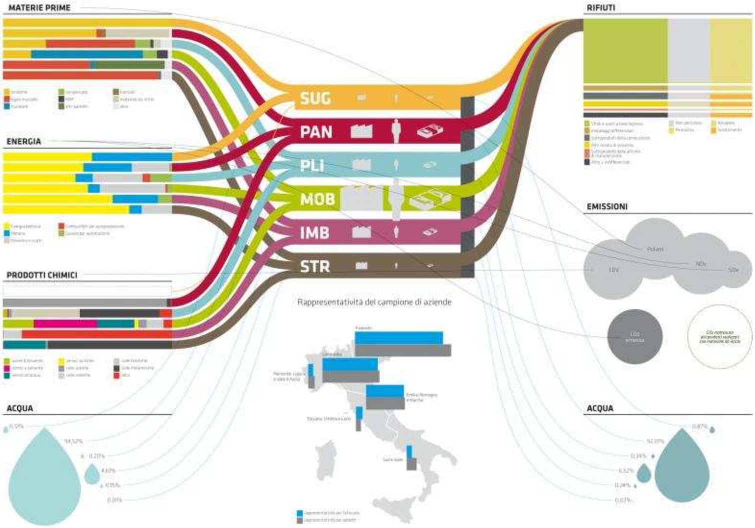


Fig. 10. Visualization of the report.

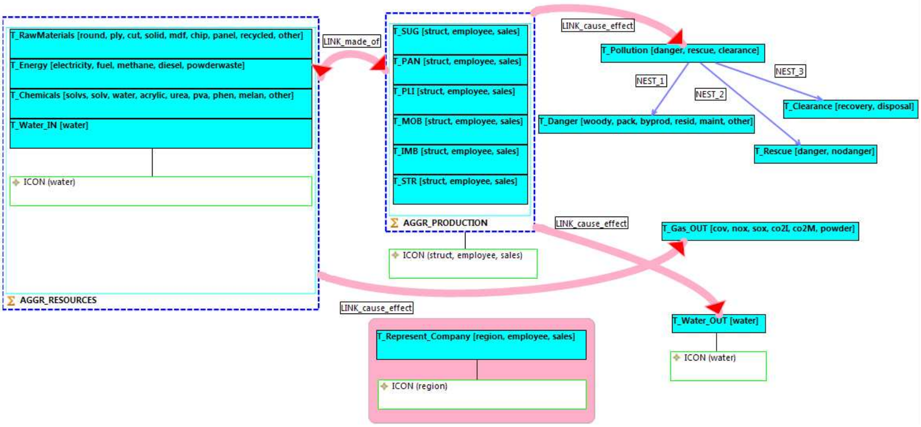

T_RawMaterials [round, ply, cut, solid, mdf, chip, panel, recycled, other]
T_Energy [electricity, fuel, methane, diesel, powderwaste]
T_Chemicals [solvs, solv, water, acrylic, urea, pva, phen, melan, other]
T_Water_IN [water]
ICON (water)
AGGR_RESOURCES
LINK_made_of
T_SUG [struct, employee, sales]
T_PAN [struct, employee, sales]
T_PLI [struct, employee, sales]
T_MOB [struct, employee, sales]
T_IMB [struct, employee, sales]
T_STR [struct, employee, sales]
AGGR_PRODUCTION
ICON (struct, employee, sales)
LINK_cause_effect
T_Pollution [danger, rescue, clearance]
NEST_1
NEST_2
NEST_3
T_Danger [woody, pack, byprod, resid, maint, other]
T_Clearance [recovery, disposal]
T_Rescue [danger, nodanger]
LINK_cause_effect
T_Gas_OUT [cov, nox, sox, co2I, co2M, powder]
LINK_cause_effect
T_Water_OUT [water]
ICON (water)
T_Represent_Company [region, employee, sales]
ICON (region)

Fig. 11. A CoDe model representing the productivity of Italian companies.

6 Concluding remarks and future works

Considerable studies have been directed to effective visual representation of single report in tabular form [3][4]. In many cases, it is not easy to rapidly acquire and interconnect information, because separated images require different stages of analysis. In this paper a graph composition methodology is presented, based on the definition of the graphic language CoDe. It allows to visualize information relationships in the same image, following the concept in term of visualization efficiency and completeness given in [3]. The language visually represents both the items of information and their interrelationships at different levels of abstraction, keeping consistency between visually displayed items and the information contained in reports extracted from a data warehouse in tabular form by using the OLAP operations.

In the future we plan to integrate in the CoDe language the OLAP operators as visual function for cube analysis. Moreover, we plan to conduct an empirical evaluation to assess the proposed methodology and the functionalities provided by the implemented tool and to evaluate the performance of CoDe via a suitable information visualization metrics (e.g. entropy-based).

References

1. Anfurrutia, F.I., Diaz, O., Trujillo, S.: A product-line approach to database reporting. Latin America Transactions, IEEE, vol.4, no.2, 2006, pp.70-76.
2. Bin, L.Y., Zhou, D.H.: The Design and Realization of the Universal Report System in the Power System Based on Physical View. International Conference on Web Information Systems and Mining (WISM), 2010, vol.1, pp. 404-408.
3. Bertin, J.: Semiology of Graphics: Diagrams, Networks, Maps. University of Wisconsin Press, 1983.
4. Card, S.K., Mackinlay, J., Shneiderman, B.: Readings in Information Visualization: Using Vision to Think. Academic Press, San Diego CA, 1999.
5. Ciuccarelli, P., Sessa, M. I., Tucci, M.: CoDe: a Graphic Language for complex system visualization. 7th Conference of the Italian Chapter of AIS, 2010.
6. Habich, D., Lehner, W., Just, M.: Materialized Views in the Presence of Reporting Functions. 18th International Conference on Scientific and Statistical Database Management (SSDBM'06), 2006. pp. 159-168.
7. Hsu, K.C., Ming-Zhong Li: Applying Clustering Analysis on Grouping Similar OLAP Reports. International Conference on Computer Engineering and Applications, 2010. pp. 417-423.
8. Kulkarni, M., Lu M., Zhang, D.: A case-based data warehousing courseware. IEEE International Conference on Information Reuse and Integration (IRI), 2010. pp. 245-248.
9. Lehner, W., Hümmer, W., Schlesinger, L.: Processing Reporting Function Views in a Data Warehouse Environment. International Conference on Data Engineering (ICDE'02), 2002. pp. 176-185.
10. Ma, N., Yuan, M., Bao, Y.,Jin, Z., Zhou, H.: The Design of Meteorological Data Warehouse and Multidimensional Data Report. International Conference on Information Technology and Computer Science (ITCS '10), 2010. IEEE Computer Society, pp. 280-283.
11. Ma, N., Zhai, Y., Bao, Y., Zhou, H.: Design of Meteorological Information Display System Based on Data Warehouse. International Conference on Management and Service Science (MASS), 2010, pp.1-4.
12. Russel, S., Norvig, P.: Artificial Intelligence: a modern approach, third ed., Prentice Hall, Upper Saddle River, NJ, 2009.
13. Su, H., Su, J.: A Study and Practice of Report System Techniques Based on Three-layer Calculating Architecture. Second International Workshop on Education Technology and Computer Science (ETCS), 2010. pp. 654-657.
14. Wojciechowski, T., Sakowicz, B., Makowski, D., Napieralski, A.: Transaction system with reporting capability in a web-based data warehouse application developed in Oracle Application Express. 10th International Conference on The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM 2009), 2009, pp. 273-276.
15. The DensityDesign group. http://www.densitydesign.org
16. The Graphical Editing Framework. http://www.eclipse.org/gef
17. The Graphical Modeling Framework. http://www.eclipse.org/gmf
18. The Adobe Illustrator. http://www.adobe.com/it/products/illustrator

Handling XML Updates in Distributed Contexts

Federico Cavalieri[1], Giovanna Guerrini[1], Marco Mesiti[2]

(1) Università di Genova, Italy - {cavalieri,guerrini}@disi.unige.it
(2) Università di Milano, Italy - mesiti@dico.unimi.it

Abstract. The evaluation of an XQuery Update expression generates a list of update primitives (named PUL - Pending Update List) to be applied on the original document. In distributed execution environments (like disconnected and cloud computing, collaborative editing and versioning) the ability to evaluate the expression on a server and to actually make the PUL effective on the original document on a different one is really relevant. PULs are thus exchanged between PUL producers and the PUL executor and operators are required to reason on them. In this paper we present a set of operators that can be exploited for this purpose and a system implementing them.

1 Introduction

Most of the works on XML updates and current implementations of the XQuery Update Facility [8] rely on the assumption that updates are executed right after and on the same server where the expression requesting them is evaluated. That is, an XQuery Update expression is evaluated identifying a list of updates to be executed on the document, the PUL (Pending Update List), which is then immediately applied. In [1, 2], we discussed the relevance of contexts (like disconnected and cloud computing, collaborative editing and versioning) in which this assumption does not hold. Thus, the process of expressing and requesting updates is decoupled from that of making them effective on documents. PULs can be exchanged among nodes and algorithms to handle them as parallel (i.e., to be integrated as a single transformation from the current document to a new one), or sequential (i.e., to be applied one after the other) update requests are required, as well as techniques to invert PUL effects (i.e., to identify another PUL that reverts the modifications of the first one). The ability of handling PULs and reasoning on their effects is thus crucial.

A PUL is an unordered list of simple tree update operations, each one targeted at a single node. Even if the list is unordered, in the language semantics a precedence among operations is defined (e.g., deletion follows all the other operations). For some insert operations, the insertion position is left open to implementation leading to a degree of non-determinism in update execution. Moreover, some operations (such as two renames) on the same node would result in a unclear semantics and thus are considered incompatible. A PUL containing incompatible operations is not applicable, in that its application produces a dynamic error.

Reasoning on XML updates has mostly been intended as static analysis of update expressions [4]. Dynamic update analysis is certainly easier, since more concrete information about data is available in PULs, but it also raises new interesting issues. A first one is node identification: since PULs have to be exchanged we cannot any longer rely on the local representation of a document. Another issue is document independence: the access to the document should be limited as much as possible by exploiting a labeling scheme [6] to store structural information on the document, which can then be incorporated in the PULs. The only approach to dynamic reasoning on XML updates we are aware of is [5], and it does not cope with these issues, since the proposed operator composes the effects of a sequence of locally executed PULs.

The specific PUL operators we consider are: PUL reduction, inversion, integration and aggregation. *Reduction* transforms a PUL with the aim of collapsing similar operations and removing operations whose effects are overridden by others. In addition, *deterministic reduction* produces a PUL whose semantics is deterministic, and a canonical form is proposed so that each PUL has a single canonical reduction. *Inversion* generates a PUL whose application reverts the effects of the original PUL. Apart from transaction processing, this operator is useful to restore a document to a previous revision or to undo certain modifications in the collaborative editing context. *Integration* produces a single PUL from a list of PULs specified on the same document containing their operations that do not conflict and a set of *conflicting operations* representing clashes among operations in the PULs. For solving conflicts, we propose the use of application-specific policies. Each producer may specify its own desiderata on the PULs it sends for execution (e.g., order matters, deletion means “I do not need it any more” rather than “It must be deleted”) and the executor may rely on these policies, as well as on its own policies (e.g., producers reputation) to reconcile conflicting updates. Finally, *aggregation* produces a single PUL from a list of PULs corresponding to their sequential execution.

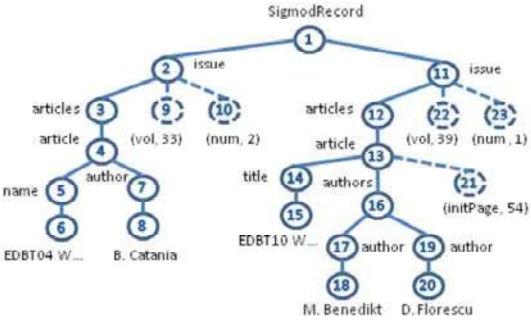
The paper is organized as follows. Section 2 introduces the notions our framework relies on, Section 3 presents the PUL operations, while Section 4 describes their implementation in a PUL handling system. Section 5 concludes.

2 Preliminaries

XML Document Representations. A document can be represented as a labeled tree D. Vertices V of the tree, represent element, attribute, or text nodes, according to the definition in [8]. Nodes are tagged by means of elements and attributes names (from the set $\mathcal{N}$) and values (from the set $\mathcal{V}$). A unique identifier, preserved upon modification, is associated with each node of D. In what follows, nodes and node identifiers are used interchangeably. Fig. 1.a shows the tree representation of a fragment of the `Sigmod-Record` document. Dotted lines are used to represent edges leading to attribute nodes and to point out that their relative order is not relevant.

Update Operations and PULs. Fig. 1.b reports the update primitives defined in [8]. In the figure $v \in V$ is a node, $P = [T_1, \ldots, T_k]$, $k \geq 0$, is a list (possibly empty in case of the `repN` or `repC` operation) of trees, $n \in \mathcal{N}$ is a name, $s \in \mathcal{V}$ is a value. The first parameter of each primitive specifies the affected node, that is, the operation *target*. An operation *op* is applicable on a document D if its target belongs to D and the applicability conditions (identified in [1]) of *op* hold.

A *pending update list* (*PUL*) [8] is an unordered list of operations among those in Fig. 1.b. Since the order of operations is irrelevant, some pairs of operations cannot occur in the same PUL. Specifically, no pairs of replacement operations of the same



a)

Operation	Description
$\mathtt{ins}^{\leftarrow}(v, P)$ $\mathtt{ins}^{\rightarrow}(v, P)$	Insert the trees in P before/after node v
$\mathtt{ins}^{\swarrow}(v, P)$ $\mathtt{ins}^{\searrow}(v, P)$	Insert the trees in P as first/last children of node v
$\mathtt{ins}^{\downarrow}(v, P)$	Inserts the trees in P as children of node v, in an implementation defined position
$\mathtt{insA}(v, P)$	Inserts the trees in P as attributes of node v
$\mathtt{del}(v)$	Deletes node v
$\mathtt{repN}(v, P)$	Replaces node v with the trees in P (possibly none)
$\mathtt{repV}(v, s)$	Replaces the value of node v with $s \in \mathcal{V}$
$\mathtt{repC}(v, v')$	Replaces the children of node v with text node v' or with nothing
$\mathtt{ren}(v, n)$	Renames node v with $n \in \mathcal{N}$

b)

Fig. 1. XML document tree-based representation (left) and XQuery Update operations (right)

type with the same target (referred to as *incompatible operations*) can occur. For a PUL to be applicable on a document (cf. function `applyUpdates` in [8]) it must contain no incompatible operations and all its operations must be applicable on the document.

The semantics of a PUL Δ on a document D is obtained by applying the operations in Δ in five stages [3]. The order of application of operations within each stage is not prescribed by [8]. Thus, when multiple insertion operations of the same type with the same target appear in the same PUL, the relative order of their inserted groups of nodes is not fixed as well. Moreover, operation $\mathtt{ins}^{\downarrow}$ is non-deterministic because the actual position of the inserted nodes in a document is not univocally specified. Therefore, the application of a PUL Δ to a document D produces a document in a set of possible outcomes, we refer to as *set of obtainable documents*, denoted as $\mathcal{O}(\Delta, D)$. The cardinality of $\mathcal{O}(\Delta, D)$ is thus greater than one when $\mathtt{ins}^{\downarrow}$ occurs in Δ or Δ contains more than one insertion operation of the same type on the same target.

Example 1. Let D be the document in Fig. 1.a. Operation $op_1 = \mathtt{del}(14)$ is deterministic and thus $\mathcal{O}(op_1, D)$ is a singleton. Operation $op_2 = \mathtt{ins}^{\downarrow}($16, <author>G.Guerrini</author>$)$, requires to inserting the element as first, second, or last author of the second paper, thus $\mathcal{O}(op_2, D)$ contains three documents. Finally, $|\mathcal{O}(\Delta, D)| = 6$ for $\Delta = \{\mathtt{ins}^{\downarrow}($16, <author> G.Guerrini</author>$)$, $\mathtt{ins}^{\searrow}($4, <initP>132</initP>$)$, $\mathtt{ins}^{\searrow}($4, <lastP>134</lastP>$)\}$. □

To reason on PULs, it is important to determine whether two PULs have the same effects, that is, they are equivalent, or one can be substituted by the other. This is determined by considering their set of obtainable documents. If $\mathcal{O}(\Delta_1, D) \subseteq \mathcal{O}(\Delta_2, D)$, we say that Δ_1 is substitutable to Δ_2.

3 Handling PULs

In this section we provide an overview of the proposed PUL handling operators. Details on their definitions, properties they ensure and their efficient implementation can be found in [1, 2].

PUL Reduction. Reducing the operations in a PUL Δ, by collapsing similar operations and removing operations whose effects are overridden, aims at obtaining a more compact PUL, denoted $\Delta^{\triangledown}$ which is substitutable to Δ. For this purpose, in [1] a set of *reduction* rules has been devised. Specifically, proposed rules eliminate operations overridden by a `repC`, `repN`, or `del` operation targeted at the same node or at an ancestor node, and collapses insertion and replacement operations targeted at the same node, or at sibling/parent/child nodes.

Example 2. Let Δ be a PUL specified on the document in Fig. 1.a consisting of the following operations: $\{$ $op_1 = \texttt{ins}^{\swarrow}$(4, <year>2004</year>), $op_2 = \texttt{ins}^{\rightarrow}$(7, <author>A.C.</author>), $op_3 = \texttt{ins}^{\searrow}$(4, <month>May</month>), $op_4 = \texttt{ren}$(5, 'title'), $op_5 = \texttt{ins}^{\leftarrow}$(5, <title>Rep...</title>), $op_6 = \texttt{ins}^{\rightarrow}$(7, <author>G.G.</author>), $op_7 = \texttt{ins}^{\rightarrow}$(7, <author>F.C.</author>), $op_8 = \texttt{repN}$(5, <author> M.M.</author>), $op_9 = \texttt{ins}^{\downarrow}$(16, <author>P.G.</author>) $\}$.

The reduction of Δ requires to remove the operation op_4 (overridden by op_8), to merge op_2, op_6, op_7 and op_3 in a single $\texttt{ins}^{\rightarrow}$ operation (op_{10}) and, finally, to merge op_1, op_5 and op_8 into a single `repN` operation (op_{11}). One of the possible outcomes of the reduction operator is, thus, $\Delta^{\triangledown} = \{$ $op_{11} = \texttt{repN}$(5, <year>2004</year>, <title>Rep...</title>, <author>M.M.</author>), $op_{10} = \texttt{ins}^{\rightarrow}$(7, <author>A.C.</author>, <author>G.G.</author>, <author>F.C. </author>, <month>May</month>), $op_9 = \texttt{ins}^{\downarrow}$(16, <author>P.G.</author>) $\}$. □

Even if a reduced PUL cannot contain insertions of the same type on the same node, it may still contain some $\texttt{ins}^{\downarrow}$ and thus have a non-deterministic semantics. A stronger form of reduction, named *deterministic reduction*, resulting in a PUL $\Delta^{\blacktriangledown}$ with deterministic semantics, is introduced. This is obtained by replacing any $\texttt{ins}^{\downarrow}$ operations still occurring in the reduced PUL with an $\texttt{ins}^{\swarrow}$ operation. Consider for instance the PUL $\Delta^{\triangledown}$ identified in Example 2. $\Delta^{\triangledown}$ is not deterministic because it contains a $\texttt{ins}^{\downarrow}$ operation. The deterministic reduction $\Delta^{\blacktriangledown} = \{$ $op_{11}, op_{10}, \texttt{ins}^{\swarrow}$(16, <a>P.G.</a>) $\}$ is obtained by transforming this operation in a $\texttt{ins}^{\swarrow}$.

Reduction rules do not guarantee a unique deterministic reduction. Indeed, more than one sequence of rule applications are possible for the same PUL leading to different possible results. A *canonical form*, that is, a "standardized" reduced representation, may be useful for reasoning on PULs. To determine such a unique reduction, we imposed a specific order on reduction rules application. Consider the PUL $\Delta^{\triangledown}$ identified in Example 2. The reduction operator might identify different PULs due to the unordered nature of PULs. For instance, the order of the nodes inserted by op_2, op_6 and op_7 is arbitrary in op_{10} and any permutations of these three nodes might be produced. The canonical form, instead, guarantees a unique reduction.

PUL Inversion. Given a PUL Δ on a document D, the inversion operator produces another PUL Δ^{-1}, that, when executed on any of the documents which can be obtained applying Δ on D, produces the original document D, thus undoing the updates in Δ. The inversion of an operation *op* applicable on a document D consists in the identification of one or more operations that revert the modifications specified by *op* on D on each document in $\mathcal{O}(op, D)$. `ins` is reverted by removing all inserted subtrees, `repV`/`ren` by restoring the original value/name of the target node, `repC` by restoring the original node children. For node deletions and replacements, a more complex behaviour is required:

the introduced nodes must be deleted, while removed nodes must be placed back in their original position. Attribute nodes are inserted by an `insA`, while other nodes through $\mathtt{ins}^{\rightarrow}$, if an adjacent left sibling exists in D, through an $\mathtt{ins}^{\swarrow}$ otherwise.

In inverting a PUL Δ the following properties must be guaranteed: (*i*) each inverse operation must be applicable in any of the obtainable documents; (*ii*) in case an operation in Δ is overridden (that is, it has no effect on the document), its inverse must have no effect; and, finally, (*iii*) the relative order of the nodes removed by Δ must be restored by its inverse. To guarantee the first two properties, overridden operations in Δ should be removed. To ensure the correct order, `repN` and `del` operations on adjacent siblings should be inverted together.

Example 3. Consider the PUL $\Delta = \{op_1 = \texttt{del}(5), op_2 = \texttt{repV}(6, \texttt{'VLDB04'}), op_3 = \texttt{repN}(7, \texttt{<author>} \texttt{X}^{25} \texttt{</author>}^{24}\ \}$ applicable on the document in Fig. 1.a. The overridden operation op_2 must not be inverted, while operations op_1 and op_3 must be considered together to ensure the correct order of restored nodes. The inverse of Δ is thus $\Delta^{-1} = \{\ \texttt{ins}^{\swarrow}(4, \texttt{<name>} \texttt{EDBT04 W...}^{6} \texttt{</name>}^{5} \texttt{<author>B.Catania}^{8} \texttt{</author>}^{7}), \texttt{del}(24)\}$. □

Parallel PULs: PUL Integration. Given two PULs Δ_1 and Δ_2 on the same document we aim at integrating them in a single PUL $\Delta_1 \odot \Delta_2$ that combines their effects. Integrating two PULs, however, is not always possible, due to different kinds of *conflicts* among the operations in the PULs, that do not allow to simply "put their operations together". Integration is, thus, a two stage operation: first conflicts are detected, then different conflict resolution policies are applied to reconcile the two PULs.

Intuitively, conflicts characterize the situations in which both PULs affect the same document nodes and cause a clash. Since we aim at detecting conflicting operations by inspecting the PULs, without computing their effects on the document, this leads us to identify the following conflicts between an operation of Δ_1 and one of Δ_2: *(i) incompatibility* according to the XQuery specification [8], that would prevent two operations with the same type and target to be in the same PUL or would result in an error when the PUL is executed (e.g. a repeated modification or a repeated insertion of the same attribute); (*ii*) *order-dependence*: operations whose effects are different depending on the order in which they are executed (e.g. two insertions of a first child into the same element); *(iii) overriding* of an operation by another, so that the first operation has no effect on the document due to the presence of the second one (e.g., the deletion of a node overrides a rename of that node).

The *integration* of two PULs can then be defined as a pair (Γ, Δ), where Γ is the set of all detected conflicts and Δ is the PUL of all non conflicting operations. Note that Δ coincides to the merge of the two PULs when no conflict arises.

Example 4. Let $\Delta_1 = \{op_1^1 = \texttt{ins}^{\rightarrow}(5, \texttt{<author>G.G.</author>}), op_1^2 = \texttt{repV}(9, \texttt{'3'}), op_1^3 = \texttt{repV}(21, \texttt{'5'})\}$ and $\Delta_2 = \{op_2^1 = \texttt{ins}^{\rightarrow}(5, \texttt{<author>A.C.</author>}), op_2^2 = \texttt{repV}(9, \texttt{'35'}), op_2^3 = \texttt{del}(13), op_2^4 = \texttt{del}(23)\}$ be two PULs to be integrated. The following conflicts can be identified between operations in the PUL Δ_1 and Δ_2: (i) op_1^1 and op_2^1 cause an order-dependence conflict since both the operations specify an insertion of the same type on the same target; (ii) op_1^2 and op_2^2 cause an incompatibility conflict since both require the same modification on the same target; finally, (iii) op_2^3 overrides op_1^3. Operation op_2^4 is not conflicted. □

The presence of conflicts may prevent PUL integration to maintain a semantics substitutable to that of a sequential application of the PULs. Therefore, this can lead to discard the PULs and asking the PUL producers to devise different modifications of the document. However, in some of the application contexts we have considered, it could be better to solve the conflicts even if this causes some operations to be discarded. An example is the data cloud context in which the PUL producers might not be available anymore. A simple solution would be to force, whenever possible, a sequential application of the PULs (i.e., PUL aggregation discussed in next section), but this blurs the semantics of parallel execution of PULs (by forcing a specific, arbitrary, application order) and may not be desirable in many contexts.

For this reason, we devise an architecture in which PUL producers can specify policies that give the possibility to the PUL executor to relax (or strictly enforce) some constraints specified in its PULs to guarantee that the different PULs on the same document can be reconciled. Moreover, PUL executors are equipped with a reconciliation algorithm, named *conflict resolution algorithm*, that, given a set of conflicts, relies on the policies of the involved PUL producers to find a solution to the identified conflicts that meets such policies (even if this means that some operations contained in one of the two PULs may be discarded), if any. The algorithm for reconciling PULs depends on the specific conflict resolution policies adopted by the executor, and on the producer policies associated with the PULs, and both of them are application-dependent.

Example 5. Consider the conflicts detected in Example 4. Suppose that the first producer requires that the inserted nodes must be present in the final document and that their position must not be altered. Then a possible resolution for the detected conflicts is to enforce the required ordering of inserted nodes by replacing op_1^1 and op_2^1 with a new operation $op = \mathtt{ins}^{\rightarrow}(5, \mathtt{<author>G.G.</author>}, \mathtt{<author>A.C.</author>})$, and excluding op_2^2 and op_2^3 from the integrated PUL. The resulting PUL is $\{\ op, op_1^3, op_2^1, op_2^4\ \}$. Note that, in case some of the producers policies cannot be met, the reconciliation would fail. □

Sequential PULs: PUL Aggregation. Given two PULs Δ_1 and Δ_2 their aggregation is a single PUL $\Delta_1 \multimap \Delta_2$ which cumulates the effects of their sequential executions $\Delta_1; \Delta_2$ in the specified order. Differently from integration where the two PULs refer to the original document, in aggregation Δ_2 can refer to the document updated through Δ_1. Moreover, there is no need of recurring to policies to reconcile conflicts, since the net result of the sequential application of two PULs is always well defined.

The aggregation of two PULs consists of their merge, provided that the operations in the second PULs do not depend on the operations specified in the first one. Otherwise, dependencies must be removed before merging the two PULs. Specifically, the position of nodes inserted by operations in Δ_2 depends on the position of a node inserted by an insert operation of the same type on the same node in Δ_1. Thus, to aggregate the PULs the effect of insert operations of the same type on the same node must be cumulated, so that the order of inserted nodes in the aggregated PUL is one of those obtainable by the sequential execution of Δ_1 and Δ_2. Additionally, whenever a node is target of a `repC`, `repV` or `ren` in both PULs, the operation in the first PUL must be removed to ensure that no incompatible operations occur. Finally, it might happen that a set of operations Δ' in Δ_2 target a node which is inserted by an operation *op* in Δ_1.

In this case, the operations in Δ' should be applied directly in the parameter of the operation op, because the XQuery Update semantics is snapshot-based and thus all target nodes must belong to the document on which the PUL is applied. Note, however, that in case the first PUL specifies a `repC` operation on a node v and the second PUL requires to insert new element nodes as children of v a more complex treatment is required, as discussed in [1].

Example 6. Let $\Delta_1 = \{\texttt{ins}^{\searrow}(3, \texttt{<article}^{24}\texttt{/>}), \texttt{ren}(5, \texttt{'title'}), \texttt{del}(23)\}$ and $\Delta_2 = \{\texttt{ren}(5, \texttt{'name'}), \texttt{ins}^{\downarrow}(24, \texttt{<title}^{25}\texttt{>XML}^{26}\texttt{</title>}), \texttt{ins}^{\searrow}(3, \texttt{<article}^{27}\texttt{/>})\}$ be two PULs to be aggregated, where for conciseness the node identifier is reported as superscript of the node. In this case, the aggregation of Δ_1 and Δ_2 must ensure that: (i) the order of the nodes inserted as last into node 3 by the evaluation of $\Delta_1 \multimap \Delta_2$ is the same as in the evaluation of $\Delta_1; \Delta_2$; (ii) only the rename operation in Δ_2 is present in $\Delta_1 \multimap \Delta_2$; (iii) the $\texttt{ins}^{\downarrow}$ operation in Δ_2, which is specified on a node inserted by Δ_1 must be merged with the operation inserting it. Thus, the aggregation of the two PULs is the following: $\{\texttt{del}(23), \texttt{ren}(5, \texttt{'name'}), \texttt{ins}^{\searrow}(3, \texttt{<article}^{24}\texttt{><title}^{25}\texttt{>XML}^{26}\texttt{</title></article><article}^{27}\texttt{/>})\}$. □

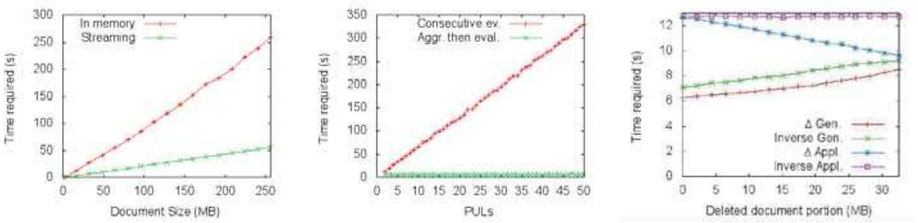
4 PUL Handler System

In our reference architecture, PUL execution is decoupled from PUL production. For this purpose, we have modified the Qizx [7] library to produce PULs and to accept PULs as input. When PULs have to be serialized and exchanged across the network, explicit identifiers need to be assigned to document nodes as node identification cannot rely on their in-memory representation. Such identifiers need to be unique in the document, immutable, not reusable and the identifiers generated by different producers must not clash. For what concerns the structural information our reasoning, with the exception of inversion, does not require accessing the document. Rather, to check whether the parent-child, element-attribute, left-right sibling, ancestor-descendant and preceder-follower structural relationship holds among two nodes (targets of operations in the PUL) a labeling scheme (e.g., [6]) is used. These relationships can be computed in constant time.

PULs are represented as XML documents containing the serialization of each PUL operation along with the identifiers and labels of the target nodes. Decoupling PUL production from PUL execution introduces additional costs in serializing and exchanging PULs on the network if compared to a scenario where PULs are executed locally and right after being produced. However, the PUL executor may take advantage of the knowledge of the exact nodes subjected to modification to execute PULs in streaming, without the need to access and load the whole document in main memory.

The computational costs and advantages of our operators and streaming evaluation has been experimentally assessed exploiting synthetic XQuery Update expressions and their corresponding PULs, generated by means of the modified Qizx library. PULs has been generated with uniform operation type distribution and have been applied on documents of various sizes, up to 256MB (labels and identifier excluded).

First we contrasted the time required for PUL execution through an adaptation of the Qizx library for directly handling PULs and a novel streaming evaluation algorithm. The former loads the entire document in memory, applies the PUL and finally serializes the document back to disk. The latter, instead, is based on a specialized SAX events



a) Streaming vs in memory PUL evaluation

b) Aggregation vs sequential evaluation

c) Inversion algorithm

Fig. 2. Experiments

transformer. As shown in Fig. 2.a streaming evaluation is up to three times faster on the considered documents. For what concerns the aggregation operator, we contrasted the time required for aggregating a list of PULs and the application of the obtained PUL with the a sequential application of each PUL in the list. Fig. 2.b demonstrates that aggregation may lead to a noteworthy advantage in terms of execution time. For what concern inversion, we considered a 64MB document and investigated the correlation between the size of the deleted document portion by a query and the time required for generating and applying the corresponding PUL/inverse PUL. PUL generation requires to load the entire document in memory (along with an id-to-node map in case of inverse PULs). Similarly, a PUL streaming application requires to scan the entire orignal document, load the PUL in memory and write back to disk the updated document. Thus, as shown in Fig. 2.c, PUL generation time is dominated by document size, while PUL application time by the size of the processed PUL and documents.

5 Conclusions

In this paper we presented an overview of our work on PULs reasoning. Some preliminary experiments have been reported that assess the utility of these operators. As future work we plan to investigate the effects of combining the operators together. Our experimental evaluation could be extended in other directions as well, explicitly taking distribution issues into account, as well as considering documents that are stored, and possibly shredded, in a DBMS.

References

1. F. Cavalieri, G. Guerrini, M. Mesiti. Dynamic Reasoning on XML Updates. In *EDBT*, pp. 165-176, 2011.
2. F. Cavalieri, G. Guerrini, M. Mesiti. Reverting the Effects of XQuery Update Expressions. In *BNCOD*, 2011. To appear.
3. M. Benedikt and J. Cheney. Semantics, Types and Effects for XML Updates. In *DBPL*, LNCS(5708), pp. 1–17, 2009.
4. M. Benedikt and J. Cheney. Static Analysis of Declarative Updates. In *EDBT/ICDT Workshops*, 2010.
5. G. Fourny, D. Florescu, D. Kossmann, M. Zacharioudakis. A Time Machine for XML: PUL Composition. In *XML Prague*, 2010.
6. L. Xu, T.W. Ling, H Wu, Z Bao. DDE: from dewey to a fully dynamic XML labeling scheme. In *SIGMOD*, pp. 719-730, 2009.
7. PIXwere. QIZX. An Open-source XQuery Processor, 2010.
8. W3C. XQuery Update Facility 1.0, June 2009.

SXPath: a Spatial Extension of XPath

Ermelinda Oro[1], Massimo Ruffolo[1], and Steffen Staab[2]

[1] High Performance Computing and Networking Institute,
Italian National Research Council
Via P. Bucci 41/C, Rende (CS), 87036, Italy
[2] Institute for Computer Science, University of Koblenz
Universitätsstraße 1, PO Box 201 602 56016. Koblenz, Germany
email: {oro,ruffolo}@icar.cnr.it
email: staab@uni-koblenz.de

Abstract. We report on a recently introduced extension of XPath, called SXPath, which is a new framework for querying Web documents by considering tree structures as well as spatial relationships between laid out elements. The underlying rationale is that frequently the rendering of tree structures is very involved and undergoing more frequent updates than the resulting layout structure. In this paper, we present the syntax and the semantics of the language that are based on a combination of a spatial algebra with formal descriptions of XPath navigation. Such language is intuitive and general enough to capture most frequent extraction patterns. Moreover, we show that the language maintains polynomial time combined complexity. Practical experiments demonstrate the usability of SXPath. This work is a short version of [11].

1 Introduction

Web designers plan Web pages contents in order to provide visual patterns that help human readers to make sense of document contents. This aspect is particularly evident in *Deep Web* pages [9], where designers always arrange data records and data items with visual regularity to meet the reading habits of humans. In the past, manual wrapper construction (e.g. [14]), or wrapper induction approaches (e.g. [4, 12, 16]) have exploited regularities in the underlying document structures, which led to such similar layout, to translate such information into relational or logical structures. However, surveying a large number of real (Deep) Web pages, we have observed that the document structure of current Web pages has become more complicted than ever implying a large conceptual gap between document structure and layout structure. Thus, it has become very difficult: (i) for human and applications aiming at manipulating Web contents (e.g. [5, 7, 14]), to query the Web by language such as XPath 1.0; (ii) for existing wrapper induction approaches (e.g. [12, 16]) to infer the regularity of the structure of deep Web pages by only analyzing the tag structure. Hence, the effectiveness of manual and automated wrapper construction are limited by the requirement to analyze HTML documents with increasing structural complexity. In the literature, approaches aimed at manipulating Web pages by leveraging the visual arrangement of page contents [7, 14], and frameworks for representing and querying multimedia and presentation databases [1, 8] have been proposed. However such approaches and frameworks provide limited capabilities in navigating and querying Web documents for information extraction purposes. Therefore, we propose to extend XPath 1.0 by new spatial navigation primitives, namely: (i) *Spatial Axes*, based on topological [13] and rectangular cardinal relations [10], that allow for selecting document nodes that have a specific spatial relation w.r.t. the context node. (ii) *Spatial Position Functions* that exploit some *spatial orderings* among document nodes, and allow for selecting nodes that are in a given spatial position w.r.t. the context node.

This paper is organized as follows. In Sec. 2 we describe SXPath data model, syntax, semantics and complexity issues. Sec. 3 reports notes on the implementation and results of experiments aimed at evaluating processing performances, SXPath usability, and qualitative enhancement in real applications. Sec. 4 concludes the paper. We have several results that, for space reason, we do not include here. We refer the reader to [11] for more details.

2 The SXPath Language

The SXPath language extends the W3C's XPath 1.0 [15] with spatial capabilities. Intuitive navigational features and querying capabilities of XPath 1.0 are central to most XML-related technologies. For this reason XPath 1.0 has attracted great attention in the computer science research community. The SXPath language adopts the path notation of XPath 1.0 augmented by a user-friendly syntax having a natural semantics that enables spatial querying. In this section, we first define the SXPath data model and describe its new spatial capabilities. Then we provide syntax, semantics, and complexity issues of the language.

2.1 Data Model

In this section we present the SXPath data model, namely *Spatial DOM* (SDOM). The SDOM considers relations existing among the visual representation of DOM nodes defined as follows.

Definition 1. *Let n be a node in the DOM of a Web page, the* minimum bounding rectangle *(MBR) of n is the minimum rectangle r that surrounds the contents of n and has sides parallel to the axes (x and y) of the Cartesian plane. The function $mbr(n)$ returns the rectangle r assigned to a DOM node by the layout engine of a Web browser. We call r_x and r_y the segments that are obtained as the projection of r on the x-axis and the y-axis respectively. Then, each side of the rectangle is represented by the segments (r_x^-, r_x^+) and (r_y^-, r_y^+), where r_x^- (resp. r_y^-) denote the* infimum *on the x-axis (y-axis) and r_x^+ (resp. r_y^+) denote the* supremum *on the x-axis (y-axis) of the segments r_x and r_y.*

Considering the function $mbr(n)$ given in Def. 1, a Web page can be modeled as a DOM enriched by spatial relations existing between MBRs. For representing such spatial relations we adopt the *Rectangle Algebra* (RA) qualitative spatial model [3], which allows for representing all possible relations between rectangles the sides of which are parallel to the axes of some orthogonal basis in a 2-dimensional Euclidean space. RA is a straightforward extension of the standard model for temporal reasoning, the *Interval Algebra* (IA) [2], to the 2-dimensional case. IA models the relative position between pairs of segments by a set of 13 atomic relations (R_{int}), namely *before* (b), *meet* (m), *overlap* (o), *start* (s), *during* (d), *finish* (f), together with theirs inverses $\{bi,\ mi,\ oi,\ si,\ di,\ fi\}$ and the relation *equal* (e). Let s and s_1 be two segments the IA relation $s\ b\ s_1$ represents that the segment s is preceded by the segment s_1. Let a and b be two rectangles, a RA relation between them is written as $a\ \rho\ b$ where $\rho = (\rho_x, \rho_y)$ is a pair of IA relations. The RA relation holds iff the IA relations $a_x\ \rho_x\ b_x$ and $a_y\ \rho_y\ b_y$ hold for segments that are obtained as projections of rectangle sides along x (i.e. $a_x,\ b_x$) and y (i.e. $a_y,\ b_y$) respectively. The expressiveness of RA covers the modeling of all qualitative spatial relations between two MBRs.

Definition 2. *SDOM is a node labeled sibling tree that provides orders among nodes, as described in [11], enriched by RA relations. It is described by the following 5-tuple:*

$$SDOM = \langle V, R_{\Downarrow}, R_{\Rightarrow}, A, f_s \rangle$$

where:

- *V is the set of labeled DOM nodes. $V = V_v \cup V_{nv}$, where V_v is the set of nodes visualized on screen, and V_{nv} is the set of nodes that are not visualized.*
- *$R_{\Downarrow}$ is the $firstchild$ relation. Let n and n' be two nodes in V, $nR_{\Downarrow}n'$ holds iff n' is the first child of n.*
- *$R_{\Rightarrow}$ is the $nextsibling$ relation. Let n and n' be two nodes in V, $nR_{\Rightarrow}n'$ holds iff n' is the next sibling of n.*
- *$A \subseteq V_v \times V_v$ is the set of arcs that represent spatial relations between pairs of nodes visualized on screen.*
- *Let R_{rec} be the set of RA relations, $f_s : A \rightarrow R_{rec}$ is the function that assigns to each element in A a RA relation in R_{rec}. So, let n and n' be two nodes in V_v, we have $a = (n, n') \in A$ holds iff $mbr(n)\ f_s(a)\ mbr(n')$.*

2.2 Spatial Axes

RA relations, stored in the SDOM, represent all qualitative spatial relations between MBRs, but they are too fine grained, verbose and not intuitive for querying. Therefore, for defining SXPath spatial axes we consider the more synthetic and intuitive *Rectangular Cardinal Relation* (RCR) [10] and *Rectangular Connection Calculus* (RCC) [13] models. In particular, RCRs express spatial axes that represent directional relations between MBRs. RCRs are computed by analyzing the 9 regions (cardinal tiles) formed by the projections of the sides of the reference MBR (i.e. r). The *atomic* RCRs are: *belongs to* (B), *South* (S), *SouthWest* (SW), *West* (W), *NorthWest* (NW), *North* (N), *NorthEast* (NE), *East* (E), and *SouthEast* (SE). Using the symbol ":" it is possible to express *conjunction* of atomic RCRs. Furthermore, the three relations inspired by the RCC calculus, namely: *contained* (CD), *container* (CR), and *equivalent* (EQ), allow for expressing spatial axes that represent topological relations between MBRs. For instance, $r\ CD\ r_2$ means that the rectangle r_2 is spatially contained in the rectangle r. Each spatial axes (expressed by a RCR or a topological relation) corresponds to a set of RA relations computed by means the *mapping function* μ [11].

Like in XPath 1.0, SXPath axes are interpreted binary relations $\chi \subseteq V \times V$. Let $self := \{\langle u, u\rangle | u \in V\}$ be the reflexive axis, remaining SXPath axes are partitioned in two sets: Δ_t and Δ_s. The set Δ_t contains *traditional* XPath 1.0 axes (*forward*, e.g. child, descendant, and *reverse*, e.g. parent, ancestor) that allow for navigating along the tree structure. They are encoded in terms of their *primitive* relations (i.e. *firstchild*, *nextsibling* and their inverses), as shown in [6]. The set Δ_s contains the novel (directional and topological) *spatial axes* corresponding to the RCRs and Topological Relations that allow for navigating along the spatial RA relations. In the following we formally define spatial axes in terms of their primitive RA relations stored in the SDOM.

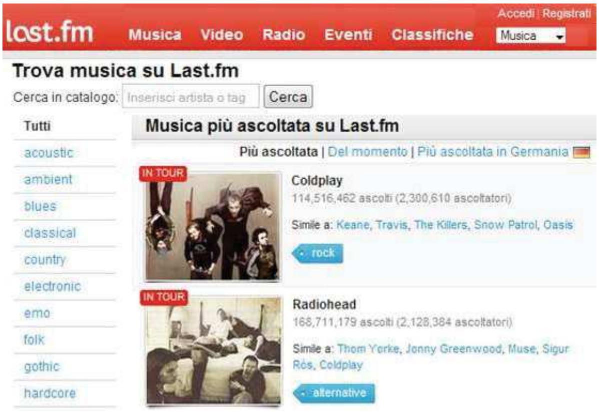
Definition 3. *SXPath spatial axes are interpreted binary relations* $\chi_s \subseteq V_v \times V_v$ *of the following form* $\chi_s = \{\langle u, w\rangle | u, w \in V_v \wedge mbr(u)\ \rho\ mbr(w) \wedge \rho \in \mu(R)\}$. *Here, R is the RCR or topological relation that names the spatial axis relation and μ is the mapping function.*

2.3 Syntax and Semantics

In this section we present the syntax of SXPath and give basic ideas which the language semantics is based on. Like XPath 1.0, the SXPath language allows for selecting sets of SDOM nodes by means of *expressions*. SXPath expressions have the same structure as the ones in XPath. SXPath extends XPath by means of: (i) A new set of *spatial axes* that can be used in location steps in the same way as traditional XPath axes. (ii) New node set functions, named *spatial position functions*, that allow for expressing predicates working on positions of nodes on the plane. These new spatial features enable spatial navigation and querying by exploiting spatial relations and spatial orders stored in the SDOM.

A SXPath location step has the following syntax $\chi :: t[p_1] \dots [p_n]$ where: (i) χ can be either a traditional XPath axis or a spatial axis. (ii) $t : \Lambda \cup \{\star, \texttt{text}\} \rightarrow 2^V$ is the node test function that returns the set of nodes that have a given label. Special labels $\star$ and `text` identifies all nodes ($t(\star) = V$) and text nodes respectively. (iii) Predicates can be based on spatial position functions. SXPath *expressions* return a value from one of the following types: *node set*, *number*, *string*, or *boolean*. Every expression evaluates relative to a *context* that extends the context of traditional XPath by considering spatial positions.

For instance, given the Web page Figure 1, a human reader can interpret the spatial proximity of images and nearby strings as a corresponding aggregation of information, namely as the complete record describing the details of a music band profile and its photo.



Trova musica su Last.fm

Cerca in catalogo: Inserisci artista o tag

Musica più ascoltata su Last.fm

Più ascoltata | Del momento | Più ascoltata in Germania

Coldplay
114,516,462 ascolti (2,300,610 ascoltatori)
Simile a: Keane, Travis, The Killers, Snow Patrol, Oasis

Radiohead
168,711,179 ascolti (2,128,384 ascoltatori)
Simile a: Thom Yorke, Jonny Greenwood, Muse, Sigur Rós, Coldplay

Fig. 1. A Page of the http://www.lastfm.it/ Web Site

The following XQuery exploits SXPath for extracting details of music bands by exploiting only the DOM nodes of type `img` and `text`, and their spatial relations.

Example 1. XQuery 1.0 and SXPath

```
for $img in document ("last-fm.htm")
1) /CD::img[N|S::img]   return
<music-band>
2)<name> {$img/E::text[posFromW()=1][posFromN()=1]} </name>
  <similar-bands>
3){$img/E::*[posFromW()=1][posFromN()=3][posSpatialNesting()=1]/CD::text}
  </similar-bands>
</music-band>
```

The *spatial location path* 1 returns images that form a vertical sequence. The spatial location path `$img/E::text` in pattern 2 and 3 returns nodes that lie on east (spatial axis `E`) of the context node represented by the variable `$img` (photos of music bands). Among these nodes the predicates select the name of the bands and its similar bands.

2.4 Complexity Issues

This section summarizes the computational complexity results of the SXPath query evaluation problem. We have considered two important fragments: (i) *Core SXPath* that is the navigational core of SXPath. It is obtained extending Core XPath [6] (the navigational core of XPath 1.0) by spatial axes introduced in Sec. 2.2. (ii) *Spatial Wadler Fragment* (SWF) that is the spatial extension of the *Extended Wadler Fragment* (EWF) [6]. It adds to Core XPath positional, logical and arithmetic features. Tab. 2.4 shows that *Core SXpath*, *Spatial Wadler Fragment* (SWF), and *Full SXpath* allow polynomial time combined complexity query evaluation with increasing degree of the polynomial. These results are compared with the fragment of XPath 1.0 that they extend. We denote by D the XML document, which has size $\Theta(|V|)$, where $|V|$ is the number of nodes of its SDOM representation. It is noteworthy that the SDOM (see Sec. 2.1) has size $O(|V|^2)$. $|Q|$ is the number of nodes of the parse tree for an input query Q.

	XPath 1.0		SXPath	
Space Time	Core[6]	$O(\lvert D\rvert * \lvert Q\rvert)$ $O(\lvert D\rvert * \lvert Q\rvert)$	Spatial Core	$O(\lvert D\rvert^2 * \lvert Q\rvert)$ $O(\lvert D\rvert^2 * \lvert Q\rvert)$
Space Time	EWF[6]	$O(\lvert D\rvert * \lvert Q\rvert^2)$ $O(\lvert D\rvert^2 * \lvert Q\rvert^2)$	SWF	$O(\lvert D\rvert^2 * \lvert Q\rvert^2)$ $O(max(\lvert D\rvert^3 * \lvert Q\rvert,\ \lvert D\rvert^2 * \lvert Q\rvert^2))$
Space Time	Full[6] Xpath 1.0	$O(\lvert D\rvert^2 * \lvert Q\rvert^2)$ $O(\lvert D\rvert^4 * \lvert Q\rvert^2)$	Full SXPath	$O(\lvert D\rvert^2 * \lvert Q\rvert^2)$ $O(\lvert D\rvert^4 * \lvert Q\rvert^2)$

Table 1. Comparison between complexity bound of SXPath and XPath 1.0 for a XML document D and a query Q.

3 Implementation and Experiments

We have implemented the language in a system that embeds the Mozilla browser[3] and computes the SDOM in real time at each variation of visualization parameters (i.e. screen resolution, browser window size, font type and dimension). In this way for each Web page and visualization condition there is a unique corresponding SDOM that enables the user to query the Web page by considering what s/he sees on the screen.

System Efficiency. For evaluating the SXPath system efficiency we have performed experiments that evidence the practical system behavior for both increasing document and query sizes. The runtime requirements for the SDOM construction and query evaluation is polynomial. For evaluating query efficiency we tested the whole system with fixed document sizes ($|D|$=1000, $|D|$=3000, $|D|$=6000) and increasing query sizes. Whereas for evaluating data efficiency of the query evaluator, we have used a fixed SWF query having size $|Q|$=167. Then, we have computed the needed query time for increasing documents sizes from $|D|$=50 to double the maximum size we found on real-world Web pages, i.e. $|D|$=7500 nodes. The obtained curves on log log scale have shown that time grows linearly with the query size (and the document size respectively) that indicate polynomial time, as described in [11].

Language Usability. Moreover we have perforned experiments aimed at assessing the usability of our approach and the enhancements provided by SXPath language over XPath 1.0. For the evaluation we have considered the situation of an expert user aiming at manually developing Web wrappers for Deep and Social Web sites. Experiments have involved ten users who where students well trained in XPath with no experience in SXPath. Each user visualizes and explores Web pages by using the SXPath system. In the experiments we have used a dataset of 125 pages obtained by collecting 5 pages per site from 25 Deep Web sites already exploited for testing wrapper learning approaches [5, 12, 16]. By carried out experiments we observed that:

1. Modifications in screen resolution and font type do not affect query results, whereas changes in browser window size and font dimension could affect the

[3] https://developer.mozilla.org/en/XULRunner_1.9.2_Release_Notes

query result. However, this aspect does not impact SXPath usability because the SXPath system: (i) Embeds the browser and computes the SDOM at each changing of visualization parameters. So, users can query what they see on the screen at each moment. (ii) SXPath queries are stored with visualization parameter settings adopted by the user during the query design process. Thus, when a query is reused on a Web page the embedded browser is set with visualization parameters for which the query has been designed.

2. The language was assessed as easy to learn and quite satisfactory to use.
3. The language is suitable for manual wrapper construction, giving the expert the possibility to look only at the visualized Web page, in comparison to XPath. In fact, by using 2 attempts on overage users were able to define a good SXPath query, whereas all the 5 available attempts were not enough for finding a good pure XPath query for all Web pages in the dataset.
4. Manually writing queries in pure XPath is more "complex" in comparison to SXPath because XPath requires the navigation of very intricate DOM structures, whereas SXPath mainly requires to look at the displayed page.
5. Even though the internal tag structure of various Web pages differ strongly (so different pure XPath queries are needed), all users have been able to use almost the same SXPath query for Web sites with similar visual arrangement. This experiment points out that SXPath allows for more general and abstract queries, that are independent from the internal structure of Web pages, in comparison to XPath.

Experiments provide a strong evidence for believing that humans aiming at manually defining Web wrappers and manipulating Web pages, may benefit from using SXPath navigation instead of pure XPath navigation. Moreover, the transportability of SXPath queries from one Web site to the next simplify manual definition of Web wrappers and can also support wrapper induction from sparsely annotated data, while the lack of such transportability observed for pure XPath is detrimental for both manual wrapper definition and wrapper induction. Details and rationales about all performed experiments are given in [11].

4 Conclusion and Future Work

In this paper, we have surveyed recent results about SXPath, the language that extends XPath to include spatial navigation into the query mechanism. We have used spatial algebras to define new navigational primitives and mapped them for query evaluation onto an extension of the XML document object model (DOM), i.e. the SDOM. Thus, we have given a formal model of the extended query language and have evaluated theoretical complexity. The theory has been implemented in a SXPath tool. Empirical evaluation have evidenced practical applicability of SXPath. The language can still be handled efficiently, yet it is easier to use and allows for more general queries than pure XPath. The exploitation of spatial relations among data items perceived from the visual rendering allows for shifting parts of the information extraction problem from low level internal tag structures to the more abstract levels of visual patterns. In the future research, we aim at making SXPath the query language for any Presentation Oriented Documents, such as PDF and PPT in addition to HTML. Moreover, we will investigate to the introduction of powerful construct in SXPath in order to automatically define Web wrappers based on visual features.

Acknowledgements

Work done by Steffen Staab was partially funded by German National Science Foundation (DFG) in the project 'Multipla'. Work done by Ermelinda Oro and Massimo Ruffolo was partially funded by Regione Calabria in the projects 'Voucher Ricercatori' and 'EasyPA - ID: 1220000277'.

References

1. S. Adali, M. L. Sapino, and V. S. Subrahmanian. An algebra for creating and querying multimedia presentations. *Multimedia Syst.*, 8(3):212–230, 2000.
2. J. F. Allen. Maintaining knowledge about temporal intervals. *Commun. ACM*, 26(11):832–843, 1983.
3. P. Balbiani, J.-F. Condotta, and L. F. d. Cerro. A new tractable subclass of the rectangle algebra. In *IJCAI*, pages 442–447, 1999.
4. C.-H. Chang, M. Kayed, M. R. Girgis, and K. F. Shaalan. A survey of web information extraction systems. *TKDE*, 18(10):1411–1428, 2006.
5. G. Gottlob, C. Koch, R. Baumgartner, M. Herzog, and S. Flesca. The lixto data extraction project: back and forth between theory and practice. In *PODS*, pages 1–12, 2004.
6. G. Gottlob, C. Koch, and R. Pichler. Efficient algorithms for processing xpath queries. *ACM Trans. Database Syst.*, 30(2):444–491, 2005.
7. J. Kong, K. Zhang, and X. Zeng. Spatial graph grammars for graphical user interfaces. *ACM Trans. Comput.-Hum. Interact.*, 13(2):268–307, 2006.
8. T. Lee, L. Sheng, T. Bozkaya, N. H. Balkir, Z. M. Özsoyoglu, and G. Özsoyoglu. Querying multimedia presentations based on content. *TKDE*, 11(3):361–385, 1999.
9. J. Madhavan, S. R. Jeffery, S. Cohen, X. . Dong, D. Ko, C. Yu, A. Halevy, and G. Inc. Web-scale data integration: You can only afford to pay as you go. In *CIDR*, 2007.
10. I. Navarrete and G. Sciavicco. Spatial reasoning with rectangular cardinal direction relations. In *ECAI*, pages 1–9, 2006.
11. E. Oro, M. Ruffolo, and S. Staab. Sxpath - extending xpath towards spatial querying on web documents. *PVLDB*, 4(2):129–140, 2010.
12. N. K. Papadakis, D. Skoutas, K. Raftopoulos, and T. A. Varvarigou. Stavies: A system for information extraction from unknown web data sources through automatic web wrapper generation using clustering techniques. *TKDE*, 17(12):1638–1652, 2005.
13. J. Renz. *Qualitative spatial reasoning with topological information*. Springer, 2002.
14. A. Sahuguet and F. Azavant. Building intelligent web applications using lightweight wrappers. *DKE*, 36(3):283–316, 2001.
15. W3C, http://www.w3.org/TR/xpath. *XML Path Language (XPath) Version 1.0*, 1.0 edition, November 1999.
16. Y. Zhai and B. Liu. Structured data extraction from the web based on partial tree alignment. *TKDE*, 18(12):1614–1628, 2006.

The NESTOR Model: Properties and Applications in the Context of Digital Archives

Nicola Ferro and Gianmaria Silvello

Department of Information Engineering, University of Padua, Italy
{ferro, silvello}@dei.unipd.it

Abstract. We present and describe the NEsted SeTs for Object hieRarchies (NESTOR) Model that allows us to model, manage, access and exchange hierarchically structured resources. The NESTOR Model is based on two set data models which can be put in relation with the tree data structure. We present these models highlighting their properties and the relationships with the tree.

We present a concrete use case based on archives that are fundamental and challenging entities in the digital libraries panorama. Within the archives we show how an archive can be represented through set data models, and how their properties can be used in this context; in particular, we focus of the problem of finding the lowest common ancestor.

1 Introduction

In Digital Libraries objects are often organized in hierarchies to help in representing, managing or browsing them. For instance, the documents in an archive are organized in a hierarchy divided into fonds, sub-fonds, series, sub-series and so on. Representing, managing, preserving and sharing efficiently and effectively the hierarchical structures is a key point for the development and the consolidation of Digital Library technology and services.

In this paper we provide a further analysis of the *NEsted SeTs for Object hieRarchies (NESTOR)* model which defines two set data models that we call: the "Nested Set Model (NS-M)" and the "Inverse Nested Set Model (INS-M)" [7]. These models are defined in the context of the ZFC (Zermelo-Fraenkel with the axiom of Choice) axiomatic set theory [3], exploiting the advantages of the use of sets in place of a tree structure. The foundational idea behind these set data models is that an opportune set organization can maintain all the features of a tree data structure with the addition of some new relevant functionalities. We define these functionalities in terms of flexibility of the model, rapid selection and isolation of easily specified subsets of data and extraction of only those data necessary to satisfy specific needs.

In this work we focus on the operations that we can perform in these set data models and we provide a use case in order to clarify the possible applications of the models as well as of their operations. In particular, we focus on the archives because they are one of the main organizations of interest for Digital Libraries;

they are a meaningful example of the need to support document management and access. The fundamental characteristic of archives resides in their internal hierarchical organization that constitutes a challenge for their representation, managing and, exchange as well as for their manipulation and querying.

In the presentation of the NESTOR Model, we concentrate on the INS-M and on how it can be used to model an archive and its resources. Furthermore, we analyze how we can define the lowest common ancestor in a hierarchy modeled by means of the INS-M. We highlight the problem of finding of the lowest common ancestor because it is an intrinsically beautiful and widely studied problem as well as a frequently performed operation in the archival context.

This paper is organized as follows: Section 2 introduces the background concepts on which this work is based; we introduce the basic set-theoretical concepts we are going to exploit and a brief definition of the tree data structure. Furthermore, we describe the basic principles of the archival practice and the standards to model and describe an archive in a digital environment. Section 3 presents the formal definition of the INS-M and proves the theorems defining how we can map a tree in the INS-M and vice versa. Moreover, we introduce a proposition showing the correlation between some operations in the tree and in the INS-M. Section 4 details how it is possible to define the lowest common ancestor in the INS-M. Section 5 presents a use case based on the archives where the INS-M properties are exploited; in particular, we explain how the INS-M has been adopted and exploited in the context of the SIAR (*Sistema Informativo Archivistico Regionale*) project. Lastly, in Section 6 we draw some final remarks.

2 Background

2.1 Set Theory: Collections of Subsets

We assume the reader to be confident with the basics of set theory that we cannot extensively treat here for space reasons [9]. The formal basis of this work is based on the concept of "Collection of subsets" that we introduce starting from the well-know concept of power set.

Let E be a set, we denote with $\mathcal{P}(E)$ the set containing all and only the subsets of E, that is, a set A belongs to $\mathcal{P}(E)$ if and only if it belongs to E. $\mathcal{P}(E)$ is called the **power set** of E. We understand that if E is a set, then there exists a set (collection) $\mathcal{P}$ such that if $A \subseteq E$, then $A \in \mathcal{P}$. The power set of a set E contains all the subsets of E, thus any collection of sets $\mathcal{C}$ composed by some subsets of E is a subcollection of the power set $\mathcal{P}(E)$, that is: $\mathcal{C}(E) \subseteq \mathcal{P}(E)$. Let us consider a set E and a collection of subsets $\mathcal{C}(E)$, we say that $\{H, K\} \in \mathcal{C}$ are incomparable, say $H||K$, is $H \nsubseteq K \wedge K \nsubseteq H$.

The following definition points out an important construction that we are going to exploit extensively in this work which is the *collection of proper subsets/supersets.*

Definition 1 *Let $\mathcal{C}$ be a collection of sets and $A \in \mathcal{C}$ be a set. We define $\mathcal{S}^+(A) = \{B \in \mathcal{C} \ : \ B \subset A\}$ to be the* ***collection of proper subsets*** *of A*

*in $\mathcal{C}$. We define $\mathcal{S}^{-}(A) = \{B \in \mathcal{C} \ : \ A \subset B\}$ to be the **collection of proper supersets** of A in $\mathcal{C}$.*

It is worthwhile for the rest of the work to introduce a formal definition of "family of subsets".

Definition 2 *Let A be a set, I a non-empty set and $\mathcal{C}$ a collection of sets of A. Then a function $\mathcal{A} : I \rightarrow \mathcal{C}$ is defined to be a **family** of subsets of A. We call I the **index** set and we say that the collection $\mathcal{C}$ is **indexed** by I.*

It is possible to use the extended notation $\{A_i\}_{i \in I}$ to indicate the family of subsets $\mathcal{A} : I \rightarrow \mathcal{C}$. The notation $A_i \in \{A_i\}_{i \in I}$ means that $\exists \ i \in I \mid \mathcal{A}(i) = A_i$. In the rest of the work to indicate a family of subsets $\mathcal{A} : I \rightarrow \mathcal{C}$ we will use the shorthand notation $\{\mathcal{A}_I\}$.

A frequently used concept is the one of **subfamily**: We indicate with $\{\mathcal{A}_J\}$ the subfamily of $\{\mathcal{A}_I\}$ defined as its **restriction** to $J \subseteq I$ and we say that $\{\mathcal{A}_J\} \subseteq \{\mathcal{A}_I\}$.

2.2 The Tree Data Structure

The most common and diffuse way to represent a hierarchy is the tree data structure, which is one of the most important non-linear data structures in computer science [11]. We define a tree as $T(V, E)$ where V is the set of nodes and E the set of edges connecting the nodes. V is composed by n nodes $V = \{v_1, \ldots, v_n\}$ and E is composed by $n-1$ edges. If $v_i, v_j \in V$ and if $e_{ij} \in E$ then e_{ij} is the edge connecting v_i to v_j, thus v_i is the parent of v_j. In this context it is convenient to talk about *inbound edges* and *outbound edges* of a node.

Definition 3 *Let $T = (V, E)$ be a rooted tree and $v_i \in V$ be a node of the tree, then we define its:*

Inbound set *to be* $E^{-}(v_i) = \{v_j \in V \mid e_{j,i} \in E\}$.
Outbound set *to be* $E^{+}(v_i) = \{v_j \in V \mid e_{i,j} \in E\}$.
Inbound degree *to be* $|E^{-}(v_i)|$[1].
Outbound degree *to be* $|E^{+}(v_i)|$.

We define with $\Gamma^{+}(v_i)$ the set of **all the descendants** of v_i in V (including v_i itself); vice versa $\Gamma^{-}(v_i)$ is the set of **all the ancestors** of v_i in V (including v_i ifself). We shall use the set Γ in the following of this work, so it is worth underlining a couple of recurrent cases. Let $v_r \in V$ be the root of a tree $T(V, E)$ then $\Gamma^{-}(v_r) = \{v_r\}$ and $\Gamma^{+}(v_r) = V$.

Furthermore, by means of this newly described notation, we can formally define the important concept of **lowest common ancestor**. The lowest common ancestor of nodes v_j and v_k in a tree is the ancestor of v_j and v_k that is located farthest from the root [2].

[1] For all nodes $v_i \in V$ such that $v_i \neq v_r$ where v_r is the root, $|E^{-}(v_i)| = 1$.

Definition 4 *Let $T(V, E)$ be a tree and $v_j, v_k \in V$ be two vertices. Then we define v_t to be the* ***lowest common ancestor*** *of v_j and v_k ($\mathsf{lca}(v_j, v_k) = v_t$) if:*

$$v_t \in \Gamma^-(v_j) \cap \Gamma^-(v_k), \textit{ and} \tag{2.1}$$

$$\nexists v_w \in V, w \neq t \mid (v_w \in \Gamma^-(v_j) \cap \Gamma^-(v_k)) \wedge (v_w \in \Gamma^+(v_t)) \tag{2.2}$$

The first condition imposes that $v_t = \mathsf{lca}(v_j, v_k)$ must be a common ancestor of v_j and v_k; the second condition says that cannot exist a vertex that is not v_t which is nearer than v_t to both v_j and v_k.

2.3 Archives

An archive represents the trace of the activities of a physical or juridical person in the course of their business which is preserved because of their continued value. Archives have to keep the context in which their records have been created and the network of relationships between them in order to preserve their informative content and provide understandable and useful information over time [8].

The context and the relationships between the documents are preserved thanks to the hierarchical organization of the documents inside the archive. Indeed, an archive is divided by fonds and then by sub-fonds and then by series and then by sub-series and so on – see Figure 1a for an example; at every level we can find documents belonging to a particular division of the archive or documents describing the nature of the considered level of the archive (e.g. a fond, a sub-fonds, etc.). The union of all these documents, the relationships and the context information permits the full informational power of the archival documents to be maintained. The archival documents are analyzed, organized, and recorded by means of the *archival descriptions* [12] that have to reflect the peculiarities of the archive [4].

2.4 Digital Archives and the NESTOR Model.

In the digital environment archival descriptions are encoded by the use of metadata; these need to be able to express and maintain the structure of the descriptions and their relationships [8].

The standard format of metadata for representing the hierarchical structure of the archive is the *Encoded Archival Description (EAD)* [13], which reflects the archival structure and holds relations between entities in an archive. In addition, EAD has a flexible structure, encourages archivists to use collective and multilevel description, and has a broad applicability. On the other hand, the EAD permissive data model may undermine the very interoperability it is intended to foster and it must meet stringent best practice guidelines to be shareable and searchable [15]. Furthermore, an archive is described by means of a unique EAD file and this may be problematic when we need to access and exchange archival metadata with a variable granularity [5] by means of DL standard technologies like the *Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH)*[2] [16].

[2] `http://www.openarchives.org/`

Several other modeling methodologies and metadata formats have been developed. Indeed, we may consider the "Tree-based Metadata" approach in which archives are described by a collection of lightweight metadata – e.g. Dublin Core Application Profiles[3] – one for each archival resource, connected one to the other by means of links to a third-party file – e.g. an external XML file – which maintains the archival structure [14]; alternative instantiations of this approach maintain the archival structure by means of an opportunely designed relational database [15]. Another possibility is to represent the archival structure by means of a collection of nested sets where each set represents an archival division and contains the metadata describing the resources belonging to that division [5]. This modeling methodology is based on the NESTOR Model which relies on two set data models called *Nested Set Model* (NS-M) and *Inverse Nested Set Model* (INS-M) [1]. Both these set data models, formally defined in the context of axiomatic set theory [10], can be used to model an archive by means of nested sets [7]. An extensive analysis of the NESTOR Model and its applications in the context of DL and archives can be found in [1]; in this paper we exploit the functionalities of the INS-M and thus we focus our presentation on this model.

The most intuitive way of understanding how the INS-M works is to see how a sample tree is mapped into an organization of nested sets based on the INS-M. We can say that a tree is mapped into the INS-M transforming each node into a set, where each parent node becomes a subset of the sets created from its children. The set created from the tree's root is the only set with no subsets and the root set is a proper subset of all the sets in the hierarchy. The leaves are the sets with no supersets and they are sets containing all the sets created from the nodes composing tree path from a leaf to the root. We can represent in a straightforward way the INS-M by means of the "*DocBall representation*" [17] – see Figure 1b. It is worthwhile to understand how the DocBall is used because the graphical tool we are going to present is based on this idea. The DocBall is composed of a set of circular sectors arranged in concentric rings; each ring represents a level of the hierarchy with the center representing the root. In a ring, the circular sectors represent the nodes in the corresponding level. We use the DocBall to represent the INS-M, thus for us each circular sector corresponds to a set; for instance, referring to Figure 1b, it is possible to say that section "Series C" is a direct superset of section "Sub-Fonds B".

It has been shown [7] that an archive can be modeled by means of the INS-M and than instantiated in such a way that allows the use of the OAI-PMH architecture to enable a variable granularity access and exchange of the archival metadata. Furthermore, in [5] it has been described a methodology to map an EAD file into the NESTOR Model preserving the full informative power of the metadata. Mapping an EAD file into the NESTOR Model means that we dispose of a methodology that maps the EAD structure into the INS-M and a collection of lightweight metadata containing the content information retained by EAD. In this way the INS-M preserves the archival structure and the metadata belonging to its sets preserve the content of archival descriptions [5]. In the same way, this

[3] `http://www.dublincore.org/`

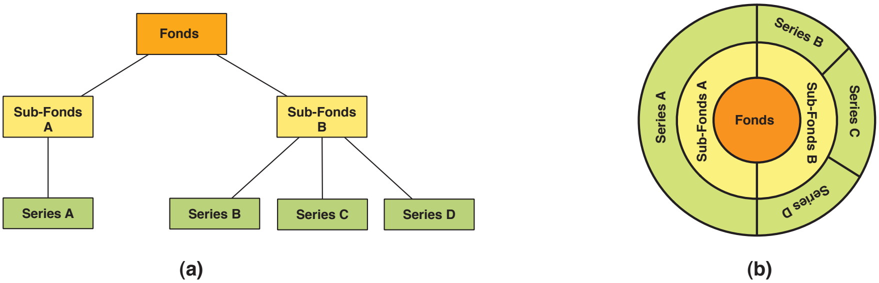


Fig. 1. The structure of a sample archive represented by: (a) a tree; (b) a Doc-Ball.

methodology is adopted with the "Tree-based metadata" approach, where the structure retained by an external XML file or by a relational database is mapped into the INS-M [1]. Thus, the INS-M can be used as a means to map archival metadata created by different systems in a common environment [5] as well as it can be adopted to model and describe an archive from scratch [7,1].

3 The Inverse Set Data Model and the Tree Data Structure

Now, we can define the Inverse Nested Set Model (INS-M):

Definition 5 *Let A be a set and let $\mathcal{C}$ be a collection. Then, $\mathcal{C}$ is an* ***Inverse Nested Set Collection*** *if:*

$$\exists! B \in \mathcal{C} \mid \forall K \in \mathcal{C}, B \subseteq K, \tag{3.1}$$

$$\forall H, K, L \in \mathcal{C} \mid H \subseteq K \wedge H || L \Rightarrow L \cap K = H \cap L. \tag{3.2}$$

Thus, we define an Inverse Nested Set Collection (INS-C) as a collection of subsets where two conditions must hold. The first condition (3.1) states that $\mathcal{C}$ must contain a bottom set, call it B, such that it is the common subset of all the sets in the collection. The second condition (3.2) states that if we consider two sets K and H such that H is a subset of K, then it cannot exist a set L incomparable to H, such that the intersection between H and L is not the same than the one between K and L.

Let us see a couple of examples regarding the set operations in the INS-M.

Example 1 *Let $\mathcal{C} = \{A, B, C\}$ be a INS-C, where $A = \{a, b\}$, $B = \{a, b, c, d\}$ and $C = \{a, b, c, d, e\}$.*

In this example $B \subseteq C$. Then, $B \cup C = \{a, b, c, d, e\} = C$, $B \cap C = \{a, b, c, d\} = B$ and $C \setminus B = \{e\} \notin \mathcal{C}$.

Example 2 *Let $\mathcal{C} = \{A, B, C\}$ be a INS-F, where $A = \{a, b\}$, $B = \{a, b, c, d\}$ and $C = \{a, b, e\}$.*

In this example $C||B$. Then, $B \cup C = \{a, b, c, d, e\} \notin \mathcal{C}$, $B \cap C = \{a, b\} = A \in \mathcal{C}$ and $B \setminus C = \{c, d\} \notin \mathcal{C}$.

We show how a tree can be mapped into a INS-C and vice versa. The following theorem formalizes the intuitive explanation about the mapping of a tree into a INS-C that we have given before. Basically, every couple of nodes v_j and v_k is mapped into a couple of sets J and K. If there exists an edge between v_j and v_k, say $e_{j,k}$ then the the set J created from v_j is defined as a subset of the set K created from v_k. The mapping between a tree and an INS-C reverses the idea described for the mapping of a tree into a NS-C; if a node is parent of another node in a tree, this is mapped into a set which is a subset of the set created from its child node. In Figure 2 we can see a tree mapped into the INS-M as defined by the next theorem.

Theorem 1 *Let $T = (V, E)$ be a tree and let $\mathcal{C}$ be a collection of subsets where $\forall v_i \in V, \exists! I = \Gamma^-(v_i)$. Then $\mathcal{C}$ is an INS-C.*

Proof. In order to prove this theorem let us consider a family of subsets $\mathcal{V}_V : V \to \mathcal{C}$ where the set of nodes V is its index set of the family and $\forall v_i \in V$, $V_{v_i} = \Gamma^-(v_i)$.

Let us prove condition 3.1 of Definition 5. Let $v_r \in V$ be the root of T. $\mathcal{V}_V(v_r) = V_{v_r} = \Gamma^-(v_r) = \{v_r\} \Rightarrow \forall v_j \in V, \Gamma^-(v_r) \subseteq \Gamma^-(v_j) \Rightarrow V_{v_r} \subseteq V_{v_j}$.

Let us prove condition 3.2 of Definition 5. Ab absurdo suppose that $\exists V_{v_k}, V_{v_h}, V_{v_l} \in \mathcal{V}_V \mid V_{v_h} \subseteq V_{v_k} \wedge V_{v_l}||V_{v_h} \Rightarrow V_{v_l} \cap V_{v_k} \neq V_{v_l} \cap V_{v_h}$.

This means that $\exists v_h, v_k, v_l \in V \mid \Gamma^-(v_h) \subseteq \Gamma^-(v_k) \wedge \Gamma^-(v_l)||\Gamma^-(v_h) \Rightarrow \Gamma^-(v_l) \cap \Gamma^-(v_k) \neq \Gamma^-(v_l) \cap \Gamma^-(v_h)$. $\exists v_j \in V \mid v_j \in (\Gamma^-(v_l) \cap \Gamma^-(v_k)) \wedge v_j \notin (\Gamma^-(v_l) \cap \Gamma^-(v_h)) \Rightarrow v_h \in \Gamma^-(v_k) \wedge v_j \in \Gamma^-(v_k) \wedge v_j \in \Gamma^-(v_l) \wedge v_j \notin \Gamma^-(v_h)$. This means that v_k and v_h must belong to the same branch of T; we know that $v_j \in \Gamma^-(v_l) \wedge v_j \in \Gamma^-(v_k)$, thus v_k and v_l must have v_j as a common ancestor and $v_j \notin \Gamma^-(v_h)$. This means that $\{v_j, v_k, v_l\} \in \Gamma^+(v_h)$ but $\Gamma^-(v_l)||\Gamma^-(v_h) \Rightarrow d_V^-(v_l) > 1 \Rightarrow T$ is not a tree.□

Now we can see how an INS-M is mapped into a tree; the following theorem shows that if we map every couple of sets A_j and A_k in an INS-F into a couple of nodes v_j and v_k in a set of nodes V such that there exists an edge $e_{j,k}$ in a set of edges E if and only if A_j is a direct subset of A_k then the graph defined by the nodes in V connected by the edges in E is a tree.

Theorem 2 *Let $\mathcal{C}$ be a INS-C, V be a set of nodes and E be a set of edges where $\forall v_j \in V, \exists! J \in \mathcal{C} \wedge \forall e_{j,k} \in E, \exists! J, K \in \mathcal{C} \mid J \subseteq K$. Then $T = (V, E)$ is a tree.*

Proof. We have to prove that $(\exists!\ v_r \in V \mid |E^-(v_r)| = 0) \wedge (\forall v_j \in V, j \neq r, |E^-(v_j)| = 1)$. Ab absurdo suppose that $\exists v_r, v_k \in V \mid (|E^-(v_r)| = 0 \wedge |E^-(v_k) = 0)| \vee \exists v_j \in V \mid |E^-(v_j)| > 1$.

If $\exists v_r, v_k \in V \mid |E^-(v_r)| = 0 \wedge |E^-(v_k)| = 0 \Rightarrow \exists J, K \in \mathcal{C} \mid \mathcal{S}^-(J) \cap \mathcal{S}^-(K) = \emptyset \Rightarrow \nexists B \in \mathcal{C} \mid B \subseteq J \wedge B \subseteq K \Rightarrow \mathcal{C}$ is not an INS-C.

If $\exists v_j \in V \mid |E^-(v_j)| > 1 \Rightarrow \exists J, K, L \in \mathcal{C} \mid K \subseteq J \wedge L \subseteq J \wedge K \cap L = \emptyset \Rightarrow L \cap K = \emptyset \neq L \cap J = L \Rightarrow \mathcal{C}$ is not an INS-C.□

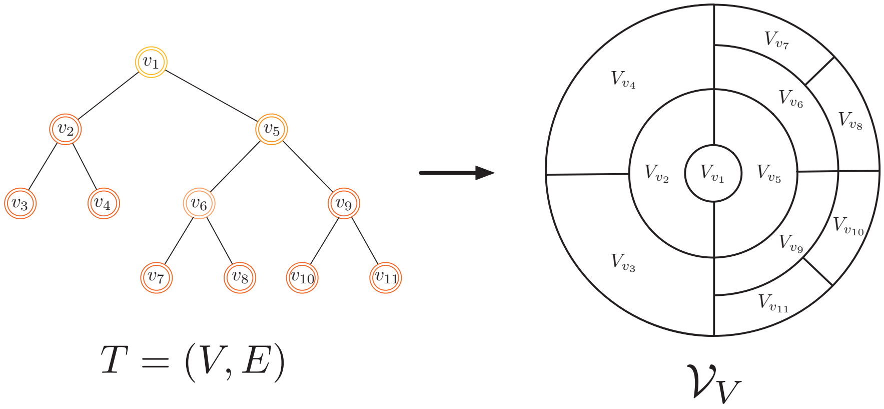


Fig. 2. A tree mapped into the INS-M.

The following proposition shows that the set-theoretic operations defined in the INS-M find a correspondent property in the tree.

Proposition 3 *Let $T = (V, E)$ be a tree, $\mathcal{C}$ be a INS-F mapped from T, $J, K, L \in \mathcal{C}$ be three sets and $v_j, v_k, v_L \in V$ be the three correspondent nodes in T. Then:*

$$J \cup K = K \Leftrightarrow v_k \in \Gamma^+(v_j) \tag{3.3}$$

$$J \cap K = J \Leftrightarrow v_j \in \Gamma^-(v_k) \tag{3.4}$$

$$J \cap K = L \Leftrightarrow v_l \in \Gamma^-(v_k) \cap \Gamma^-(v_j) \tag{3.5}$$

Proof. Property 3.3. Let us prove ($\Rightarrow$). Ab absurdo suppose that $J \cup K = K \Rightarrow v_k \notin \Gamma^+(v_j)$. This means that $J \notin \mathcal{S}^+(K) \Rightarrow J \nsubseteq K \Rightarrow J \cup K \neq K$.

Let us prove ($\Leftarrow$). Ab absurdo suppose that $v_k \in \Gamma^+(v_j) \Rightarrow J \cup K \neq K$. $J \cup K \neq K \Rightarrow J \nsubseteq K \Rightarrow \Gamma^-(v_j) \nsubseteq \Gamma^-(v_k) \Rightarrow v_k \notin \Gamma^+(v_j)$.

Property 3.4. The proof of this property is symmetric to the proof of Property 3.3.

Property 3.5. Let us prove ($\Rightarrow$). Ab absurdo suppose that $J \cap K = L \Rightarrow v_l \notin \Gamma^-(v_k) \cap \Gamma^-(v_j)$. This implies that $L \nsubseteq J \wedge L \nsubseteq K \Rightarrow L \notin \mathcal{S}^+(J) \wedge L \notin \mathcal{S}^+(K) \Rightarrow J \cap K \neq L$

Let us prove ($\Leftarrow$). Ab absurdo suppose that $v_l \in \Gamma^-(v_k) \cap \Gamma^-(v_j) \Rightarrow J \cap K \neq L$. This means that $L \nsubseteq J \wedge L \nsubseteq K \Rightarrow \Gamma^-(v_l) \nsubseteq \Gamma^-(v_k) \wedge \Gamma^-(v_l) \nsubseteq \Gamma^-(v_j) \Rightarrow v_l \notin \Gamma^-(v_k) \wedge v_l \notin \Gamma^-(v_j) \Rightarrow v_l \notin \Gamma^-(v_k) \cap \Gamma^-(v_j)$.□

Property 3.3 shows that if the union of two sets $\{J, K\} \in \mathcal{C}$ returns J it means that $v_j \in V$ is a descendant of $v_k \in V$; this property is a direct consequence of the definition of INS-F. Property 3.4 shows that if the intersection of two sets $\{J, K\} \in \mathcal{C}$ returns J, it means that $v_j \in V$ is an ancestor of $v_k \in V$.

Property 3.5 points out an interesting result: if the intersection of two sets $J, K \in \mathcal{C}$ returns a third set $L \in \mathcal{C}$, then this set corresponds to a common ancestor v_l of the nodes v_j and v_k.

4 The Lowest Common Ancestor in the INS-M

An important operation performed in the tree data structure is to determine the *lowest common ancestor* (lca) of two nodes. As a first thing let us define the lowest common ancestor in an INS-C.

Definition 6 *Let $\mathcal{C}$ be an INS-C, and $J, K, L \in \mathcal{C}$ be three sets. $L = J \cap K$ is defined to be the* ***lowest common ancestor*** *between J and K, say $\mathsf{lca}_{\mathcal{C}}(J, K) = L$.*

The relationship between the lca in a tree and in an INS-C can be easily determined by exploiting Theorem 1 which shows how to map a tree into an INS-C. Indeed, in the INS-M, the children of a node in a tree correspond to the supersets of the set mapped from that node in the INS-C mapped from the tree.

Proposition 4 *Let $T = (V, E)$ be a tree, $v_j, v_k, v_l \in V$ be three nodes, $\mathcal{C}$ be a INS-F mapped from T and $J, K, L \in \mathcal{C}$ be three sets. Then:*

$$v_l = \mathsf{lca}_V(v_j, v_k) \Leftrightarrow L = \mathsf{lca}_{\mathcal{C}}(J, K). \tag{4.1}$$

Proof. Let us prove ($\Rightarrow$). Ab absurdo suppose that $v_l = \mathsf{lca}_V(v_j, v_k) \Rightarrow L \neq J \cap K$. This implies that $L \nsubseteq J \vee L \nsubseteq K \vee (L \nsubseteq J \wedge L \nsubseteq K) \Rightarrow L \notin \mathcal{S}^+(J) \vee L \notin \mathcal{S}^+(K) \vee (L \notin \mathcal{S}^+(J) \wedge L \notin \mathcal{S}^+(K)) \Rightarrow v_l \notin \Gamma^-(v_j) \vee v_l \notin \Gamma^-(v_k) \vee (v_l \notin \Gamma^-(v_j) \wedge v_l \notin \Gamma^-(v_k)) \Rightarrow v_l \neq \mathsf{lca}_V(v_j, v_k)$.

Let us prove ($\Leftarrow$). Ab absurdo suppose that $L = J \cap K \Rightarrow v_l \neq \mathsf{lca}_V(v_j, v_k)$. This means that $(v_l \notin (\Gamma^-(v_j) \cap \Gamma^-(v_k))) \vee (\exists v_w \in V, v_w \neq v_l \mid (v_w \in (\Gamma^-(v_j) \cap \Gamma^-(v_k))) \wedge (v_w \in \Gamma^+(v_l)))$.

If $v_l \notin (\Gamma^-(v_j) \cap \Gamma^-(v_k)) \Rightarrow L \notin ((\mathcal{S}^+(J) \cup J) \cap (\mathcal{S}^+(K) \cup K)) \Rightarrow J \cap K \neq L$.

If $\exists v_m \in V, v_m \neq v_l \mid (v_m \in \Gamma^-(v_j) \cap \Gamma^-(v_k)) \wedge (v_m \in \Gamma^+(v_l)) \Rightarrow v_l \in \Gamma^-(v_m) \Rightarrow L \subset M \Rightarrow (M \subseteq J \cap K) \wedge (L \subseteq J \cap K) \wedge (M \in \mathcal{S}^-(L) \Rightarrow J \cap K = M$.□

This proposition shows that if we map a tree into a correspondent INS-C also the nodes of the tree are mapped into sets in the collection and thus the lca between two nodes is mapped into the lca between the correspondent sets. Furthermore, we can see that the lca between two sets in the INS-M can be determined *by the intersection of the considered sets.*

Example 3 *Let $T = (V, E)$ be a tree, and let $\mathcal{C}$ the INS-C mapped from T. In order to clearly understand the correspondence between the nodes of the tree and the sets of the collection, let us consider the family of subsets $\mathcal{V}_V : V \rightarrow \mathcal{C}$. If we consider the nodes v_7 and v_{11} the $\mathsf{lca}_V(v_7, v_{11}) = v_5$ because the path $v_7 P v_1$ intersected with the path $v_{11} P v_1$ returns two nodes: v_1 and v_5; v_1 is the root and by definition its depth is 0, instead v_5 has depth 1 thus, it is the lowest common ancestor between v_7 and v_{11}.*

We consider the sets V_{v_7} and $V_{v_{11}}$ in $\mathcal{V}_V$ represented in Figure 2; we can see that V_{v_1} is a common subset of both V_{v_7} and $V_{v_{11}}$ as well as V_{v_5}. But $V_{v_1} \subset V_{v_5}$. Furthermore, $V_{v_7} \cap V_{v_{11}} = V_{v_5}$ which correspond to the node $v_5 \in V$ of the tree.

From this example we can see the correspondence between $\mathsf{lca}_V(v_7, v_{11})$ in T and $\mathsf{lca}_{\mathcal{V}}(V_{v_7}, V_{v_{11}})$ in $\mathcal{V}_V$.

5 Use Case: Modeling an Archive through the INS-M

The tree data structure is adequate to represent the structure of an archive because it properly represents the hierarchical relationships between the archival divisions – see Figure 1a; on the other hand, in a tree it is not straightforward to represent the documents belonging to each archival division. We can say that the tree can represent the structural aspects of an archive but it needs to be somehow extended in order to represent also the content – i.e. the archival resources.

One of the main features of the NESTOR Model is the possibility to express both the hierarchical structure by means of the nested sets and the content by means of the elements belonging to the sets. By means of the NESTOR Model, the archival divisions are represented as nested sets and the hierarchical relationships are retained by their inclusion order. On the other hand, the archival resources are represented as elements belonging to the sets – please see Figure 1b. The INS-M allows us to straightforwardly represent an archive; from the Theorem 1, we know that a tree can be mapped into a INS-F and thus we know that its expressive power is preserved by the INS-M. In this case we can see that the INS-M allows us to define a further level of expressiveness respect to the tree. Furthermore, the INS-M is well-suited for the archival practice; indeed, the idea of "set" shapes the concept of archival division which is a "container" comprising distinct elements that have some properties in common.

The use of the INS-M to model the archives enables their resources to be accessed and shared with a variable granularity in a distributed environment [1]. This is eased by the straightforward integration of the INS-M with the standard de-facto for metadata exchange in distributed environment which is the OAI-PMH [7]. A consequence of the possibility of instantiate the representation of the archives by means of INS-M into OAI-PMH is the further integration of archives in the digital library systems. For these reasons we chose to adopt the NESTOR Model a basic brick of the SIAR (*Sistema Informativo Archivistico Regionale*) system. [6].

The SIAR is a project supported by the Italian Veneto Region which aim is to design and develop a Digital Archive System. The main goal of the SIAR is to develop a system for managing and sharing archive metadata in a distributed environment. Furthermore, another SIAR objective is to develop an information system able to create, manage, access, share and provide advanced services on archival metadata. The design and development of the SIAR system rely on the NESTOR Model; indeed, the INS-M is adopted to model and represent the archives and the archival resources. In this work we do not present the system in details but we focus on the use of the INS-M to perform frequently requested operations on the archives that in particular regard the manipulation and the querying of the archival structure and of the archival resources.

In this context we focus on the querying of the archival structure and resources; in particular, we have seen that the relationships between the archival documents are as important as the documents themselves, thus it is necessary to easily exploit these relationships to infer information from the documents. One on the most important operation is to define the correlation between two or more documents in the archive. The archivists have to be able to understand why two or more documents belong to the same archive and which is the document or the archival division that put them in relation. We can see that this operation can be modeled as a lowest common ancestor problem; indeed, two or more documents are in relation thanks to a common ancestor that connect them.

By means of the INS-M in the SIAR system we can infer the context of two archival documents without navigating the whole archival hierarchy. In fact, by means of the INS-M when we need to find out the common archival division which contains two or more archival documents we just need to intersect the sets containing the selected documents. The intersection of these sets returns one of their common superset; thanks to Proposition 3 we know it belongs to the INS-C representing the archive and thanks to Proposition 4 we know it is the lowest common ancestor. This property gives us a way to calculate the lowest common ancestor between two elements in a hierarchy – i.e. two documents in an archive – without taking into account the whole hierarchy but just the sets at which these elements belong. The lowest common ancestor represents a relevant case where we exploit the relationships between the tree data structure and the INS-M and the ratio between the operations in a tree and the operations in a INS-C mapped from it. Moreover, the formal basis we defined provides us with the necessary consistency to manipulate and query the archival resources modeled in the INS-M as well as we would do in the tree data structure. This fact allows us to be consistent with the other data models and systems adopted to handle the archives and archival resources.

6 Final Remarks

In this paper we presented the NESTOR Model focusing on the Inverse Nested Set Model and its properties detailing its formal definition and the relationships with the tree data structure. In particular, we define the problem of calculating the lowest common ancestor in the INS-M comparing it with the same problem in the tree. We presented a concrete use case based on the archive showing how it is possible to model an archive throughout the INS-M and to apply the presented properties to query the archival resources. The use case is described in the context of the SIAR project.

Acknowledgments

The work reported has been envisaged in the context of an agreement between the Italian Veneto Region and the University of Padua. EuropeanaConnect[4] (Contract ECP-2008-DILI-52800) and the PROMISE network of excellence[5] (contract n. 258191) projects, as part of the 7th Framework Program of the European Commission, have partially supported the reported work.

[4] http://www.europeanaconnect.eu/

[5] http://www.promise-noe.eu/

References

1. A. Agosti, N. Ferro, and G. Silvello. The NESTOR Framework: Manage, Access and Exchange Hierarchical Data Structures. In *Proceedings of the 18th Italian Symposium on Advanced Database Systems*, pages 242–253. Società Editrice Esculapio, Bologna, Italy, 2010.
2. M. A. Bender, M. Farach-colton, G. Pemmasani, S. Skiena, and P. Sumazin. Lowest Common Ancestors in Trees and Directed Acyclic Graphs. *J. Algorithms*, 57:75–94, 2005.
3. B. A. Davey and H. A. Priestley. *Introduction to Lattices and Order - 2nd Ed.* Cambridge University Press, Cambridge, UK, 2002.
4. L. Duranti. *Diplomatics: New Uses for an Old Science.* Society of American Archivists and Association of Canadian Archivists in association with Scarecrow Press, Lanham, Maryland, USA, 1998.
5. N. Ferro and G. Silvello. A Methodology for Sharing Archival Descriptive Metadata in a Distributed Environment. In *Proc. 12th Eur. Conf. on Research and Advanced Tech. for Digital Libraries*, pages 268–279. LNCS 5173, Springer, Germany, 2008.
6. N. Ferro and G. Silvello. Design and Development of the Data Model of a Distributed DLS Architecture for Archive Metadata. In 5^{th} *IRCDL - Italian Research Conference on Digital Libraries*, pages 12–21. DELOS: an Association for Digital Libraries, 2009.
7. N. Ferro and G. Silvello. The NESTOR Framework: How to Handle Hierarchical Data Structures. In *Proc. 13th Eur. Conf. on Research and Advanced Tech. for Digital Libraries*, pages 215–226. LNCS 5714, Springer, Germany, 2009.
8. A. J. Gilliland-Swetland. *Enduring Paradigm, New Opportunities: The Value of the Archival Perspective in the Digital Environment.* Council on Library and Information Resources, Washington, DC, USA, 2000.
9. P. R. Halmos. *Naive Set Theory.* D. Van Nostrand Company, Inc., New York, NY, USA, 1960.
10. Thomas Jech. *Set Theory.* Springer-Verlag, Berlin, Germany, 2003.
11. D. E. Knuth. *The Art of Computer Programming, third edition*, volume 1. Addison Wesley, Reading, MA, USA, 1997.
12. R. Pearce-Moses. *Glossary of Archival And Records Terminology.* Society of American Archivists, 2005.
13. D. V. Pitti. Encoded Archival Description. An Introduction and Overview. *D-Lib Magazine*, 5(11), 1999.
14. C. J. Prom and T. G. Habing. Using the Open Archives Initiative Protocols with EAD. In *Proc. 2nd ACM/IEEE Joint Conf. on Digital Libraries*, pages 171–180. ACM Press, USA, 2002.
15. C. J. Prom, C. A. Rishel, S. W. Schwartz, and K. J. Fox. A Unified Platform for Archival Description and Access. In *Proc. 7th ACM/IEEE Joint Conf. on Digital Libraries*, pages 157–166. ACM Press, USA, 2007.
16. H. Van de Sompel, C. Lagoze, M. Nelson, and S. Warner. The Open Archives Initiative Protocol for Metadata Harvesting (2nd ed.). Technical report, Open Archive Initiative, p. 24, 2003.
17. J. Vegas, F. Crestani, and P. de la Fuente. Context Representation for Web Search Results. *Journal of Information Science*, 33(1):77–94, 2007.

Inconsistency-tolerant Semantics for Description Logic Ontologies (extended abstract)[⋆]

Domenico Lembo, Maurizio Lenzerini, Riccardo Rosati,
Marco Ruzzi, and Domenico Fabio Savo

Dipartimento di Informatica e Sistemistica
Sapienza Università di Roma
lastname@dis.uniroma1.it

1 Introduction

It is well-known that inconsistency causes severe problems in classical logic. In particular, since an inconsistent logical theory has no model, it logically implies every formula, and, therefore, query answering on an inconsistent knowledge base becomes meaningless. In this paper, we address the problem of dealing with inconsistencies in Description Logic (DL) knowledge bases. Our goal is both to study DL semantical frameworks which are inconsistency-tolerant, and to devise techniques for answering queries posed to DL knowledge bases under such inconsistency-tolerant semantics.

A DL knowledge base is constituted by two components, called the TBox and the ABox, respectively. Intuitively, the TBox includes axioms sanctioning general properties of concepts and relations (such as Dog isa Animal), whereas the ABox contains axioms asserting properties of instances of concepts and relations (such as Bob is an instance of Dog). The various DLs differ in the language (set of constructs) used to express such axioms. We are particularly interested in using DLs for the so-called "ontology-based data access" [8] (ODBA), where a DL TBox acts as an ontology used to access a set of data sources. Since it is often the case that, in this setting, the size of the data at the sources largely exceeds the size of the ontology, DLs where query answering is tractable with respect to the size of the ABox have been studied recently. In this paper, we will consider DLs specifically tailored towards ODBA, in particular DLs of the *DL-Lite* family [8], where query answering can be done efficiently with respect to the size of the ABox.

Depending on the expressive power of the underlying language, the TBox alone might be inconsistent, or the TBox might be consistent, but the axioms in the ABox might contradict the axioms in the TBox. Since in ODBA the ontology is usually represented as a consistent TBox, whereas the data at the sources do not necessarily conform to the ontology, the latter situation is the one commonly occurring in practice. Therefore, our study is carried out under the assumption that the TBox is consistent, and inconsistency may arise between the ABox and the TBox (inconsistencies in the TBox are considered, e.g., in [5, 9]).

There are many approaches for devising inconsistency-tolerant inference systems [1], originated in different areas, including Logic, Artificial Intelligence, and Databases. Our work is especially inspired by the approaches to consistent query answering in databases [3], which are based on the idea of living with inconsistencies (i.e.,

[⋆] This paper is an extended abstract of [6].

data that do not satisfy the integrity constraints) in the database, but trying to obtain only consistent information during query answering. But how can one obtain consistent information from an inconsistent database? The main tool used for this purpose is the notion of database repair: a repair of a database contradicting a set of integrity constraints is a database obtained by applying a minimal set of changes which restore consistency. In general, there are many possible repairs for a database D, and, therefore, the approach sanctions that what is consistently true in D is simply what is true in all possible repairs of D. Thus, inconsistency-tolerant query answering amounts to compute the tuples that are answers to the query in all possible repairs.

In [7], a semantics for inconsistent knowledge bases expressed in *DL-Lite* has been proposed, based on the notion of repair. More specifically, an ABox $\mathcal{A}'$ is a repair of the knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is the TBox and $\mathcal{A}$ is the ABox, if $\mathcal{A}'$ is a maximal subset of $\mathcal{A}$ consistent with $\mathcal{T}$. In this paper, we call such semantics the *ABox Repair (AR) semantics*, and we show that for the DLs of the *DL-Lite* family, inconsistency-tolerant query answering under such a semantics is coNP-complete even for ground atomic queries, thus showing that inconsistency-tolerant instance checking is already intractable. For this reason, we propose a variant of the AR-semantics, based on the idea that inconsistency-tolerant query answering should be done by evaluating the query over the intersection of all AR-repairs. The new semantics, called the *Intersection ABox Repair (IAR) semantics*, is a sound approximation of the AR-semantics, and it enjoys a desirable property, namely that inconsistency-tolerant query answering is polynomially tractable.

Then, we highlight some drawbacks of the AR semantics, and propose a variant called the *Closed ABox Repair (CAR) semantics*, that essentially considers only repairs that are "closed" with respect to the knowledge represented by the TBox. We show that, while inconsistency-tolerant instance checking is tractable under this new semantics in *DL-Lite*, query answering is coNP-complete for unions of conjunctive queries. For this reason, we also study the "intersection-based" version of the CAR-semantics, called the *Intersection Closed ABox Repair ($ICAR$) semantics*, showing that it is a sound approximation of the CAR-semantics, and that inconsistency-tolerant query answering under this new semantics is again polynomially tractable.

2 Preliminaries

Description Logics (DLs) are logics that represent the domain of interest in terms of *concepts*, denoting sets of objects, *value-domains*, denoting sets of values, *attributes*, denoting binary relations between objects and values, and *roles*, denoting binary relations over objects. DL expressions are built starting from an alphabet Γ of symbols for atomic concepts, atomic value-domains, atomic attributes, atomic roles, and object and value constants. We denote by Γ_O the set of object constants, and by Γ_V the set of value constants. Complex expressions are constructed starting from atomic elements, and applying suitable constructs. Different DLs allow for different constructs.

A DL knowledge base (KB) is constituted by two main components: a *TBox* (i.e.,"Terminological Box"), which contains a set of universally quantified assertions stating general properties of concepts and roles, thus representing intensional knowledge of the domain, and an *ABox* (i.e.,"Assertional Box"), which is constituted by assertions on individual objects, thus specifying extensional knowledge. Again, different DLs allow for different kinds of TBox and/or ABox assertions.

Formally, if $\mathcal{L}$ is a DL, then an $\mathcal{L}$-knowledge base $\mathcal{K}$ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a TBox expressed in $\mathcal{L}$ and $\mathcal{A}$ is a ABox. In this paper we assume that the ABox assertions are *atomic*, i.e., they involve only atomic concepts, attributes and roles. The alphabet of $\mathcal{K}$, denoted by $\Gamma_{\mathcal{K}}$, is the set of symbols from Γ occurring in $\mathcal{T}$ and $\mathcal{A}$. The semantics of a DL knowledge base is given in terms of first-order (FOL) interpretations. We denote with $\mathit{Mod}(\mathcal{K})$ the set of models of $\mathcal{K}$, i.e., the set of FOL interpretations that satisfy all the assertions in $\mathcal{T}$ and $\mathcal{A}$, where the definition of satisfaction depends on the kind of expressions and assertions in the specific DL language in which $\mathcal{K}$ is specified. As usual, a KB $\mathcal{K}$ is said to be *satisfiable* if it admits at least one model, i.e., if $\mathit{Mod}(\mathcal{K}) \neq \emptyset$, and $\mathcal{K}$ is said to *entail* a First-Order Logic (FOL) sentence ϕ, denoted $\mathcal{K} \models \phi$, if $\phi^{\mathcal{I}} = \mathtt{true}$ for all $\mathcal{I} \in \mathit{Mod}(\mathcal{K})$. In the following, we are interested in particular in UCQ entailment, i.e., the problem of establishing whether a DL KB entails a *boolean union of conjunctive queries (UCQ)*, i.e., a first order sentence of the form $\exists \boldsymbol{y_1}. conj_1(\boldsymbol{y_1}) \vee \cdots \vee \exists \boldsymbol{y_n}. conj_n(\boldsymbol{y_n})$, where $\boldsymbol{y_1}, \ldots, \boldsymbol{y_n}$ are terms (i.e., constants or variables), and each $conj_i(\boldsymbol{y_i})$ is a conjunction of atoms of the form $A(z)$, $P(z, z')$ and $U(z, z')$ where A is an atomic concept, P is an atomic role and U is an atomic attribute, and z, z' are terms.

3 Inconsistency-tolerant semantics

In this section we present our inconsistency-tolerant semantics for DL knowledge bases. As we said in the introduction, we assume that for a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, $\mathcal{T}$ is satisfiable, whereas $\mathcal{A}$ may be inconsistent with $\mathcal{T}$, i.e., the set of models of $\mathcal{K}$ may be empty. The challenge is to provide semantic characterizations for $\mathcal{K}$, which are *inconsistency-tolerant*, i.e., they allow $\mathcal{K}$ to be interpreted with a non-empty set of models even in the case where it is unsatisfiable under the classical first-order semantics.

The inconsistency-tolerant semantics we give below are based on the notion of *repair*. Intuitively, given a DL KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, a repair $\mathcal{A}_R$ for $\mathcal{K}$ is an ABox such that the KB $\langle \mathcal{T}, \mathcal{A}_R \rangle$ is satisfiable under the first-order semantics, and $\mathcal{A}_R$ "minimally" differs from $\mathcal{A}$. Notice that in general not a single, but several repairs may exist, depending on the particular minimality criteria adopted. We consider here different notions of "minimality", which give rise to different inconsistency-tolerant semantics. In all cases, such semantics coincide with the classical first-order semantics when inconsistency does not come into play, i.e., when the KB is satisfiable under standard first-order semantics.

The first notion of repair that we consider can be phrased as follows: a repair $\mathcal{A}_R$ of a KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ is a maximal subset of $\mathcal{A}$ such that $\langle \mathcal{T}, \mathcal{A}_R \rangle$ is satisfiable under the first-order semantics, i.e., there does not exist another subset of $\mathcal{A}$ that strictly contains $\mathcal{A}_R$ and that is consistent with $\mathcal{T}$. Intuitively, each such repair is obtained by throwing away from $\mathcal{A}$ a minimal set of assertions to make it consistent with $\mathcal{T}$. In other words, adding to $\mathcal{A}_R$ another assertion of $\mathcal{A}$ would make the repair inconsistent with $\mathcal{T}$. The formal definition is given below.

Definition 1. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL KB. An* ABox Repair *(AR) of $\mathcal{K}$ is a set $\mathcal{A}'$ of membership assertions such that: (i) $\mathcal{A}' \subseteq \mathcal{A}$; (ii) $\mathit{Mod}(\langle \mathcal{T}, \mathcal{A}' \rangle) \neq \emptyset$; (iii) there exists no $\mathcal{A}''$ such that $\mathcal{A}' \subset \mathcal{A}'' \subseteq \mathcal{A}$ and $\mathit{Mod}(\langle \mathcal{T}, \mathcal{A}'' \rangle) \neq \emptyset$. The set of AR-repairs for $\mathcal{K}$ is denoted by AR-Rep$(\mathcal{K})$.*

Based on the above notion of repair, we now define ABox repair models.

Definition 2. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL KB. An interpretation $\mathcal{I}$ is an* ABox repair model*, or simply an AR-*model*, of $\mathcal{K}$ if there exists $\mathcal{A}' \in$ AR-Rep$(\mathcal{K})$ such that $\mathcal{I} \models \langle \mathcal{T}, \mathcal{A}' \rangle$. The set of ABox repair models of $\mathcal{K}$ is denoted by AR-Mod$(\mathcal{K})$.*

The following notion of consistent entailment is the natural generalization of classical entailment to the ABox repair semantics.

Definition 3. *Let $\mathcal{K}$ be a DL KB, and let ϕ be a first-order sentence. We say that ϕ is AR-*consistently entailed*, or simply AR-*entailed*, by $\mathcal{K}$, written $\mathcal{K} \models_{AR} \phi$, if $\mathcal{I} \models \phi$ for every $\mathcal{I} \in$ AR-Mod$(\mathcal{K})$.*

Example 1. Consider the *DL-Lite*$_{\mathcal{A}}$ knowledge base $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, where $\mathcal{T}$ contains the following assertions:

$$\begin{array}{lll} \mathsf{Mechanic} \sqsubseteq \mathsf{TeamMember} & \mathsf{Driver} \sqsubseteq \mathsf{TeamMember} & \mathsf{Driver} \sqsubseteq \neg\mathsf{Mechanic} \\ \exists\mathsf{drives} \sqsubseteq \mathsf{Driver} & \exists\mathsf{drives}^- \sqsubseteq \mathsf{Car} & (\mathsf{funct}\ \mathsf{drives}) \end{array}$$

Assertions from the first row, from left to right, respectively specify that drivers are team members, mechanics are team members, and drivers are not mechanics (disjointness). In the second row, first two assertions say that the role drives is specified between Driver (domain) and Car (range), and that it is functional, i.e., every driver can drive at most one car. $\mathcal{A}'$ is an ABox constituted by the set of assertions $\{\mathsf{Driver}(felipe)$, $\mathsf{Mechanic}(felipe), \mathsf{TeamMember}(felipe), \mathsf{drives}(felipe, ferrari)\}$. This ABox states that $felipe$ is a team member and that he is both a driver and a mechanic. Notice that this implies that $felipe$ drives $ferrari$ and that $ferrari$ is a car. It is easy to see that $\mathcal{K}$ is unsatisfiable, since $felipe$ violates the disjointness between driver and mechanic. The set *AR-Rep*$(\mathcal{K}')$ is constituted by the set of $\mathcal{T}$-$consistent$ ABoxes:

AR-rep_1= $\{\mathsf{Driver}(felipe), \mathsf{drives}(felipe, ferrari), \mathsf{TeamMember}(felipe)\}$;
AR-rep_2= $\{\mathsf{Mechanic}(felipe), \mathsf{TeamMember}(felipe)\}$.

The AR-semantics given above in fact coincides with the inconsistency-tolerant semantics for DL KBs presented in [7], and with the loosely-sound semantics studied in [2] in the context of inconsistent databases. Although this semantics can be considered to some extent the natural choice for the setting we are considering, since each ABox repair stays as close as possible to the original ABox, it has the characteristic to be dependent from the form of the knowledge base. Suppose that $\mathcal{K}'' = \langle \mathcal{T}, \mathcal{A}'' \rangle$ differs from the inconsistent knowledge base $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$, simply because $\mathcal{A}''$ includes assertions that logically follow, using $\mathcal{T}$, from a consistent subset of $\mathcal{A}$ (implying that $\mathcal{K}''$ is also inconsistent). One could argue that the repairs of $\mathcal{K}''$ and the repairs of $\mathcal{K}'$ should coincide. Conversely, the next example shows that, in the AR-semantics the two sets of repairs are generally different.

Example 2. Consider the KB $\mathcal{K}'' = \langle \mathcal{T}, \mathcal{A}'' \rangle$, where $\mathcal{T}$ is the same as in $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$ of Example 1, and the ABox $\mathcal{A}''$ is as follows:

$$\begin{aligned} \mathcal{A}'' = \{&\mathsf{Driver}(felipe), \mathsf{Mechanic}(felipe), \mathsf{TeamMember}(felipe), \mathsf{Car}(ferrari), \\ &\mathsf{drives}(felipe, ferrari)\}. \end{aligned}$$

Notice that $\mathcal{A}''$ can be obtained by adding $\mathsf{Car}(ferrari)$ to $\mathcal{A}'$. Since $\mathsf{Car}(ferrari)$ is entailed by the KB $\langle \mathcal{T}, \{\mathsf{drives}(felipe, ferrari)\}\rangle$, i.e., a KB constituted by the TBox $\mathcal{T}$ of $\mathcal{K}'$ and a subset of $\mathcal{A}'$ that is consistent with $\mathcal{T}$, one intuitively would expect that $\mathcal{K}'$ and $\mathcal{K}''$ have the same repairs under the AR-semantics. This is however not the case, since we have that *AR-Rep*$(\mathcal{K}'')$ is formed by:

AR-*rep*$_3$= $\{\mathsf{Driver}(felipe), \mathsf{drives}(felipe, ferrari), \mathsf{TeamMember}(felipe),$
$\mathsf{Car}(ferrari)\}$;
AR-*rep*$_4$= $\{\mathsf{Mechanic}(felipe), \mathsf{TeamMember}(felipe), \mathsf{Car}(ferrari)\}$.

Let us finally consider the ground sentence $\mathsf{Car}(ferrari)$. It is easy to see that $\mathsf{Car}(ferrari)$ is AR-entailed by the KB $\mathcal{K}''$ but it is not AR-entailed by the KB $\mathcal{K}'$.

Depending on the particular scenario, and the specific application at hand, the above behavior might be considered incorrect. This motivates the definition of a new semantics that does not present such a characteristic. According to this new semantics, that we call *Closed ABox Repair*, the repairs take into account not only the assertions explicitly included in the ABox, but also those that are implied, through the TBox, by at least one subset of the ABox that is consistent with the TBox.

To formalize the above idea, we need some preliminary definitions. Given a DL KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$, we denote with *HB*$(\mathcal{K})$ the *Herbrand Base of* $\mathcal{K}$, i.e. the set of ABox assertions that can be built over the alphabet of $\Gamma_{\mathcal{K}}$. Then we define the *consistent logical consequences of* $\mathcal{K}$ as the set *clc*$(\mathcal{K}) = \{\alpha \mid \alpha \in$ *HB*$(\mathcal{K})$ and there exists $S \subseteq \mathcal{A}$ such that *Mod*$(\langle \mathcal{T}, S\rangle) \neq \emptyset$ and $\langle \mathcal{T}, S\rangle \models \alpha\}$. With the above notions in place we can now give the definition of Closed ABox Repair.

Definition 4. *Let* $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ *be a DL KB. A* Closed ABox Repair (CAR) *for* $\mathcal{K}$ *is a set* $\mathcal{A}'$ *of membership assertions such that:* $(i)\mathcal{A}' \subseteq$ *clc*$(\mathcal{K})$*,* (ii)*Mod*$(\langle \mathcal{T}, \mathcal{A}'\rangle) \neq \emptyset$*,* (iii) *there exists no* $\mathcal{A}'' \subseteq$ *clc*$(\mathcal{K})$ *such that Mod*$(\langle \mathcal{T}, \mathcal{A}''\rangle) \neq \emptyset$ *The set of* CAR*-repairs for* $\mathcal{K}$ *is denoted by CAR-Rep*$(\mathcal{T}, \mathcal{A})$.

In words, a CAR-repair is a maximal subset of *clc*$(\mathcal{K})$ consistent with $\mathcal{T}$. The set of CAR-models of a KB $\mathcal{K}$, denoted *CAR-Mod*$(\mathcal{K})$, is defined analogously to AR-models (cf. Definition 2). Also, CAR-entailment, denoted $\models_{CAR}$, is analogous to AR-entailment (cf. Definition 3).

Example 3. Consider the two KBs $\mathcal{K}'$ and $\mathcal{K}''$ presented in the Example 1 and Example 2. It is easy to see that both *CAR-Rep*$(\mathcal{K}')$ and *CAR-Rep*$(\mathcal{K}'')$ are constituted by the two sets below:

CAR-*rep*$_1$=$\{\mathsf{Driver}(felipe), \mathsf{drives}(felipe, ferrari), \mathsf{TeamMember}(felipe),$
$\mathsf{Car}(ferrari)\}$;
CAR-*rep*$_2$=$\{\mathsf{Mechanic}(felipe), \mathsf{TeamMember}(felipe), \mathsf{Car}(ferrari)\}$.

It follows that both $\mathcal{K}'$ and $\mathcal{K}''$ CAR*-entail* the ground sentence $\mathsf{Car}(ferrari)$, differently from what happen under the AR-semantics, as showed in Example 2.

As we will see in the next section, entailment of a union of conjunctive queries from a KB $\mathcal{K}$ is intractable both under the AR-semantics and the CAR-semantics. Since this can be an obstacle in the practical use of such semantics, we introduce here approximations of the two semantics, under which we will show in the next section that entailment of unions of conjunctive queries is polynomial. In both cases, the approximation consists in taking as unique repair the intersection of the AR-repairs and of the CAR-repairs, respectively. This actually corresponds to follow the WIDTIO (When you are in doubt throw it out) approach, proposed in belief revision and update [10, 4].

Definition 5. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL KB. An* Intersection ABox Repair (IAR) *for $\mathcal{K}$ is the set $\mathcal{A}'$ of membership assertions such that $\mathcal{A}' = \bigcap_{\mathcal{A}_i \in \textit{AR-Rep}(\mathcal{K})} \mathcal{A}_i\}$. The (singleton) set of IAR-repairs for $\mathcal{K}$ is denoted by IAR-Rep$(\mathcal{K})$.*

Analogously, we give below the definition of Intersection Closed ABox Repair.

Definition 6. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL KB. An* Intersection Closed ABox Repair ($ICAR$) *for $\mathcal{K}$ is the set $\mathcal{A}'$ of membership assertions such that $\mathcal{A}' = \bigcap_{\mathcal{A}_i \in \textit{CAR-Rep}(\mathcal{K})} \mathcal{A}_i\}$. The (singleton) set of $ICAR$-repairs for $\mathcal{K}$ is denoted by ICAR-Rep$(\mathcal{K})$.*

The sets *IAR-Mod*$(\mathcal{K})$ and *ICAR-Mod*$(\mathcal{K})$ of IAR-models and $ICAR$-models, respectively, and the notions of IAR-entailment and $ICAR$-entailment are defined as usual (cf. Definition 2 and Definition 3). Consider for example the KB $\mathcal{K}' = \langle \mathcal{T}, \mathcal{A}' \rangle$ presented in Example 1. Then *IAR-Rep*$(\mathcal{K}')$ is the singleton formed by the ABox IAR-rep = AR-$rep_1 \cap AR$-$rep_2 = \{\mathsf{TeamMember}(felipe)\}$. In turn, referring to Example 3, *ICAR-Rep*$(\mathcal{K}')$ is the singleton formed by the ABox $ICAR$-rep_1 = CAR-$rep_1 \cap CAR$-rep_2 = $\{\mathsf{TeamMember}$(felipe),Car(ferrari)$\}$. It is not difficult to show that the IAR-semantics is a sound approximation of the AR-semantics, and that the $ICAR$-semantics is a sound approximation of the CAR-semantics. It is also easy to see that the converse is not true in general. For instance, the sentence $\mathsf{Driver}(felipe)$ is entailed by $\mathcal{K} = \langle \mathcal{T}, \{\mathsf{drives}(felipe, ferrari), \mathsf{drives}(felipe, mcLaren)\}\rangle$, where $\mathcal{T}$ is the TBox of Example 1, under the AR-semantics, but it is not entailed under the IAR-semantics. It can also be proved that the IAR-semantics is a sound approximation of the $ICAR$-semantics (and not vice versa).

4 Reasoning

In this section we study reasoning in the inconsistency-tolerant semantics introduced in the previous section. In particular, we analyze the problem of UCQ entailment under such semantics in the specific DL *DL-Lite*$_{\mathcal{A}}$ [8] for which reasoning under standard FOL semantics is tractable. We will also consider instance checking, which is a restricted form of UCQ entailment. In this section we will focus on the *data complexity* of query answering, i.e., we will measure the computational complexity only with respect to the size of the ABox (which is usually much larger than the TBox and the queries). It follows from the results in [8] that query answering in *DL-Lite*$_{\mathcal{A}}$ is in AC_o, which is a complexity class contained in PTIME, and therefore is tractable in data complexity.

We start by considering the AR-semantics. It is known that UCQ entailment is intractable under this semantics [7]. Here, we strengthen this result, and show that instance checking under the AR-semantics is already coNP-complete in data complexity even if the KB is expressed in *DL-Lite*$_{core}$. We recall that *DL-Lite*$_{core}$ is the least expressive logic in the *DL-Lite* family, as it only allows for concept expressions of the form $C ::= A|\exists R|\exists R^-$, and for TBox assertions of the form $C_1 \sqsubseteq C_2$, $C_1 \sqsubseteq \neg C_2$.

Theorem 1. *Let $\mathcal{K}$ be a DL-Lite$_{core}$ KB and let α be an ABox assertion. Deciding whether $\mathcal{K} \models_{AR} \alpha$ is coNP-complete with respect to data complexity.*

Next, we focus on the CAR-semantics, and obtain that UCQ entailment under this semantics is coNP-complete even if the TBox language is restricted to *DL-Lite*$_{core}$.

Theorem 2. *Let $\mathcal{K}$ be a DL-Lite$_{core}$ KB and let Q be a UCQ. Deciding whether $\mathcal{K} \models_{CAR} Q$ is coNP-complete with respect to data complexity.*

Notice that, differently from the AR-semantics, the above intractability result for the CAR-semantics does not hold already for the instance checking problem: we will show later in this section that instance checking is indeed tractable under the CAR-semantics.

We now turn our attention to the IAR-semantics, and define the algorithm *Compute-IAR-Repair* (see Figure 1) for computing the IAR-repair of a *DL-Lite*$_{\mathcal{A}}$ KB $\mathcal{K}$. The algorithm simply computes the set $\mathcal{D} \subseteq \mathcal{A}$ of ABox assertions which must be eliminated from the IAR-repair of $\mathcal{K}$.

```
Algorithm Compute-IAR-Repair($\mathcal{K}$)
input: DL-Lite$_{\mathcal{A}}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
output: DL-Lite$_{\mathcal{A}}$ ABox $\mathcal{A}'$
begin
  let $\mathcal{D} = \emptyset$;
  for each fact $\alpha \in \mathcal{A}$ do
    if $\langle \mathcal{T}, \{\alpha\} \rangle$ unsatisfiable
    then let $\mathcal{D} = \mathcal{D} \cup \{\alpha\}$;
  for each pair of facts $\alpha_1, \alpha_2 \in \mathcal{A} - \mathcal{D}$ do
    if $\langle \mathcal{T}, \{\alpha_1, \alpha_2\} \rangle$ unsatisfiable
    then let $\mathcal{D} = \mathcal{D} \cup \{\alpha_1, \alpha_2\}$;
  return $\mathcal{A} - \mathcal{D}$
end
```

```
Algorithm Compute-ICAR-Repair($\mathcal{K}$)
input: DL-Lite$_{\mathcal{A}}$ KB $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$
output: DL-Lite$_{\mathcal{A}}$ ABox $\mathcal{A}'$
begin
  Compute clc($\mathcal{K}$);
  let $\mathcal{D} = \emptyset$;
  for each pair of facts $\alpha_1, \alpha_2 \in$ clc($\mathcal{K}$) do
    if $\langle \mathcal{T}, \{\alpha_1, \alpha_2\} \rangle$ unsatisfiable
    then let $\mathcal{D} = \mathcal{D} \cup \{\alpha_1, \alpha_2\}$;
  return clc($\mathcal{K}$) $- \mathcal{D}$
end
```

Fig. 1. The *Compute-IAR-Repair* and *Compute-ICAR-Repair* algorithms

The following property, based on the correctness of the previous algorithm, establishes tractability of UCQ entailment under IAR-semantics.

Theorem 3. *Let $\mathcal{K}$ be a DL-Lite$_{\mathcal{A}}$ KB, and let Q be a UCQ. Deciding whether $\mathcal{K} \models_{IAR} Q$ is in PTIME with respect to data complexity.*

We now turn our attention to the $ICAR$-semantics and present the algorithm *Compute-ICAR-Repair* (see Figure 1) for computing the $ICAR$-repair of a *DL-Lite*$_{\mathcal{A}}$ KB $\mathcal{K}$. This algorithm is analogous to the previous algorithm *Compute-IAR-Repair*. The main differences are: (i) the algorithm *Compute-ICAR-Repair* returns (and operates on) a subset of *clc*$(\mathcal{K})$, while the algorithm *Compute-IAR-Repair* returns a subset of the original ABox $\mathcal{A}$; (ii) differently from the algorithm *Compute-IAR-Repair*, the algorithm *Compute-ICAR-Repair* does not need to eliminate ABox assertions α such that $\langle \mathcal{T}, \{\alpha\} \rangle$ is unsatisfiable, since such facts cannot occur in *clc*$(\mathcal{K})$.

Again, through the algorithm *Compute-ICAR-Repair* it is possible to establish the tractability of UCQ entailment under $ICAR$-semantics.

Theorem 4. *Let $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ be a DL-Lite$_{\mathcal{A}}$ KB and let Q be a UCQ. Deciding whether $\mathcal{K} \models_{ICAR} Q$ is in PTIME with respect to data complexity.*

Finally, we consider the instance checking problem under CAR-semantics, and obtain that instance checking under CAR-semantics coincides with instance checking under the $ICAR$-semantics.

Lemma 1. *Let $\mathcal{K}$ be a DL-Lite$_{\mathcal{A}}$ KB, and let α be an ABox assertion. Then, $\mathcal{K} \models_{CAR} \alpha$ iff $\mathcal{K} \models_{ICAR} \alpha$.*

The above property and Theorem 4 allow us to establish tractability of instance checking under the CAR-semantics.

Theorem 5. *Let $\mathcal{K}$ be a DL-Lite$_{\mathcal{A}}$ KB, and let α be an ABox assertion. Deciding whether $\mathcal{K} \models_{CAR} \alpha$ is in PTIME with respect to data complexity.*

We remark that the analogous of Lemma 1 does *not* hold for AR, because AR-repairs are not deductively closed. This is the reason why instance checking under AR-semantics is harder, as stated by Theorem 1.

5 Conclusions

Our work can proceed along different directions. One notable problem we aim at addressing is the design of new algorithms for inconsistency-tolerant query answering both under the IAR-semantics and the $ICAR$-semantics, based on the idea of rewriting the query into a FOL query to be evaluated directly over the inconsistent ABox. We would also like to study reasoning under inconsistency-tolerant semantics in Description Logics outside the DL-lite family.

References

1. L. E. Bertossi, A. Hunter, and T. Schaub, editors. *Inconsistency Tolerance*, volume 3300 of *LNCS*. Springer, 2005.
2. A. Calì, D. Lembo, and R. Rosati. On the decidability and complexity of query answering over inconsistent and incomplete databases. In *Proc. of PODS 2003*, pages 260–271, 2003.
3. J. Chomicki. Consistent query answering: Five easy pieces. In *Proc. of ICDT 2007*, volume 4353 of *LNCS*, pages 1–17. Springer, 2007.
4. T. Eiter and G. Gottlob. On the complexity of propositional knowledge base revision, updates and counterfactuals. *Artificial Intelligence*, 57:227–270, 1992.
5. Z. Huang, F. van Harmelen, and A. ten Teije. Reasoning with inconsistent ontologies. In *Proc. of IJCAI 2005*, pages 454–459, 2003.
6. D. Lembo, M. Lenzerini, R. Rosati, M. Ruzzi, and D. F. Savo. Inconsistency-tolerant semantics for description logics. In *Proc. of RR 2010*, 2010.
7. D. Lembo and M. Ruzzi. Consistent query answering over description logic ontologies. In *Proc. of RR 2007*, 2007.
8. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking data to ontologies. *J. on Data Semantics*, X:133–173, 2008.
9. G. Qi and J. Du. Model-based revision operators for terminologies in description logics. In *Proc. of IJCAI 2009*, pages 891–897, 2009.
10. M. Winslett. *Updating Logical Databases*. Cambridge University Press, 1990.

Optimizing the Distributed Evaluation of Stratified Datalog Programs via Structural Analysis*

Rosamaria Barilaro, Nicola Leone, Francesco Ricca, and Giorgio Terracina

Department of Mathematics, University of Calabria, Italy
{barilaro,leone,ricca,terracina}@mat.unical.it

Abstract. The database community has spent many efforts on the optimization of distributed queries. However, *efficiently reasoning* over natively distributed data through deductive databases is still an open issue. Three main problems must be faced in this context: *(i)* rules to be processed may contain many atoms and may involve complex joins among them; *(ii)* the original distribution of input data is a fact and must be considered in the optimization process; *(iii)* the integration of the deductive database engines with DBMSs must be tight enough to allow efficient interactions but general enough to avoid limitations in kind and location of databases. This paper aims to be a first step toward this direction. In fact, it provides an optimization strategy based on structural analysis facing these issues. Preliminary experimental results on real world data are encouraging.

1 Introduction

Recent developments in IT, and in particular the expansion of networking technologies, have made quite common the availability of software architectures where data sources are distributed across multiple (physically-different) sites. As a consequence, the number of applications requiring to efficiently query and *reason* on natively distributed data is constantly growing. The basic problem of querying distributed deductive databases has been already studied in the literature [12, 2]. The proposed techniques for program evaluation mostly focus on shipping the data for distributing the evaluation of rules (e.g., according to the *copy-and-constraint* [12] technique), and heuristically balancing the load on available machines [5]; however, these approaches usually do not consider as founding hypothesis that data is natively distributed. Moreover, an important role for efficiently evaluating a rule is also played by its structure; in particular, the interactions among join variables might (or might not) affect evaluation costs [11]. This follows from known results regarding conjunctive query evaluation, by considering that the evaluation of a rule body is similar to the evaluation of a conjunctive query.

Structural methods [4, 9] allow the transformation of a conjunctive query into a (tree-like) set of sub queries, allowing efficient evaluations, e.g., by the well-known Yannakakis algorithm [13]. Despite the attention received by structural query optimization in the field of databases, the specialization of such techniques for querying natively

* This work has been partially supported by the Calabrian Region under PIA (Pacchetti Integrati di Agevolazione industria, artigianato e servizi) project DLVSYSTEM approved in BURC n. 20 parte III del 15/05/2009 - DR n. 7373 del 06/05/2009.

distributed data, has not been considered much. Moreover, to the best of our knowledge, their application for distributed evaluation of datalog programs has not been investigated. In this paper we focus on a scenario, where data *natively* resides on different *autonomous* sources and it is necessary to deal with reasoning tasks via deductive database techniques. Three main problems must be faced in this context: *(i)* rules to be processed may contain many atoms and may involve complex joins among them; *(ii)* the original distribution of input data is a fact and this must be considered in the optimization process; *(iii)* the integration of the reasoning engine with DBMSs must be tight enough to allow efficient interactions but general enough to avoid limitations in kind and location of databases.

This paper aims to be a first step toward this direction. Specifically, issue *(i)* is addressed by an optimization strategy based on structural analysis, and in particular on a hypertree decomposition method [9] that we extend in order to take into account both the distribution of the sources and possible negation in rule bodies. The proposed technique includes strategies for deciding whether to ship rules or data among distributed sites; this addresses issue *(ii)*. Finally, we adopt DLV^{DB} [10] as core reasoning engine, which allows to transparently evaluate logic programs directly on commercial DBMSs; this answers issue *(iii)*.

In order to asses the effectiveness of the proposed approach, we carried out a preliminary experimental activity, on both real world and synthetic benchmarks, for comparing the performance of our approach with commercial solutions. Obtained results, reported in the paper, are encouraging and confirm our intuitions.

We next present our optimization approach for programs composed of one single rule first, and then we generalize to generic programs.

2 Single rule optimized evaluation

A single logic rule r can be seen as a conjunctive query (possibly with negation), whose result must be stored in the head predicate h.

The optimized evaluation of r starts from the computation of its structural decomposition, based on an extension of the algorithm `cost-k-decomp`, introduced in [9]; then the output of the algorithm is a hypertree which is interpreted as a distributed query plan. In more detail, `cost-k-decomp` has been extended as follows.

In order to take into account data distribution within the computation of the decomposition, each node p of the hypertree $HD = \langle N, E\rangle$ is labelled with the database where the partial data associated with it are supposed to reside. Formally, let $Site(p)$ denote the site associated with the node p in HD and let $net(Site(p), Site(p'))$ be the unitary data transfer cost from $Site(p)$ to $Site(p')$ (clearly, $net(Site(p), Site(p)) = 0$). Let $\lambda(p)$ be the set of atoms referred by p, and $\chi(p)$ the variables covered by p. $Site(p)$ is chosen among the databases where the relations in $\lambda(p)$ reside by computing: $h_m = arg\ min_{h_i \in \lambda(p)}\ \{\Sigma_{h_j \in \lambda(p)} |rel(h_j)| \times net(Site(h_j), Site(h_i))\}$. Then, $Site(p) = Site(h_m)$.

In order to handle the presence of negated atoms in r (which are not handled by `cost-k-decomp`), the algorithm for identifying valid hypertrees has been modified in such a way that each node containing a negated atom must be a leaf node and must

```
Procedure SolveRule(Hypertree Node $p$)
begin
if $p$ is a leaf and $\lambda(p)$ contains only one relation $h$ then
    if $p$ has a father $p'$ then project $h$ on $\chi(p')$
    Store the result in a relation $h_p$ in $Site(p)$ and Tag the node $p$ as solved
else if $p$ is a leaf and $\lambda(p)$ contains relations $b_1, \cdots, b_k$ then
    Set the working database of DLV$^{DB}$ as $Site(p)$
    Transfer each $b_i$ not in $Site(p)$ with the USE clause of DLV$^{DB}$
    Call DLV$^{DB}$ to evaluate the rule $h_p :- b_1, \cdots, b_k$ on $Site(p)$
    if $p$ has a father $p'$ then project $h_p$ on $\chi(p')$
    Store $h_p$ in $Site(p)$ and Tag the node $p$ as solved
else
    for each $p'_i \in \{p'_1 \cdots, p'_m\}$ being an unsolved child of $p$
        Launch a process executing SolveRule($p'_i$);
    Synchronize processes (barrier)
    Set the working database of DLV$^{DB}$ as $Site(p)$
    Let $b_1, \cdots b_k$ be the relations in $p$
    Transfer each $b_i$ and $p'_i$ not in $Site(p)$ with the USE clause of DLV$^{DB}$
    Call DLV$^{DB}$ to evaluate the rule $h_p :- b_1, \cdots, b_k, h_{p'_1} \cdots, h_{p'_l}, not\ h_{p'_{l+1}}, \cdots, not\ h_{p'_m}$
        where $p'_1, \cdots, p'_l$ are child nodes of $p$ corresponding to positive atoms in $r$
        and $p'_{l+1}, \cdots, p'_m$ are child nodes of $p$ corresponding to negated atoms in $r$
    if $p$ has a father $p'$ then project $h_p$ on $\chi(p')$
    Store $h_p$ in $Site(p)$ and Tag the node $p$ as solved
end else;
end Procedure;
```

Fig. 1. Procedure *SolveRule* for evaluating a rule optimized by hypertree decomposition

not contain other atoms. This is needed to isolate negated atoms in order to specifically handle them in the overall computation. However, observe that since r is safe these constraints are not actual limitations for computing valid hypertrees.

In order to take into account both data transfer costs and the impact of negated predicates on the overall computational costs, the cost function adopted in `cost-k-decomp` is changed to: $\omega^S_{\mathcal{H}}(HD) = \Sigma_{p \in N}(est(E(p)) + min_{h_i \in \lambda(p)}\{\Sigma_{h_j \in \lambda(p)}|rel(h_j)| \times net(Site(h_j), Site(h_i))\} + \Sigma_{(p,p') \in E}(est^*(p,p') + est(E(p')) \times net(Site(p'), Site(p))))$ where

$$est^*(p,p') = \begin{cases} est(E(p)) - est(E(p) \bowtie E(p')) & \text{if } p' \text{ is negated in } r \\ est(E(p) \bowtie E(p')) & \text{otherwise} \end{cases}$$

Here, if $\lambda(p)$ contains only one relation h and p is a leaf in HD, $est(E(p))$ is exactly the number of tuples in h; otherwise, it estimates the cardinality of the expression associated with p, namely $E(p) = \bowtie_{h \in \lambda(p)} \Pi_{\chi(p)} rel(h)$. Let R and S be two relations, $est(R \bowtie S)$ is computed as: $est(R \bowtie S) = \frac{est(R) \times est(S)}{\Pi_{A \in attr(R) \cap attr(S)} max\{V(A,R), V(A,S)\}}$ where $V(A,R)$ is the selectivity of attribute A in R. For joins with more relations one can repeatedly apply this formula to pair of relations according to a given evaluation order. A more detailed discussion on this estimation can be found in [11].

We are now able to describe how the evaluation of a rule r is carried out in our approach: *(i)* Create the hypergraph $\mathcal{H}_r$ for r. *(ii)* Call `cost-k-decomp` extended as described above on $\mathcal{H}_r$, and using $\omega^S_{\mathcal{H}}(HD)$. *(iii)* Tag each node of the obtained hypertree HD_r as *unsolved*. *(iv)* Call the Procedure *SolveRule* shown in Figure 1 to compose from HD_r a distributed plan for r and execute it.

Intuitively, once the hypertree decomposition is obtained, *SolveRule* evaluates joins bottom-up, from the leaves to the root, suitably transferring data if the sites of a child node and its father are different. Independent sub-trees are executed in parallel processes. In this way, the evaluation can benefit from parallelization.

It is worth pointing out that the benefits of parallelization possibly exploited in *SolveRule* are currently not considered in the overall cost function $\omega^{S}_{\mathcal{H}}(HD)$; in fact, the choices that can maximize parallelization are orthogonal to those aiming at minimizing join and data transfer costs. As a consequence, in a first attempt, we decided to privilege the optimization of join costs, while taking benefit of possible parallelization. Observe that this choice allows our approach to be safely exploited also when all relations reside on the same database.

3 Evaluation of the program

Three main kinds of optimization characterize our approach, namely *(i)* rule unfolding optimization, *(ii)* inter-components optimization, *(iii)* intra-component optimization. We next describe each of them.

Rule unfolding optimization. [8] In many situations, when evaluating a program, one is interested in the results of only a subset of the predicates in the program. As a matter of fact, many evaluators allow for the specification of "filters" (or output predicates) for the program. As far as stratified programs are concerned, the specification of a filter basically corresponds to specifying a relevant sub-portion of the program. Observe, moreover, that a set of filters specified on a program, can be also seen as a set of queries over the program itself, each of which asking for all the data of the corresponding predicate. Then, query oriented optimizations can be exploited in this case. Since the rule optimization strategy presented in the previous section is particularly suited for rules have long bodies, in presence of filters (or queries) in the program we adopt a standard unfolding optimization step that has the effect of making rule bodies as long as possible and, possibly, reduce their number.

Program unfolding proceeds as usual [8], it starts from the predicates specified in the filter and, following the dependencies, properly substitutes the occurrence of atoms in the body by their definitions.

As an example, consider the program **P1** in Table 1, where the output predicate is `active_courses`. The unfolding of **P1** produces the rule:

```
active_courses(CD) :- esame(A1,C,CD,A2),
      affidamenti_ing_informatica(C,A3,A4),
      dati_esami(A5,A6,C,A7,A8,A9,A4),dati_professori(A3,A10,A11).
```

Inter-Components optimization. [3]. The *Inter-Components optimization*, consists of dividing the input (possibly unfolded) program $\mathcal{P}$ into subprograms, according to the dependencies among the predicates occurring in it, and by identifying which of them can be evaluated in parallel. In detail, each program $\mathcal{P}$ can be associated with a graph, called the *(positive) Dependency Graph* of $\mathcal{P}$, which, intuitively, describes how predicates of $\mathcal{P}$ (positively) depend on each other. For a program $\mathcal{P}$, the *Dependency Graph* of $\mathcal{P}$ is a directed graph $G_{\mathcal{P}} = \langle N, E\rangle$, where N is a set of nodes and E is a set of arcs. N contains a node for each predicate of $\mathcal{P}$, and E contains an arc $e = (p, q)$ if there is a rule r in $\mathcal{P}$ such that q occurs in the head of r and p occurs in a positive literal of the body of r. The dependency graph $G_{\mathcal{P}}$ induces a subdivision of $\mathcal{P}$ into subprograms (also called *modules*) allowing for a modular evaluation. For each strongly connected component (SCC) C of $G_{\mathcal{P}}$, the set of rules defining all the predicates in C is called *module* of C. Moreover, a partial ordering among the SCCs is induced by $G_{\mathcal{P}}$, defined as follows: for any pair of SCCs A, B of $G_{\mathcal{P}}$, we say that B directly depends on A if there is an arc from a predicate of A to a predicate of B; and, B depends on A if there is a path in $G_{\mathcal{P}}$ from A to B. Intuitively, this partial ordering guarantees that a component A precedes a component B if the program module corresponding to A has to be evaluated before the one of B (because the evaluation of A produces data which are needed for the instantiation of B). In this way, when a rule r is evaluated, the extension of each body predicate of r is already stored in a database. Moreover, the partial ordering allows for determining which modules can be evaluated in parallel. Indeed, if two components A and B, do not depend on each other, then the evaluation of the corresponding program modules can be performed simultaneously, because the evaluation of A does not require the data produced by B and vice versa.

Intra-Component optimization. [3]. The *Intra-Component optimization*, allows for concurrently evaluating rules involved by the same component. Observe that rules in $\mathcal{P}$ may be recursive. A rule r occurring in a *module* of a component C (i.e., defining some predicate in C) is said to be *recursive* if there is a predicate $p \in C$ occurring in the positive body of r; otherwise, r is said to be an *exit rule*. Recursive rules are evaluated following a semi-naïve schema [11]. Specifically, for the evaluation of a module M, first all exit rules are processed in parallel by exploiting the data computed during the instantiation of the modules which M depends on (according to the partial ordering induced by the dependency graph). Only afterward, recursive rules are processed several times by applying a semi-naïve evaluation technique in which, at each iteration n, the instantiation of all the recursive rules is performed concurrently (synchronization occurs at the end of each iteration), and by exploiting only the significant information derived during iteration $n - 1$. Both exit and recursive rules are evaluated with the optimization schema described in Section 2. Observe that while input facts for the program are already stored in source databases, and, thus, their location is known beforehand, the location of data generated by each rule is dynamically chosen by the rule optimizer. Then, an head predicate p, is tagged in $G_{\mathcal{P}}$ with the location $Site(p)$ assigned to it the first time it is evaluated; this information is used by the subsequent rules with p in their body.

Remark. The reason for considering these last two optimizations comes from two considerations: *(i)* the distributed context we are facing can benefit of distributed and parallel computations; *(ii)* most of modern computers are equipped with multiple core CPUs (at least two, and often 4 cores) and, thus, they provide an easy way to handle concurrent processes.

4 Experiments

In this section we present preliminary results of the experiments we carried out by exploiting a prototypical implementation of our approach. In the following, after describing compared methods and benchmark settings, we address tests on both a real world scenario and synthetic benchmarks from OpenRuleBench [7].

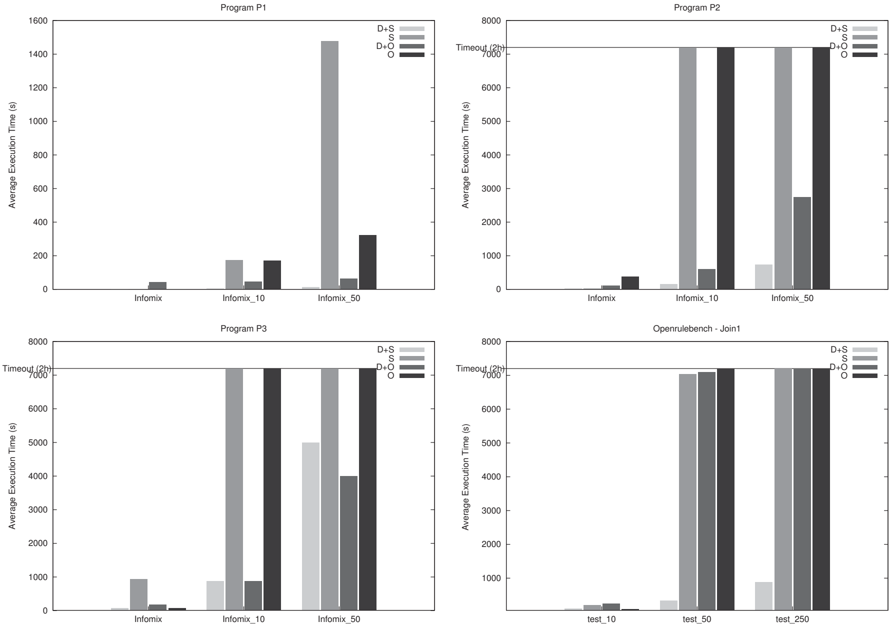


Fig. 2. Tests results.

Compared Methods and Benchmark Settings. We compared our approach with two well known DBMSs allowing to manipulate and query distributed data, namely Oracle and SQLServer. Since we are interested in comparing the behaviour of our approach with commercial DBMSs, we evaluated the programs with our approach and the corresponding (set of) SQL queries with the DBMSs. SQLServer allows to query distributed data via *linked servers*, whereas Oracle provides *database links*. In our approach DLV^{DB} has been coupled with both SQLServer and Oracle. The hardwares used for the experiments are rack mount HP ProLiant DL120 G6 equipped with Intel Xeon X3430, 2.4 GHz, with 4 Gb Ram, running Windows 2003 Server Operating System. We set a time limit of 120 minutes after which the execution of a system has been stopped. For each benchmark we have averaged the results of three consecutive runs after the first (which was not considered in order to factor out internal DBMSs optimizations like caching). In the graphs (see Figure 2), we report the execution times required by DLV^{DB} coupled with SQLServer (D+S), SQLServer (S), DLV^{DB} coupled with Oracle (D+O), and Oracle (O).

Tests on a real world scenario. We exploited the real-world data integration framework developed in the INFOMIX project (IST-2001-33570) [6], which integrates data from a real university context. In particular, considered data sources were made available by the University of Rome “La Sapienza”. We call this data set **Infomix** in the following. Moreover, we considered two further data sets, namely **Infomix-x-10** and **Infomix-x-50** storing 10 and 50 copies of the original database, respectively. It holds that **Infomix** $\subset$ **Infomix-x-10** $\subset$ **Infomix-x-50**. We then distributed the Infomix data sets over 5 sites and we compared the execution times of our prototype with the behavior of the two DBMSs on three programs.

P1:

```
exam_record(X1,X2,Z,W,X4,X5,Y) :- dati_esami(X1,A1,X2,X5,X4,A2,Y),
     affidamenti_ing_informatica(X2,X3,Y), dati_professori(X3,Z,W).
course(X1,X2) :- esame(A1,X1,X2,A2).
active_courses(CD):-course(C,CD), exam_record(X0,C,X1,X2,X3,X4,X5).
```

P2:

```
student(X1,X2,X3,X4,X5,X6,X7) :-
     studenteS1(X1,X3,X2,A1,A2,A3,A4,A5,A6,A7,A8,A9,X6,X5,A10,A11,
     X4,A12,A13,A14,A15,Y,A16), diploma_maturitaS1(Y,X7).
exams(X1,X2) :- dati_esami(X1,A1,X2,X5,X4,A2,Y).
hasCommon(X1,X3) :- student(X1,X2,X3,X4,X5,X6,X7), exams(X1,C),
     student(Y1,Y2,Y3,Y4,Y5,Y6,X7), exams(Y1,C).
```

P3:

```
teaching(X1,Z,W,X3) :- affidamenti_ing_informatica(X1,X2,X3),
     dati_professori(X2,Z,W).
exam_record(X1,X2,Z,W,X4,X5,Y) :- dati_esami(X1,A1,X2,X5,X4,A2,Y),
     affidamenti_ing_informatica(X2,X3,Y), dati_professori(X3,Z,W).
course(X1,X2) :- esame(A1,X1,X2,A2).
student(X1,X2,X3,X4,X5,X6,X7) :-
     studenteS1(X1,X3,X2,A1,A2,A3,A4,A5,A6,A7,A8,A9,X6,X5,A10,A11,
     X4,A12,A13,A14,A15,Y,A16), diploma_maturitaS1(Y,X7).
commonProf(M1,CCODE1,M2,CCODE2):- student(M1,A2,A3,A4,A5,A6,A7),
     exam_record(M1,CCODE1,B3,B4,B5,B6,B7), course(CCODE1,E2),
     exam_record(M2,CCODE2,C3,C4,C5,C6,C7), course(CCODE2,F3),
     student(M2,D2,D3,D4,D5,D6,D7),
     teaching(CCODE1,PFN,PLN,G4), teaching(CCODE2,PFN,PLN,H5).
```

Table 1. Tested programs.

The programs we tested are reported in Table 1.

Program **P1** has output predicate `active_courses`, and determines descriptions of courses for which exists at least one exam. Program **P2** has the objective of finding students having the same degree and at least one common exam; here the output predicate is `hasCommon`. Program **P3** identifies students that had a common professor, hence the focus is on `commonProf`. Note that, the rules composing the above three programs were unfolded w.r.t. output predicates and the corresponding rules rewritten as SQL queries to be tested on both Oracle and SQLServer.

The results of our experiments are presented in Figure 2. From the analysis of this figure it is possible to observe that our approach allows to obtain significant scalability improvements. In fact, while for the smallest data set times vary from few seconds (**P1**) to hundreds of seconds (**P2** and **P3**), DBMSs exceed the timeout in both **P2** and **P3** already for **Infomix-x-10**. Moreover, when DBMSs do not exceed the timeout, `D+S` allows to obtain a gain up to 99% w.r.t. `S` and `D+O` up to 80% w.r.t. `O`.

Tests from OpenRuleBench. In [7] a benchmark for testing rule based systems has been presented. In this paper, we consider the program and the instances called **join1** in [7], distributing the sources in three sites. Results are shown in Figure 2; in this case, the scalability of `D+S` is impressive w.r.t. the other systems, whereas it has been quite surprising the behaviour of `D+O`. We further investigated on this and found that: *(i)* the time required for data transfers between database links in Oracle is double w.r.t. that required by linked servers in SQLServer; *(ii)* while `D+O` required almost 2 hours for **test_50**, `O` did not finish this test in 5 hours; *(iii)* we also tried to run both `D+O` and `O` on a single machine but we had to stop them after 3 hours. Thus, we had to conclude that this test was particularly tough for Oracle, independently of our optimization.

5 Conclusions

In this paper we described an introductory approach to efficiently querying deductive databases over natively distributed data. The main innovation of the proposed approach is the adoption of a structural decomposition method for rule optimization, which takes into account also data transfer costs between different sites. Moreover, easily identifiable parallel properties are also exploited to further speed up the evaluation process.

Preliminary results of the experiments are very encouraging: the proposed approach beats out commercial DBMSs on both real world and synthetic benchmarks.

As far as future directions are concerned, we are aware that there are some points that can be tackled to further improve the effectiveness of the approach. As an example, when the hypertree decomposition groups in one node of the hypertree atoms residing in different databases, we currently choose the reference site for the node on a “local” basis. Taking into account the overall query plan could further decrease computation costs; however, considering also this aspect is not trivial if one wants to maintain low the cost of computing the decomposition.

References

1. K. R. Apt, H. A. Blair, and A. Walker. Towards a Theory of Declarative Knowledge. In Minker [8], pp. 89–148.
2. M. Balduccini, E. Pontelli, O. Elkhatib, and H. Le. Issues in parallel execution of non-monotonic reasoning systems. *Parallel Computing*, 31(6):608–647, 2005.
3. F. Calimeri, S. Perri, and F. Ricca. Experimenting with Parallelism for the Instantiation of ASP Programs. *Journal of Algorithms in Cognition, Informatics and Logics*, 63(1–3):34–54, 2008.
4. C. Chekuri and A. Rajaraman. Conjunctive query containment revisited. pp. 56–70. 1998.
5. H. M. Dewan, S. J. Stolfo, M. Hernández, and J.-J. Hwang. Predictive dynamic load balancing of parallel and distributed rule and query processing. In *Proc. of ACM SIGMOD 1994*, pp. 277–288, New York, USA, 1994. ACM.
6. N. Leone et al. The INFOMIX system for advanced integration of incomplete and inconsistent data. In *Proc. of SIGMOD’05*, pp. 915–917, New York, USA, 2005. ACM.
7. S. Liang, P. Fodor, H. Wan, and M. Kifer. Openrulebench: an analysis of the performance of rule engines. In *Proc. of WWW’09*, pp. 601–610, 2009.
8. J. Minker, editor. *Foundations of Deductive Databases and Logic Programming*. Washington DC, 1988.
9. F. Scarcello, G. Greco, and N. Leone. Weighted hypertree decompositions and optimal query plans. *JCSS*, 73(3):475–506, 2007.
10. G. Terracina, N. Leone, V. Lio, and C. Panetta. Experimenting with recursive queries in database and logic programming systems. *TPLP (TPLP)*, 8(2):129–165, 2008.
11. J. D. Ullman. *Principles of Database and Knowledge Base Systems*. Computer Science Press, 1989.
12. O. Wolfson and A. Ozeri. A new paradigm for parallel and distributed rule-processing. In *SIGMOD Conference 1990*, pp. 133–142, New York, USA, 1990.
13. M. Yannakakis. Algorithms for acyclic database schemes. In *Proc. of VLDB81*, pp. 82–94, Cannes, France, 1981.

Querying Business Processes and Ontologies in a Logic Programming Environment

(Extended Abstract)

Michele Missikoff, Maurizio Proietti, Fabrizio Smith

IASI-CNR, Viale Manzoni 30, 00185, Rome, Italy
{michele.missikoff,maurizio.proietti,fabrizio.smith}@iasi.cnr.it

1 Introduction

In recent years there has been an acceleration towards new forms of cooperation between enterprises, such as virtual enterprises, networked enterprises, or business ecosystems. A networked enterprise integrates the resources and Business Processes (BPs) of the participating organizations allowing them to operate as a unique (vitual) organization. In particular, starting from a set of BPs that exist in the various participating enterprises, the objective is to build a global BP by selecting the local BPs to be included. This operation is not an easy one, since the local BPs are often built by using different tools, according to different business logics, and using different labels and terminology to denote activities and resources. To this end, the various participating enterprises need to agree on a common view of the business domain, and provide descriptions of the local BPs according to such agreed common view. Much work has been done[1] towards the enhancement of BP management systems [1] by means of well-established techniques from the area of the Semantic Web and, in particular, computational ontologies [2]. An enterprise ontology supports unambiguous definitions of the entities occurring in the domain, and eases the interoperability between software applications and the reuse/exchange of knowledge between human actors.

In this frame, we focus on the problem of querying repositories of semantically annotated BPs. The proposed solution is based on a synergic use of an ontological framework (OPAL [3]) aimed at capturing the semantics of a business scenario, and a business process modelling framework (BPAL [4]) to represent the workflow logic. Then, the semantic annotation of BPs w.r.t. ontologies allows us to query BPs in terms of the ontology vocabulary, easing the retrieval of local BP (or process fragments) to be reused in the composition of new BPs. Figure 1 depicts a birds-eye view of the querying approach, with the local BP repositories ($\mathrm{LBPR}_{\mathrm{X}}$), the common set of ontologies and vocabularies (Reference Ontology) used for the semantic annotation (Σ_{X}) of the BP repositories, and the query engine operating on the above structures.

[1] See, e.g., the SUPER (http://www.ip-super.org/), COIN (http://www.coin-ip.eu/) and PLUG-IT (http://plug-it.org/) initiatives.

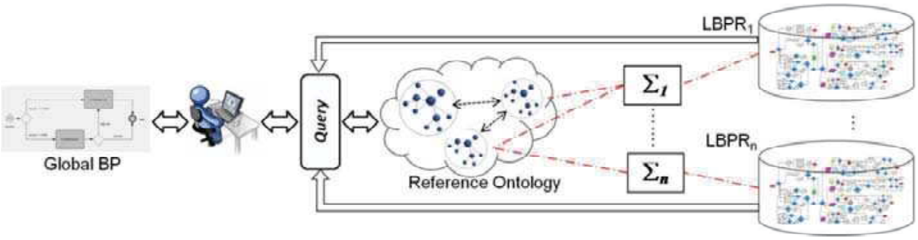


Fig. 1. Business Process Querying Approach

The proposed approach provides a uniform and formal representation framework, suited for automatic reasoning and equipped with a powerful inference mechanism supported by the solutions developed in the area of Logic Programming [5]. At the same time it has been conceived to be used in conjunction with the existing BP management tools as an ‘add-on’ to them, by supporting BPMN [6] and in particular its XPDL [7] linear form as a modeling notation and OWL [8], for the definition of the reference ontologies.

2 Knowledge Representation Framework

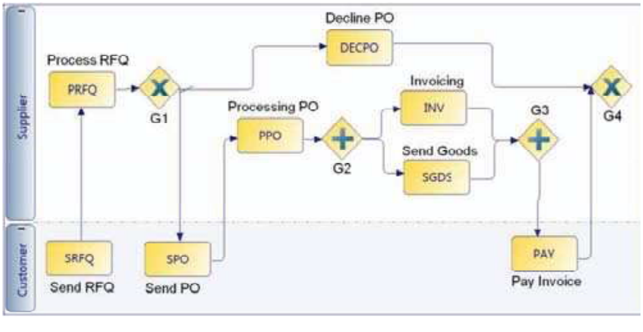
In this section we introduce the knowledge representation framework which is at the basis of the querying approach that will be proposed in Section 3. In this framework we are able to define an *Enterprise Knowledge Base* (EKB) as a collection of logical theories where: *i)* the representation of the *workflow graph* associated with each BP, together with its *behavioral semantics*, i.e., a formal description of its execution, is provided by a BPAL specification; *ii)* the representation of the *domain knowledge* regarding the business scenario is provided through an OPAL ontology.

2.1 Introducing BPAL

BPAL [4] is a logic-based language that provides a declarative modeling method capable of fully capturing the procedural knowledge in a business process. Hence it provides constructs to model activities, events, gateways and their sequencing. For branching flows, BPAL provides predicates representing *parallel* (AND), *exclusive* (XOR), and *inclusive* (OR) *branching/merging* of the control flow. A BPAL BP Schema (BPS) describes a workflow graph through a set of *facts* (ground atoms) constructed from the BPAL alphabet. In Figure 2 an exemplary BPS modeled in BPMN is depicted, together with the corresponding BPAL translation.

In order to perform several reasoning tasks over BPAL BPSs, three core theories have been defined, namely the meta-model theory ***M,*** the trace theory ***TR*** and the dependency constraint theory ***D***. ***M*** formalizes a set of structural properties of a BPS, that at this level is regarded as a labeled graph, to define how the constructs provided by the BPAL language can be used to build a *well-formed* BPS. Two categories of properties should be verified by a well-formed BPS: *i) local* properties related to the elementary components of the workflow graph (for instance, every activity must have at most one ingoing and at most one outgoing sequence flow), and *ii) global* properties related to the overall structure of the process (for instance, in this paper we assume that processes are *structured,* i.e., each branch point is matched with a merge point of the same type, and such branch-merge pairs are also properly nested).

TR provides a formalization of the trace semantics of a BP schema, where a *trace* models an execution (or instance, or enactment) of a BPS as a sequence of occurrences of activities called *steps*. ***D*** is introduced for the purpose of efficiently verifying properties regarding the possible executions of a BPS. ***D*** defines properties in the form of constraints stating that the execution of an activity is dependent on the execution of another activity, e.g., two activities have to occur together (or in mutual exclusion) in the process (possibly, in a given order). Examples of such constraints are *i) precedence(a,b,p,s,e)*, i.e., in the sub-process of *p* starting with *s* and ending with *e,* if *b* is executed then *a* has been previously executed; *ii) response(a,b,p,s,e), i.e.,* in the sub-process of *p* starting with *s* and ending with *e,* if *a* is executed then *b* will be executed. In a structured BPS, like the ones considered in this paper, such constraints could be verified by an exhaustive exploration of the set of correct traces. However, this approach would be inefficient, especially when used for answering complex queries of the kind described in Section 3. Thus, we follow a different approach for defining the constraint patterns discussed in [9] by means of logic rules that infer the absence of a counterexample. The set of these rules constitutes the theory ***D***. This approach is indeed more efficient because, in order to construct a counterexample, we can avoid to actually construct all possible interleavings of the traces generated by the execution of parallel sub-processes and, in fact, we only need to perform suitable traversals of the workflow graph.



BPAL BPS
activity(SPO)
activity(PPO)
exc_branch_pt(G1)
par_branch_pt(G2)
…
seq(PPO,G2)
exc_branch(G1,SPO,DECPO)
exc_merge(DECPO,PAY,G4)
par_branch(G2,INV,SGDS)
par_merge(INV,SGDS,G3)
…

Fig. 2. BPMN eProcurement Process (left-side), partial BPAL translation (right-side)

2.2 Semantic Annotation through a Business Reference Ontology

For the design of a *Business Reference Ontology* (**BRO**) to be used in the alignment of the terminology and conceptualizations used in different BP schemas, we consider as the reference framework the OPAL methodology [3]. **OPAL** organizes concepts through a number of meta-concepts aimed at supporting the domain expert in the conceptualization process, identifying active entities (*actors*), passive entities (*objects*), and transformations (*processes*). The latter are represented only in terms of their information structure and static relationships, without modeling behavioral issues, i.e., sequencing of activities, for which BPAL is delegated to. OPAL concepts may be defined in terms of concepts described in an ontology (or set of ontologies) describing a specific domain (or set of domains). Then the BRO is composed by an OPAL model linked to a set of domain ontologies, that can be already existing resources or artifacts developed on purpose.

The *Semantic Annotation* **Σ** defines a correspondence between elements of a BPS and concepts of a BRO, in order to describe the meaning of the former through a suitable conceptualization of the domain of interest provided by the latter in terms of related *actors, objects,* and *processes*. **Σ** is specified by the relation σ, which is defined by a set of assertions of the form *σ(El,C),* where *El* is an element of a BPS and *C* is an OPAL concept.

Technically, the language adopted for the definition of a BRO is a fragment of OWL, falling within the OWL-RL profile. OWL-RL, is an OWL subset designed for practical implementations using rule-based techniques. In the *EKB,* ontologies are encoded using the triple notation by means of the predicate *t(s,p,o)*, representing a generalized RDF triple (with subject *s*, predicate *p*, and object *o*). For the semantics of an OWL-RL ontology we refer to the axiomatization (OWL 2 RL/RDF rules) described in [8].

Figure 3 reports an example of semantic annotation related to the eProcurement process of Figure 2, where a basic definition in terms of *inputs*, *outputs* and related *actors* is provided for *IssuingPO* (we assume the usual prefixes *rdfs* and *owl* for the RDFS/OWL vocabulary, plus *opal* for the introduced vocabulary and *bro* for the specific example).

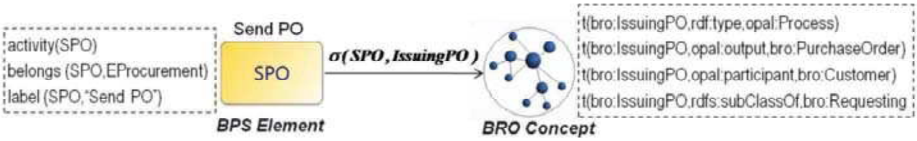

activity(SPO)
belongs (SPO,EProcurement)
label (SPO,"Send PO")
Send PO
SPO
BPS Element
σ(SPO, IssuingPO)
BRO Concept
t(bro:IssuingPO,rdf:type,opal:Process)
t(bro:IssuingPO,opal:output,bro:PurchaseOrder)
t(bro:IssuingPO,opal:participant,bro:Customer)
t(bro:IssuingPO,rdfs:subClassOf,bro:Requesting

Fig. 3. Semantic Enrichment of Process Schemas

3 Querying an Enterprise Knowledge Base

An *EKB* is formalized by a First Order Logic theory, defined by putting together the theories introduced in the previous section:

$$\boldsymbol{EKB = BRO \cup OWL\_RL \cup \Sigma \cup M \cup B \cup TR \cup D}$$

where: *i)* ***BRO*** ∪ ***OWL_RL*** ∪ **Σ** represents the *domain knowledge*, i.e., ***BRO*** is an OPAL Business Reference Ontology, encoded as a set of triples of the form *t(s,p,o)*; ***OWL_RL*** is the OWL 2 RL/RDF rule set, included into the *EKB* to support reasoning over the *BRO*; and **Σ** is a semantic annotation, including a set of assertions of the form *σ(El,C); ii)* ***M*** ∪ ***B*** represents the *structural knowledge* about the business processes, i.e., ***M*** is the meta-model theory and ***B*** is a *repository* consisting of a set of BP schemas defined in BPAL; *iii)* ***TR*** ∪ ***D*** is a formalization of the *behavioral semantics* of the BP schemas, i.e., ***TR*** is the trace theory and ***D*** is the theory defining the dependency constraints.

A relevant property of the *EKB* is that it has a straightforward translation to a logic program [5], which can be effectively used for reasoning within a Prolog environment. This translation allows us to deal within a uniform framework with several kinds of reasoning tasks and combinations thereof. Every component of the *EKB* defines a set of predicates that can be used for querying the knowledge base. The reference ontology ***BRO*** and the semantic annotation **Σ** allow us to express queries in terms of the ontology vocabulary. The predicates defined by the meta-model theory ***M*** and by the BP schemas in ***B*** allow us to query the schema level of a BP, verifying properties regarding the flow elements occurring in it (*activities, events, gateways*) and their relationships (*sequence flows*). Finally ***TR*** and ***D**,* allow us to express queries about the behavior of a BP schema at execution time, i.e., verify properties regarding the execution semantics of a BP schema.

In order to provide the user with a simple and expressive query language that does not require to understand the technicalities of the logic engine, we propose *QuBPAL*, a query language based on the SELECT-FROM-WHERE paradigm (see [10] for more details) that can be translated to logic programs and evaluated by using the XSB engine (*http://xsb.sourceforge.net*). More specifically, QuBPAL queries which do not involve predicates defined in ***TR***, i.e., queries that do not explicitly manipulate traces, are translated to logic programs belonging to the fragment of Datalog with stratified negation. For this class of programs the tabling mechanism of XSB guarantees an efficient, sound and complete top-down evaluation. As an example, below we report a *QuBPAL* query and its corresponding Datalog translation. We prefix variables names by a question mark (e.g., *?x*) and we use the notation *?x::Conc* to indicate the semantic typing of a variable, i.e., as a shortcut for *σ(x,y)* ∧ *t(y,rdfs:subClassOf,Conc),* in order to easily navigate the ontology taxonomy.

```
SELECT <?p,?s,?e>
WHERE activity(?s::bro:Requesting) AND belongs(?b::bro:FinancialTransaction,?p,?s,?e) AND
precedence(?a::bro:Invoicing,?b,?p,?s,?e)
```

```
q(P,S,E):- t(C_1,rdfs:subClassOf,bro:Requesting),t(C_2,rdfs:subClassOf,bro:FinancialTransaction),
t(C_3,rdfs:subClassOf,bro:Invoicing),σ(S,C_1),σ(B,C_2),σ(A,C_3),belongs(S,P),belongs(E,P),
belongs(A,P,S,E),belongs(B,P,S,E), wf_subproc(P,S,E),precedence(A,B,P,S,E).
```

This query returns every well-formed process fragment (i.e., structured block) that starts with a *requesting* activity and that contains a *financial transaction* preceded (in every possible run) by an *invoicing*. The *SELECT* statement defines the output of the query evaluation, which in this case is a process fragment identified by the triple *<?p,?s,?e>*, where *?p* is a BP identifier, *?s* is the starting element, and *?e* is the ending element. The query may include a *FROM* statement (absent in the above example), indicating the process(es) from which data is to be retrieved (possibly the whole repository). In the *WHERE* statement it can be specified an expression which restricts the data returned by the query, built from the set of predicates defined in the *EKB,* the = predicate and the onnectives AND, OR, NOT with the standard logic semantics. If we consider the process fragment of Section 2.1, the answer to the above query contains the sub-process starting with SPO and ending with PAY.

This query shows the interplay of the different components of the *EKB*: the notions of well-formed process fragment (*wf_subproc*) and containment (*belongs*) are formalized in the BPAL meta-model theory*, precedence* is a dependency constraint regarding the behavioral semantics of the BPS, σ and *t* are defined in terms of the semantic description of the domain specified in the BRO.

4 Implementation

A prototype of the proposed framework has been implemented as a Java application, interfaced with the XSB logic programming engine through the Interprolog library (*http://www.declarativa.com/interprolog*). The population of an *EKB* is based on two modules: *i) XPDL2BPAL* to import a process repository ***B*** from XPDL files *ii) OWL2LP,* based on the Jena2 toolkit (*http://jena.sourceforge.net/*), to import the reference ontology ***BRO*** and the semantic annotation ***Σ*** from OWL documents by a translation into a set of ground facts in the triple notation. The *EKB* is then completed by the Prolog programs encoding the meta-model theory ***M***, the trace theory ***TR****,* the dependency constraints ***D*** and the OWL 2 RL/RDF rule set ***OWL_RL.*** Having populated the *EKB,* the reasoning tasks are performed by querying the knowledge base through *QuBPAL* queries that are translated into Datalog by the module *QuBPAL2LP* and evaluated by the XSB engine. Finally, the computed results can be exported through the *XpdlWriter* module as a new XPDL file, for its visualization in a BPMS and its further reuse.

We conducted in [10] a preliminary evaluation of the system performance on a desktop machine (Intel Core2 E4500 CPU (2x2.20 GH), 2GB of RAM), to show the feasibility of the approach. In particular, the rule-based implementation of the OWL reasoner and the effective goal-oriented evaluation mechanism of the Prolog engine shown good response time and significant scalability. The results are summarized in Table 1. Timings are expressed in seconds and represent the average value over 10 runs. We generated artificial XPDL files, describing three BP repositories, **T1-T3** of different size and structure. In the first part of Table 1 we report, for each repository, the number of BPs, the total size, i.e. the total number of flow elements, the total number of gateways and the size of the smallest and biggest BP. As Business Reference Ontology we created an eProcurement ontology (about 400 named concepts described by about 2500 triples), by including part of the OWL translation of the SUMO ontology (*http://www.ontologyportal.org/translations/ SUMO.owl*). In particular, we used the *Process* hierarchy introduced in SUMO as root for the activity taxonomy (about 250 concepts) adopted for the random annotation of the generated BPs. First, we tested the set up phase (middle part of Table 1), by importing into the platform each repository from an XPDL file, the ontology and the semantic annotation from OWL. Then, we performed three queries **Q1-Q3** against each repository. Q1 is analogous to the one shown in Section 3. Q2 *retrieves every opal:Object that is related to a concept used for the annotation of an activity lying on* *a path from an activity annotated with B to an activity annotated with C.* Q3 *retrieves every sub-process that is executed as an alternative to one where an activity annotated with C is eventually executed.* We report for each run (bottom part of Table 1) the number of results obtained and the total time spent for the evaluation, including the QuBPAL query translation (*QuBPAL2LP*), the communication overhead between Java and XSB and the export of the results as a new XPDL file (*XpdlWriter*).

Table 1. Evaluation Results

Test Data Sets						
	Nr. of BPS	***Tot. Size***	***Nr. of Gateways***		***Min BPS Size***	***Max BPS Size***
T1	50	11757	4114		172	308
T2	100	18888	6442		157	237
T3	200	25229	8556		104	164
Set Up Phase Evaluation						
	BP Repository Import		**BRO Import**		**Σ Import**	
	XPDL2BPAL	*XSB Compile*	*OWL2LP*	*XSB Compile*	*OWL2LP*	*XSB Compile*
T1	3.6	7.4	1	0.7	1.8	1.2
T2	7.8	11.2	1	0.7	2.5	1.7
T3	15.3	18	1	0.7	3.3	2.5
Run Time Phase Evaluation						
	Q1		**Q2**		**Q3**	
	Nr. of Res.	*Time*	*Nr. of Res.*	*Time*	*Nr. of Res.*	*Time*
T1	11	2.5	133	4.8	47	10.2
T2	15	5.3	125	11.3	66	14.7
T3	9	8	109	17.2	44	16.9

5 Related Work and Conclusions

In this paper we presented a framework conceived to complement existing BPMS by providing advanced querying services. The proposed solution is based on a synergic use of ontologies to capture the semantics of a business scenario, and a business process modelling framework, to represent the underlying application logic. Both frameworks are seamlessly connected thanks to their grounding in logic programming and therefore it is possible to apply effective reasoning methods to query the knowledge base encompassing the two.

A first body of related works is based on the use of techniques developed in the context of the semantic web that have been extended to business process management. Relevant work in this field has been done within the SUPER project (http://www.ip-super.org/), where several foundational ontologies to model functional, organizational, informational and behavioral perspectives have been developed. In [11] a querying framework based on such ontologies is presented. Other approaches based on meta-model ontologies have been discussed, e.g., [12,13]. Unlike the aforementioned works, where the behavioral aspects are hidden or abstracted away, properties defined in terms of the execution semantics can be used in a QuBPAL query. Hence, the *EKB* provides a homogeneous framework where one can evaluate complex queries that combine properties related to the ontological dscription, the workflow structure, and the behavioral semantics of the modeled BPs.

Other approaches for BP querying are grounded in graph matching, through visual languages [14,15] grounded in graph grammars. Such approaches allow the user to query the graph representation of a process workflow in an intuitive way, but they need to be combined with external tools to reason about properties of the behavioral semantics (e.g., [14] implements translations to finite state models to be verified by using model checking techniques). Our framework not only provides a method based on Datalog for querying the structure of the workflow graph, but due to the logic-based representation it also integrates additional reasoning services. In particular, a very relevant advantage provided by our approach is the possibility of formulating queries involving the knowledge represented in domain models formally encoded by means of ontologies, hence: *i)* decoupling queries from specific processes, *ii)* overcoming semantic heterogeneities deriving, e.g., from different terminologies, *iii)* posing queries at different generalization levels, taking advantage of the semantic relations defined in the ontology, such as *subsumption*.

Future works are intended to increase the expressivity of the approach, by supporting a larger number of workflow patterns [1], and to perform the optimization of the query evaluation process, that can be strongly improved by exploiting query rewriting techniques.

6 References

1. ter Hofstede, A.H.M., van der Aalst, W.M.P., Adams, M., Russell, N.: Modern Business Process Automation: YAWL and its Support Environment. Springer, 2010.
2. Hepp, M., et al: Semantic business process management: A vision towards using semantic web services for business process management. Proc. ICEBE 2005.
3. De Nicola A., Missikoff M., Navigli R.: A software engineering approach to ontology building. *Information Systems,* 34(2):258--275(2009).
4. De Nicola, A., Missikoff, M., Proietti, M., Smith, F.: An Open Platform for Business Process Modeling and Verification. Proc. DEXA 2010. LNCS 6261, pp. 66--90, Springer, 2010.
5. Lloyd, J.W.: Foundations of Logic Programming. Springer-Verlag, Berlin, 1987. 2nd Ed.
6. OMG: Business Process Model and Notation, http://www.omg.org/spec/BPMN/2.0.
7. XPDL 2.1 Complete Specification, http://www.wfmc.org/xpdl.html.
8. OWL 2: Profiles, http://www.w3.org/TR/owl2-profiles.
9. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. Proc. ICSE'99, pp. 411-420, 1999.
10. Missikoff, M., Proietti, M., Smith, F.: Querying semantically annotated business processes, IASI-CNR, R. 10-22, 2010.
11. Markovic, I. Advanced Querying and Reasoning on Business Process Models. Proc. BIS 2008. LNBIP 7, pp.189--200, Springer, 2008.
12. Di Francescomarino, C., Tonella, P.: Crosscutting Concern Documentation by Visual Query of Business Processes. Business Process Management Workshops 2008.
13. Haller, A. Gaaloul, W., Marmolowski, M.: Towards an XPDL Compliant Process Ontology. SERVICES I 2008, pp.83-86, 2008.
14. Awad, A., Decker, G., Weske, M.: Efficient compliance checking using BPMN-Q and temporal logic. Proc. BPM 2008. LNCS 5240, pp. 326--341. Springer, 2008.
15. Beeri, C., Eyal, A., Kamenkovich, S., and Milo, T.: Querying business processes with BP-QL. *Information Systems*. 33, 6 (Sep. 2008), 477-507.

Machine Learning approaches for Contact Maps prediction in CASP9 experiment

Giuseppe Tradigo[1], Pierangelo Veltri[1], and Gianluca Pollastri[2]

[1] University Magna Græcia of Catanzaro, Catanzaro 88100, Italy,
{gtradigo,veltri}@unicz.it
[2] University College Dublin, Ireland,
gianluca.pollastri@ucd.ie

(Extended Abstract)

Abstract. Residue contact maps are bi-dimensional data structures that encode the three-dimensional structure of a protein by storing the presence of contacts among protein backbone residues. Contact maps have a key role in most state-of-the-art protein structure prediction pipelines, i.e. the prediction of three dimensional space conformation of aminoacids composing the proteins. We have designed a system (XXStout-beta) for the prediction of residue contact maps from the sequence of amino acids composing the protein. The system is based on Recursive Neural Networks, which are capable of learning an input-output mapping from sets of examples. Moreover data structures and loading/unloading algorithms have been designed for efficiently managing contact maps in primary and secondary memory. XXStout-beta has performed well at the latest CASP9 world-wide protein structure prediction competition, and is integrated in the public, high-throughput structure prediction server Distill[3]. In this paper we present informally results of XXStout-beta in CASP9 competition.

1 Contact Map prediction

In Nature, the process of protein folding is observed when a protein is synthesized in the cell. Although many factors may facilitate the folding process, the shape of a protein, with only limited exceptions is directly determined by its amino acid sequence, i.e. one can assume that the map between the sequence and the structure is a function. Finding this function is the so called Protein Folding Problem (*PFP*). PFP has been an hot topic for decades among a vast community of scientists. This has been tackled in various ways, ranging from physical simulations to knowledge-based methods such as machine learning. In the latter case one needs to observe a (sufficiently large) number of known protein structures and try to understand/learn the laws behind the folding process, or more simply the map between sequences and structure, either in part (e.g. sequence to secondary structure) or as a whole. Solving the PFP computationally

[3] http://distill.ucd.ie/distill/

is appealing, as experimental determination of the structure is a complex, time-consuming and expensive process, and as a consequence as of 2011 we only know the structures of approximately 70,000 proteins, or approximately one every 200 proteins for which we have revealed the sequence.

Every two years the CASP experiment challenges computational folding methods to predict the (then unknown) structure of several tens of proteins. Among other categories, CASP assesses the prediction of protein residue contact maps, i.e. the set of mutual distances between residues in a protein, quantised into two states (contact, for distances smaller than a threshold, and non-contact otherwise).

Correct contact maps have been shown to be lead to reasonably good 3D structures [5, 6], and predicted contact maps have been used for driving protein folding in the ab initio case (that is, when a protein is folded without relying on homology to another protein of known structure), for selecting and ranking folded protein models, for predicting folding times, protein domain boundaries, secondary structure, etc.

We have designed a novel predictor of protein residue contact maps. The predictor exploits a diverse, complex set of inputs, including the residue sequence, evolutionary information in the form of a profile of residue frequencies extracted from a multiple sequence alignment of homologous proteins (of unknown structure), predicted secondary structure, solvent accessibility and contact density and, most importantly, near and remote structural templates (when available) obtained by various methods. The predictor has two types of outputs: a simple contact/non-contact binary classification (as per CASP rules); a 4-class distance map. The latter output is used as a constraint to reconstruct 3-dimensional protein structures.

2 Neural Networks for Prediction

2.1 Methods

We predict contact and distance maps by 2D-RNNs (two-dimensional Recursive Neural Networks), which were previously described in [14] and [15]. This is a family of adaptive models for mapping two-dimensional matrices of variable size into matrices of the same size.

If $o_{j,k}$ is the entry in the j-th row and k-th column of the output matrix, and $i_{j,k}$ is the input in the same position, the input-output mapping is modelled as:

$$\begin{aligned}
o_{j,k} &= \mathcal{N}^{(O)}\left(i_{j,k}, h^{(1)}_{j,k}, h^{(2)}_{j,k}, h^{(3)}_{j,k}, h^{(4)}_{j,k}\right)\\
h^{(1)}_{j,k} &= \mathcal{N}^{(1)}\left(i_{j,k}, h^{(1)}_{j-1,k}, .., h^{(1)}_{j-s,k}, h^{(1)}_{j,k-1}, .., h^{(1)}_{j,k-s}\right)\\
h^{(2)}_{j,k} &= \mathcal{N}^{(2)}\left(i_{j,k}, h^{(2)}_{j+1,k}, .., h^{(2)}_{j+s,k}, h^{(2)}_{j,k-1}, .., h^{(2)}_{j,k-s}\right)\\
h^{(3)}_{j,k} &= \mathcal{N}^{(3)}\left(i_{j,k}, h^{(3)}_{j+1,k}, .., h^{(3)}_{j+s,k}, h^{(3)}_{j,k+1}, .., h^{(3)}_{j,k+s}\right)
\end{aligned}$$

$$h_{j,k}^{(4)} = \mathcal{N}^{(4)}\left(i_{j,k}, h_{j-1,k}^{(4)}, .., h_{j-s,k}^{(4)}, h_{j,k+1}^{(4)}, .., h_{j,k+s}^{(4)}\right)$$
$$j, k = 1, \dots, N$$
$$s = 1, \dots, S$$

where $h_{j,k}^{(n)}$ for $n = 1, \dots, 4$ are planes of hidden vectors transmitting contextual information from each corner of the matrix to the opposite corner. We parametrise the output update, and the four lateral update functions (respectively $\mathcal{N}^{(O)}$ and $\mathcal{N}^{(n)}$ for $n = 1, \dots, 4$) using five two-layered feed-forward neural networks, as in [15]. Stationarity is assumed for all residue pairs (j, k), that is the same parameters are used across all $j = 1, ..., N$ and $k = 1, ..., N$. Each of the 5 neural network contains its own individual parameters, that are not constrained to the ones of the other networks.

Since we are trying to predict both a 4-class map and a binary map, we model both classification problems within the same 2D-RNN. Hence the output $o_{j,k}$ will have two components:

$$o_{j,k} = (o_{j,k}^{(4)}, o_{j,k}^{(2)})$$

where $o_{j,k}^{(4)}$ is a vector of four numbers representing the estimated probabilities of residues j and k belonging to each of the four distance classes, and $o_{j,k}^{(2)}$ is the same for the two binary (contact vs. non-contact) classes. Both components are implemented by (independent) softmax units.

As modelled in the input-output mapping equations above, we use 2D-RNNs with *shortcut connections*. This means that a memory state depends explicitly on more that the memory state immediately previous to it along the direction of contextual propagation, i.e. the memory span is greater than one. This is effective because gradient-based learning in deep layered architectures suffers from the well known vanishing gradient problem [18]. Allowing shortcuts of length S (i.e. the memory state in position i depends directly on the state in position $i-S$) creates new paths of roughly $1/S$ of the length of the ones induced by 1-step memory dependencies, thus facilitating the transmission of contextual information over larger distances. Indeed, shortcut connections can be placed starting at any of the previous states $i - s$ for any $s \in 1, .., S$. A selective placement of shortcuts was used to produce near perfect secondary structure predictions in a bidirectional recurrent neural network when (i, s) represent native contacts [16]. Notice that increasing the number of shortcuts increases the parameters resulting in a model that may more easily overfit the data. Extending the shortcut idea beyond the 2D case or in any direction of contextual propagation is straightforward. Shortcut directions and patterns are not strictly constrained (so long as cycles are not introduced in the directed graph representing the network) and may even be learned.

The choice of input $i_{j,k}$ is an important factor for the algorithm. In the case of contact map prediction the simplest input is the amino acid symbols at (j, k). Different input encodings can be constructed to improve the algorithm. In the Input Design section we describe the input encoding we used in this study.

Training Learning proceeds by gradient descent by minimising the relative cross entropy between target and output. Since there are two independent output components (a 4-class and a binary one), the error is in fact the sum of two cross entropies, which are weighed equally. Careful management of the gradient must take place, not letting it be too small or too large: the absolute value of each component of the gradient is kept within the [0.1,1] range, meaning that it is set to 0.1 if it is smaller than 0.1, and to 1 if it is greater than 1. The learning rate is set to 0.0375 divided by the the total number of proteins in the dataset. The weights of the networks are initialised randomly.

Input format Input $i_{j,k}$ associated with the j-th and k-th residue pair contains primary sequence information, evolutionary information, structural information, and direct contact information derived from the PDB templates:

$$i_{j,k} = (i^{(E)}_{j,k}, i^{(T)}_{j,k}) \tag{1}$$

where, assuming that e units are devoted to evolutionary sequence information and structural information in the form of secondary structure[11, 10], solvent accessibility [11] and contact density [13]:

$$i^{(E)}_{i,j} = (i^{(1)^{(E)}}_{j,k}, \ldots, i^{(e)^{(E)}}_{j,k}) \tag{2}$$

Template information is placed in the remaining t units:

$$i^{(T)}_{j,k} = (i^{(1)^{(T)}}_{j,k}, \ldots, i^{(t)^{(T)}}_{j,k}) \tag{3}$$

Hence $i_{j,k}$ contains a total of $e + t$ components.

In this work $e = 58$. 20+20 units correspond to the frequencies of residues observed in the two columns j and k of the multiple sequence alignment. Structural information in the form of secondary structure (three classes), solvent accessibility (two classes), and contact density (four classes) for residue j and k are placed in the remaining 6,4 and 8 input units respectively.

For the template units we use $t = 5$, representing weighted contact class information from the templates and one template quality unit. Assume that $d^{(p)}_{j,k}$ is a 4-component binary vector encoding the contact class of the (j,k)-th residue pair in the p-th template. Then, if P is the total number of templates for a protein:

$$(i^{(1)^{(T)}}_{j,k}, \ldots, i^{(4)^{(T)}}_{j,k}) = \frac{\sum_{p=1}^{P} w_p d^{(p)}_{j,k}}{\sum_{p=1}^{P} w_p} \tag{4}$$

where w_p is the weight attributed to the p-th template. If the sequence identity between template p and the query is id_p and the quality of a template (measured as X-ray resolution + R-factor/20 or 10 for NMR hits, as in [9]) is q_s, then the weight is defined as:

$$w_p = q_p id_p^3 \tag{5}$$

Taking the cube of the identity between template and query allows us to drastically reduce the contribution of low-similarity templates when good templates are available. For instance a 90% identity template is weighed two orders of magnitude more than a 20% one. In preliminary tests (not shown) this measure performed better than a number of alternatives.

The final unit of $i_{j,k}$, the quality unit, encodes the weighted average coverage and similarity of a column of the template profile as follows:

$$i_{j,k}^{(5)^{(T)}} = \frac{\sum_{p=1}^{P} w_p c_p}{\sum_{p=1}^{P} w_p} \tag{6}$$

where c_p is the coverage of the sequence by template p (i.e. the fraction of non-gaps in the alignment). Encoding template information for the binary maps is similar.

Ab initio based predictions use only the first part of the input, $i_{j,k}^{(E)}$ from equation 2, including secondary structure, solvent accessibility and contact density, although these are predicted ab initio. The template based predictions use the complete $i_{j,k}$ as input.

2.2 Data

We use two datasets to train our predictors. The first set (D1) is obtained from the January 2007 25% pdb_select list [9]. After processing and selection of proteins no longer than 200 residues, D1 contains 2,452 proteins (and 70 million residue pairs), which we divide into a training set of 1,978 instances, and a test set of 474. The dataset is further processed to generate maps between C_β atoms. We only use this dataset to predict binary C_β maps with a 8Å threshold, as per CASP rules. The second set is obtained from the October 2009 25% pdb_select list, containing 4,818 proteins, which become 3,645 (over 100 million residue pairs) after processing and selection of sequences no longer than 200 residues. This second set, in which C_α distances are taken into account, is split into 5 approximately equal parts and 5-fold cross validation trainings are run on it in two settings: prediction of 4-class distance maps without structural homologues (templates) as inputs, or free-modelling setting (FM); including templates, or template-based modelling setting (TBM).

For CASP9 contact map predictions we ensemble 15 models from 5 different trainings (with different structural parameters, such as shortcut lengths) on D1. The models trained on D2 feed into our 3D predictor Distill [13].

3 Experimental results

The trained systems, set up as web servers, took part to the CASP9 worldwide competition. Overall results of top participating servers for contact map prediction are reported in Table 1, during the CASP9 conference held in Asilomar, California in December 2010 and are available at [19]. According to the

Server	Group	N Targets	Z score	Metapredictor
MULTICOM-CLUSTER	2	25	1.258	
Infobiotics	51	28	1.073	
Distill	**214**	**28**	**0.880**	
SAM-T08-server	103	28	0.840	
ProC_S1	375	25	0.740	*n.a.*
MULTICOM-REFINE	119	26	0.674	
PSICON	422	28	0.628	*n.a.*
SMEG-CCP	391	27	2.391	yes
MULTICOM	490	27	2.388	yes
ProC_S3	138	24	1.011	yes
MULTICOM-CONSTRUCT	80	26	0.776	yes
SAM-T06-server	244	25	0.678	yes

Table 1. The top 12 groups partecipating at the IX edition of CASP 2010 in the contact prediction category, from the official CASP assessment.

official assessment our system was one of the top three standalone predictors, and the second best that submitted all the proteins that were considered in the evaluation. The Z score value for Distill with XXStout-beta server has been calculated and evaluated in 0.880. The Z score is a standard performance measure for protein structure prediction.

References

1. Murzin A. G., Metamorphic Proteins, *Science*, 230, 1725-26, 2008
2. Fraenkel A. S., Complexity of protein folding, *Bulletin of Mathematical Biology*, 55(6): 1199-1210, 1993
3. Hart W. E., Istrail S., Robust Proofs of NP-Hardness for Protein Folding: General Lattices and Energy Potentials, *Journal of Computational Biology*, 4(1): 1-22, 1997
4. Crescenzi P., Goldman D., Papadimitriou C., Piccolboni A., Yannakakis M., On the Complexity of Protein Folding, *Journal of Computational Biology*, 5(3): 423-465, 1998
5. Vendruscolo M., Kussell E., Domany E., Recovery of protein structure from contact maps, *Folding and Design*, 2(5): 295-306, 1997
6. Walsh I., Baú D., Martin A.J.M., Mooney C., Vullo A., Pollastri G., Ab initio and template-based prediction of multi-class distance maps by two-dimensional recursive neural networks, *BMC Structural Biology*, 9(1): 5, 2009
7. Zhang Y., Skolnick J., Scoring function for automated assessment of protein structure template quality, *Proteins*, 57: 702-710, 2004
8. Berman H., Westbrook J., Feng Z., Gilliland G., Bhat T., Weissig H., Shindyalov I., Bourne P., The Protein Data Bank, *Nucl Acids Res*, 28: 235-242, 2000
9. Griep S., Hobohm U., PDBselect 1992-2009 and PDBfilter-select, *Nucleic Acids Research*, 38(1), 2009
10. Pollastri G., McLysaght A., Porter, a new, accurate server for protein secondary structure prediction, *Bioinformatics*, 21(8): 1719-1720, 2005

11. Mooney C., Pollastri G., Beyond the Twilight Zone: Automated prediction of structural properties of proteins by recursive neural networks and remote homology information, *Proteins*, 77(1): 181-90, 2009
12. Mooney C., Vullo A., Pollastri G., Protein Structural Motif Prediction in Multidimensional Φ-Ψ Space leads to improved Secondary Structure Prediction, *Journal of Computational Biology*, 13(8): 1489-1502, 2006
13. Baú D., Martin A.J.M., Mooney C., Vullo A., Walsh I. Pollastri G., Distill: a suite of web servers for the prediction of one-, two- and three-dimensional structural features of proteins, *BMC Bioinformatics*, 7(1): 402, 2006
14. Pollastri G., Baldi P., Prediction of Contact Maps by Recurrent Neural Network Architectures and Hidden Context Propagation from All Four Cardinal Corners, *Bioinformatics*, 18(Suppl 1): S62-S70, 2002
15. Baldi P., Pollastri G., The Principled Design of Large-Scale Recursive Neural Network Architectures - DAG-RNNs and the Protein Structure Prediction Problem, *Journal of Machine Learning Research*, 4(Sep): 575-602, 2003
16. Ceroni A., Frasconi P., Pollastri G., Learning Protein Secondary Structure from Sequential and Relational Data, *Neural Networks*, 18(8):1029-39, 2005.
17. Altschul S., Madden T., Schaffer A., Gapped BLAST and PSI-BLAST: a new generation of protein database search programs, *Nucleic Acids Research*, 25: 3389-3402, 1997
18. Bengio Y., Simard P., Frasconi P., Learning Long-Term Dependencies with Gradient Descent is Difficult, *IEEE Transactions on Neural Networks*, 5: 157-166, 1994
19. Casp9 website, official contact maps predictors assessment, *http://www.predictioncenter.org/casp9/doc/presentations/CASP9_RR.pdf*, 2011

Automatic Discovery and Resolution of Protein Abbreviations from Full-Text Scientific Papers: A Light-Weight Approach Towards Data Extraction from Unstructured Biological Sources (Extended Abstract)

Paolo Atzeni, Fabio Polticelli, Daniele Toti

Dipartimento di Informatica e Automazione, Università Roma Tre
atzeni@dia.uniroma3.it, toti@dia.uniroma3.it
Dipartimento di Biologia, Università Roma Tre
polticel@uniroma3.it

Abstract. We propose a methodology for discovering and resolving a wide range of protein name abbreviations from the full-text versions of scientific articles, as implemented in our PRAISED framework. Three processing steps lie at the core of our approach: an abbreviation identification phase, carried out via largely domain-independent metrics based on lexical clues and exclusion rules, whose purpose is to identify all possible abbreviations within a scientific text; an abbreviation resolution phase, which takes into account a number of syntactical and semantic criteria and corresponding optimization techniques, in order to match an abbreviation with its potential explanation; and a dictionary-based protein name identification, which is meant to eventually sort out those abbreviations actually belonging to the biological domain. We have tested our implementation against the well-known Medstract Gold Standard Corpus and a relevant subset of real scientific papers extracted from the PubMed database, obtaining significant results in terms of recall, precision and overall correctness. In comparison to other methods, our approach retains its effectiveness without compromising performance, while addressing the complexity of full-text papers instead of the simpler abstracts more generally used. At the same time, computational overhead is kept to a minimum and its light-weight approach further enhances customization and scalability.

1 Introduction

Abbreviations, in the form of acronyms, aliases or simply short versions for longer names, are commonly scattered all over the scientific publications. As far as proteins are concerned, no standardized rule or naming convention has been established so far, and writing guidelines or suggested best-practices are often ignored and disregarded. The resulting explosion of protein abbreviations has become a critical issue that can dramatically harm research productivity, and therefore yearns for as orderly and clean a solution as it can be provided. Tackling this problem is no easy feat, given the data deluge itself and the inner complexity of the biological domain, which features hundreds of short names and acronyms for proteins as well as for molecules, compounds and so forth.

In this paper, we propose a light-weight strategy for identifying and resolving protein abbreviations as found in the full texts of biological papers, making up the core components of our framework for the Identification, Disambiguation and StoragE of Protein-Related Abbreviations (PRAISED).

This strategy consists in a three-phase process where (i) candidate abbreviations are detected within a full text, via a ranking process based on lexical clues and exclusion rules (*Abbreviation Identification*); (ii) abbreviations are matched with their potential explanation, using syntactical and semantic criteria combined with fitting optimization techniques (*Abbreviation Resolution*); and (iii), resulting abbreviation-explanation pairs are sorted out according to our domain of interest (*Protein Name Identification*).

We have tested our implementation of the aforementioned methodology within our PRAISED framework against a variety of input texts, the most relevant of which being the Medstract Gold Standard Corpus and a manually annotated subset of biological articles extracted from the PubMed online database. Results were encouraging, for significantly high levels of recall were achieved, along with promising levels of precision and extremely short execution time.

The paper is structured as follows. In Section 2, related work is discussed. In Section 3, we delve deeper into the details of our whole discovery process and the three phases building it up. In Section 4, experimental results are reported and commented. And finally, in Section 5, we draw our conclusions.

2 Related Work

Automatically extracting data from unstructured sources has become more and more significant a research subject over the years, due to the increased diffusion and availability of information repositories and archives. The ideal purpose of such an information extraction system is to place a perfectly clear semantic structure upon the retrieved, often messy, data, which usually goes hand-in-hand with the creation of a corresponding relational database for storing the structured information.

In this area, several research groups have proposed a certain number of methodologies for trying to discover acronyms within a source text, ranging from general approaches to more specific techniques. These include the use of regular expressions ([4]), linguistic cues and pattern-based recognitions([6, 7, 3, 5, 8]), as well as machine learning algorithms, natural language processing and mixed methods ([2, 1]).

Recall is limited, for their scope is usually restricted to abstracts only,where a narrower variety of abbreviation forms can be detected. Also, performance in terms of execution time is seldom mentioned or not at all, and the overall results are not so easily comparable, for often modified corpora are used to test the resolution algorithms.

3 Protein Abbreviation Discovery Process

In this section we will describe the details of our protein abbreviation discovery methodology, seen as a three-phase process meant to successfully detect abbreviation-explanation pairs within a paper.

3.1 Phase one: Abbreviation Identification

The first phase of the process lies in identifying abbreviations within a text, and, as we said before, is mainly based on a series of simple but effective syntactical and lexical checks: their purpose is to take in a wide range of different terms, while at the same time keeping a considerable level of precision among the retrieved candidates. Overall, this phase involves a tokenization of the input text and a ranking assignment for each of the considered words: those scoring higher than a certain threshold will be the resulting candidate abbreviations retrieved.

Exclusion rules A preliminary cleaning of the text is performed in order for a certain set of skippable words not to be passed to the actual ranking process. This is done first by applying general-purpose rules, which are meant to remove stop-words (*and*, *of*, *or* etc.), a list of known, recurring non-acronym words (*Fig.*, *Table* etc.), and those derived from some known patterns to be excluded (like words containing no letters, or one character-long terms).

Besides, here we also apply a collection of domain-oriented exclusion rules, so that the subset of words to be later passed to the resolution phase is even more accurate. We thus tell apart aminoacids (*Tyr382, Asp383* etc.), ions (Fe^{2+} etc.) and nucleobases (*GTX, A(G/A)(A/G/T)* etc.), which might be wrongly captured by the subsequent lexical checks for possessing an abbreviation-like form.

Then, the ranking process takes place, via the consecutive checks described below.

Length check and decomposition of compound words The first check is related to the word's length, in order to establish whether the considered word has a reasonably short length. If that is the case, rank is increased accordingly, since there is a very high chance for an abbreviation to be sufficiently short. Here, we also check for the presence of special suffixes and compound words, especially delimited by linguistic elements like slashes. If suffixes are found (the most notable of which being *-like*), they are removed and the remaining word is ranked according to the subsequent metrics. If we are dealing with a composite abbreviation (e.g. *LysoPAFAT/LPCAT2*), we split it accordingly and the single words building it up are individually ranked using the subsequent metrics, eventually resulting in a ranking that will be the computed average of the individual ones.

Plain bracket check The second check tests a word for the presence of plain brackets, either left, right or both. Rank is increased upon successful discovery of parentheses, proportionally to their number, and the fully or partially bracket-enclosed words are cleaned of brackets and passed to the final ranking stage.

Multiple lexical checks The final ranking substep consists of a composition of syntactical metrics based on linguistic elements, for instance: "all uppercase letters", "more uppercase letters than lowercase letters" etc. During this last ranking step, the actual score is assigned according to the distribution of abbreviation forms in our domain of interest. Along with each candidate abbreviation, a set of contiguous words are stored, to be subsequently used in the resolution phase as the search space for the potential abbreviation origin. The size s of each set of words is dependent on the candidate abbreviation's length, resulting in $n + k$, where n is the abbreviation's length and k is a configurable factor.

3.2 Phase 2: Abbreviation Resolution

The second phase of the process is responsible of trying to match a candidate abbreviation with its potential explanation among its contiguous words previously stored. Let us now review this phase.

Pre-processing: detection of the abbreviation's building elements Before proceeding to the application of any resolution pass, a pre-processing step is performed, where the considered abbreviation is split into its basic subelements, which roughly correspond to each of its characters. We say "roughly", for letters are individually split, while contiguous digits are treated as a single unit. For instance, the elements resulting from *Cyp33* will be *C*, *y*, *p* and *33*.

The purpose of the subsequent resolution phase is to match each subelement with a term among the contiguous words we have stored: the resulting match ratio m_r will therefore be computed as $(E_m/E) * 100$, where E_m is the number of matched elements and E is the total number of elements.

First pass: matching initial characters The actual resolution process begins with this step, where the various elements of the candidate abbreviation are scanned and we check whether there are terms starting with those elements within the search space. The seeming simplicity of this step may hide its effectiveness: most "standard" abbreviations, usually from a huge range of different domains, fall into the category *A Beautiful Concept (ABC)*, and are therefore correctly resolved right after this check.

Second pass: checking for trailing "*s*" Many abbreviations are cited in the scientific papers as plural nouns, and are consequently explained as such. *Dendritic epidermal T cells (DETCs)*, *toll-like receptors (TLRs)*, *yeast artificial chromosomes (YACs)* are all examples in this regard. The missing trailing lowercase “*s*”, which the first pass could not obviously match with any initial character within the search space, is checked for its presence by the second pass, and the match ratio is updated accordingly if the last matched word is actually a plural noun.

Third pass: checking for spelled-out numbers The explanation of an abbreviation can also contain spelled-out cardinal or ordinal numbers that might correspond to actual digits featured as elements of the abbreviation itself, in any number of positions (usually at the beginning or at the end of the explanation). *Third plant homeodomain (PHD3)* is a significant example of these particular cases. The third resolution pass is meant to check for the correspondence between digits as abbreviation subelements and their spelled-out version in the search space.

Fourth pass: combining lowercase letters After the first three passes, there might still be syntactically unmatched elements of the considered abbreviation. An interesting subset of abbreviations is structured in a way that a multi-letter prefix is used instead of a single initial character. This is the case of abbreviations like *glutamate receptor (GluR)*, *lidocaine (Lid)* or *murine leukemia virus (MuLV)*. The fourth pass tries to find unresolved elements of the abbreviation that are actually lowercase letters, and checks whether they can be combined with previously resolved elements (usually uppercase letters) to form the prefix of some word within the search space, generally already matched with another abbreviation element. Elements resolved in this fashion will consequently be deemed matched as well, increasing the overall match ratio as a consequence.

Semantic expressions There can be cases where an explanation, or part of it, does not match any subelements of the corresponding abbreviation, for no syntactical bond can be traced back among some or all the abbreviation elements and the explanation terms. This is especially true when the explanation refers to another abbreviation, or more generally when correlation expressions like *as known as*, *also called* etc. are used to link theoretically uncorrelated words. When the main passes listed so far have failed to produce a match ratio ≥ 50, semantic rules are applied in order to detect these correlations between abbreviations and their origin. If found, the match ratio undergoes an increase proportional to the proximity of the correlation expression to the abbreviation itself, with a maximum value of 51 (so that, in the worst case, it might still end up above the 50 threshold).

Proximity correction Correctness of the matched terms can decrease when dealing with a large search space, usually resulting from very long abbreviations. In order to adjust the matching precision, a proximity correction is performed after the last syntactical pass. Basically, it tries to detect resolved elements “distant” to their next element of the abbreviation, in terms of the position of their matched word among the search space. If such a “distant” element is found, it looks for another word whose proximity to the next matched word is higher, and tries to match this word with the considered element, employing the criterion used in the first pass. If a match is established, the element’s previously matched word is replaced with the new-found, nearer one. We have detected an average 30% increase of correctness for long abbreviations after applying this proximity correction. Match ratio is unaffected by such a process.

3.3 Phase 3: Protein Name Identification

The third and last phase of the process has the purpose of eventually discriminating those resolved abbreviations that actually correspond to known protein names, via a dictionary-based matching step. We use a local copy of the UniProt database[1]as our source repository for all the known proteins, and apply an indexing and a subsequent search step in order to match our input with one of the records within the database. The result is a list of candidate protein names, each with a certain score: those scoring higher are more likely to be the actual proteins appearing in our input (a score of 100 means a perfect match). At the same time, a list featuring candidates having very low scores, or no candidates at all, is likely to mean that our resolved abbreviation does not refer to a protein whatsoever. A refinement is performed in the end, by considering the string similarity between the input words and the candidate protein names, so that the score of those with a greater proximity to the input is increased or maximized. Further details of this phase are described below.

Index building and search step We use the open-source Apache Lucene library[2]for building an index upon the data stored in our local copy of UniProt, and then proceed to search within it for relevant matches. For each UniProt record, we decide to store within our index both the protein name “as-is” and its potentially expanded form in terms of its spelled-out elements, if any. So, for example, in the case of *p-21 activated kinase I*, we store its expanded form *p-twenty one activated kinase first* as well. The actual search is performed against the index thus created, in the shape of two queries: one will look for a match between the input words and the index fields representing the protein names, and the other will do the same between the spelled-out versions of the input words and the spelled-out versions of each protein name. This search mechanism will assign a score >0 to those protein names somehow matching the input words; we apply here a threshold so that only the most relevant matches are returned as candidate protein names.

Distance-based refinement For determined input texts, an unsatisfactory situation might however occur. This is due to the fact that Lucene tends to

[1] http://www.uniprot.org/

[2] http://lucene.apache.org

assign the same score value to any protein name candidate that includes the same number of input words. Therefore, any protein names matching exactly the considered input words may not necessarily appear as first in the candidate list, ordered by decreasing score, as provided by Lucene. In order to make up for such a behavior, we opted to use the notion of string similarity for adjusting the scores of those candidates that indeed represented the most accurate matches with the protein abbreviation explanations. Specifically, we make use of LingPipe's implementation of the weighted edit distance and Jaccard distance[3], thanks to which we check how much the input words and the candidate list returned by Lucene are similar. This strategy allows us to almost fully compensate Lucene's imprecisions and obtain very accurate candidate lists, where the desired protein name is not in the first position in less than 3% of the cases.

4 Experimental Results

In this section we will report the results achieved during the tests we performed so far for our whole abbreviation discovery process.

We first tested our system against the widely used and publicly available Medstract Gold Standard Corpus[4], made up of a number of MEDLINE abstracts containing numerous abbreviation-explanation pairs. Here, we obtained significant results of 93.6% recall and 90% precision (f-measure: 91.7). The high level of recall is due to the presence of a limited variety of abbreviation forms.

We then proceeded to test our process against biological full-text articles, by using a manually annotated subset of unadulterated PubMed papers (∼100 papers) as the input source of our discovery process. The results we obtained displayed 80.1% recall and 60.3% precision (f-measure: 68.8)). The lower precision level is easily explained: we came by a score of biomedical non-protein abbreviations, correctly resolved by our process, which contained several terms appearing in as many known protein names from our repository (e.g. most chemical compounds, like *Nitric oxide*). Thus, several of these explanations generated non-empty candidate lists of protein names. Refining the protein name identification in this regard is one of our top priorities for improving the correctness of our whole process.

In terms of performance, the execution time for our whole process is considerably low. On a i7 Quad-Core machine with a medium load from other tasks, we managed to process an average of 80000 words per minute from the scientific texts used, resulting in a very quick (sometimes almost instantaneous) elaboration of most individual papers.

Scalability and customization have been also taken into great consideration: changing the combination of identification metrics or introducing new ones, modifying the resolution process by adding/removing passes and criteria, or fine-tuning the ranking assignment and threshold settings for other potential application domains, can all be performed in no time and with minimal effort.

[3] http://alias-i.com/lingpipe/

[4] http://medstract.org/gold-standards.html

5 Conclusions

In this paper we have proposed a methodology for the automatic identification and resolution of protein abbreviations extracted from full-text biological papers.

The implementation of this methodology in our PRAISED framework has achieved promising results in terms of precision and recall, especially within the Medstract Gold Standard Corpus; the respective values decreased when facing the far more complex full-text papers, but this is a somewhat expected outcome, given the highly unstructured domain we approached and the challenging task we took upon ourselves.

There is of course room for improvement. Based on the empirical evidence we obtained while digging into the disordered context of scattered and non-homogeneous abbreviations as seen in the scientific publications, as well as from the results of the testing stage of our discovery strategy, we have identified what we believe to be the available areas of improvement.

While our identification checks and exclusion rules proved dramatically effective, for our resolution phase semantic criteria can be extended and enhanced, in order to resolve more intricately correlated abbreviations. Furthermore, the accuracy of our protein name matching phase will have to be strengthened, by extending our post-processing step in order to increase its precision when dealing with non-protein abbreviation explanations. Refining our approach in this regard will allow us to resolve even more particular cases and further improve our resolution potential.

References

1. J. T. Chang, H. Schtze and R. B. Altman. Creating an Online Dictionary of Abbreviations from MEDLINE. In *Journal of American Medical Informatics Association (JAMIA)*, 9(6), pages 612-620, 2002.
2. D. Nadeau and P. D. Turney. A Supervised Learning Approach to Acronym Identification. In *18th Conference of the Canadian Society for Computational Studies of Intelligence, Canadian AI*, 2005.
3. Y. Park and R. J. Byrd. Hybrid Text Mining for Finding Abbreviations and Their Definitions. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*, 2001.
4. J. Pustejovsky, J. Castao, B. Cochran, M. Kotecki, M. Morrell and A. Rumshisky. Automatic Extraction of Acronym-meaning Pairs from MEDLINE Databases. In *MEDINFO*, 2001.
5. A. Schwartz and M. Hearst. A simple algorithm for identifying abbreviation definitions in biomedical texts. In *Proceedings of the Pacific Symposium on Biocomputing (PSB)*, 2003.
6. K. Taghva and J. Gilbreth. Recognizing acronyms and their definitions. In *International journal on Document Analysis and Recognition*, pages 191-198, 1999.
7. S. Yeates. Automatic extraction of acronyms from text. In *Third New Zealand Computer Science Research Students' Conference*, pages 117-124, 1999.
8. H. Yu, G. Hripcsak and C. Friedman. Mapping abbreviations to full forms in biomedical articles. In *J. Am. Med. Inform. Assoc.*, 9, pages 262-272, 2002.

User-Centered Design for Citizens' Empowerment through the Portal of the Italian Ministry of Health (Extended Abstract)*

Tiziana Catarci, Maddalena D'Addario, Paolo Felli, Laura Franceschetti,
Domenico Lembo, Massimo Mecella, Tatiana Pipan,
Alessandro Russo, Annarita Vestri, and Paolo Villari

SAPIENZA Università di Roma
firstname.lastname@uniroma1.it

Current Web portals of ministries of Health, in Europe and all over the world, are evolving from simple information sites, mostly oriented to offer institutional and administrative information, to really interactive e-health systems, providing citizens and operators with various services related to health promotion and prevention, as well as facilitating the access to services of the National Health Systems (NHSs) (cf. http://www.dh.gov.uk and http://www.nhs.uk in UK, and http://www.bmg.bund.de in Germany).

In this paper we report on a study concerning the redesign of the Web portal of the Italian Ministry of Health (http://www.salute.gov.it), jointly conducted by the ministry and Sapienza Università di Roma, from February 2010 to January 2011. Sapienza participated in the project with a multidisciplinary team involving computer scientists and engineers, sociologists and experts in communication, doctors and experts in public health, in order to fully identify and understand citizens' needs in terms of health information, on one hand, and to apply the most innovative methodologies and techniques for user-centered design and interfaces, on the other. The main products realized within this investigation have been:

- guidelines for online communication on protection and promotion of the health and access to the Italian NHS [2], targeted mainly to Italian local (i.e., at the regional and municipal level) health administrations; such guidelines are focused on communication issues ("how to talk to citizens?") and on Web design suggestions;
- a mock-up for the future Web portal of the Italian Ministry of Health, together with its design specifications [5].

This work is part of the process of renewing the relationship between health organizations and citizens, in order to improve the condition of *empowered citizens*, repeatedly

[*] This extended abstract refers to the paper with same authors and title appeared in Proc. 1st International Workshop on Engineering Interactive Computing Systems For Medicine And Health Care (EICS4Med) at ACM SIGCHI Symposium on Engineering Interactive Computing Systems, Pisa, Italy, June 13, 2011. The authors would like to thank the team of the Italian Ministry of Health for the fruitful and active collaboration during the project, in particular Daniela Rodorigo, Massimo Aquili, Claudia Biffoli, Claudia Spicola, Daniela Cricenti, Cristina Giordani. Moreover, they would like to thank all the Sapienza people collaborating in the project, namely Sara Ausiello, Andrea De Angelis, Priscilla Carcione, Eliana Ferroni, Emanuele Leoncini, Lucia Marinelli, Orazio Giancola and Domenico Lovecchio.

emphasized by the World Health Organization (WHO)[1]. The empowered citizen is able to understand and choose, to define her own life-style and to take an active role in managing her well-being, and is thus able to interact responsibly with those to whom she refers, e.g., the NHS. Empowerment means "giving power" to citizen. A citizen who has control over her state of health is a citizen able to participate in her own diagnostic, therapeutic, and rehabilitative processes. Even if a citizen is not yet a patient, provided health information could play a double role: on the one hand, it could prevent the onset of specific diseases, and, on the other hand, play an educational role that makes the citizen more aware of her rights and duties towards the NHS.

The aim of this experience paper is to present both the particular methodology adopted to produce the guidelines and the mock-up, and the main technical innovations of the portal, in terms of interactivity and user interface.

The Methodology Adopted in the Project. To devise the mock-up and the guidelines, we adopted a User-Centered Design (UCD) approach, which, as standardized in the ISO 13407, identifies four principal activities: *(i)* understand and specify the context of use, *(ii)* understand and specify the user and business requirements, *(iii)* design the product, in particular by creating a prototype, and *(iv)* evaluate the design. In this project, we focused on *(i)* – *(iii)*, whereas activity *(iv)* is currently on progress. In the initial activities, the main challenge has been to understand the context and the requirements of a portal which potentially could be visited and used by millions of citizens. In particular, we had the need to obtain useful insights on the following questions:

- which are the health needs of the Italian population, i.e., what is the epidemiological situation of the main diseases and risk factors for diseases in Italy?
- who searches the Internet/Web for health information, which forms she adopts to surf the Web, and which kind of health information is actually looked for?
- what are other information needs in terms of protection and promotion of health that could be satisfied through online communication?
- what works on Internet/Web, i.e., which types of Internet/Web health interventions are actually efficacious and effective for improving health?

To this aim, we carried out our analysis in three main stages: *(i)* design and administration of an online questionnaire aimed at identifying the needs of the citizens using (also) Internet/Web to access health information and health services; *(ii)* systematic study of the literature concerning health needs and what has been discussed and demonstrated about the effectiveness of Internet-based interventions on public health; *(iii)* systematic analysis of a significative number of websites of local health administrations, in order to derive best practices and identify the critical points to be addressed.

Then, on the basis of the outcomes, a mock-up of the portal has been devised. The rest of the paper provides some insights on the various activities. For the full details (including all the collected data) the reader is referred to [2].

[1] cf. the Alma Ata Declaration (1978), the Ottawa Charter (1986), the Jakarta Declaration (1998), the Bangkok Charter (2005).

The On-line Questionnaire. In order to collect data concerning the online information needs of users of the Italian NHS, we have run a survey in the period ranging from April 14 to September 21, 2010, through the definition and the online administration of a questionnaire. The questionnaire has been advertised on various sites, including the current portal of the Ministry of Health, the sites of some local health organizations, and Facebook. We collected 2381 responses, 2324 of which have been analyzed, after checking the quality. According to those who filled out the questionnaire, the health campaigns that should be promoted through the web sites of the Ministry of Health and other NHS organizations, as first option, should concern blood donation (20%), organ donation/transplantation (16%), workplace safety (13%), and responsible use of medicines (12%). About half of the respondents consider important to find on institutional sites information on attitudes and actions necessary to maintain good health (49%), and indications concerning the levels of quality of health services provided by the local health organizations and hospitals (50%). For the 78% of respondents, the data on levels of quality should be published on the portal of the Ministry of Health. The questionnaire, the collected results, and their analysis can be found in [2].

The Review of the Literature. To understand the Internet health information needs of the population, also to validate the online questionnaire, a systematic review of the literature was performed. 52 cross-sectional surveys, mainly from the USA and Europe, were considered. The analysis of the main surveys realized in USA, Europe and Italy [6,4,1] allowed us to identify the main determinants of Internet use for health purposes, the health information most frequently searched on the Web, and other useful data.

The evaluation of the effectiveness of health interventions delivered through Internet is a difficult task. The field has suffered from a lack of clarity and consistency. The absence of professional leadership and of accepting governing approaches, terminology, professional standards, and methodologies has caused the field of the evaluation of these intervention to be diffused and unstructured. The methodological quality of the studies evaluating the effectiveness of Internet health intervention is not high, and the possibility of some conflict of interest cannot be ruled out in some situations. Despite these limitations, considering the most comprehensive reviews on the topic [10,8,7], it is possible to draw different conclusions: (i) the most effective Internet health interventions are the tailored interventions, with adaptation to the user, personalization and feedback; (ii) the sizes of these interventions could be categorized as small to medium for population-based interventions; (iii) however, the potential impact of these interventions on population health is high, and the cost-effectiveness ratio potentially highly favorable compared to other health interventions; (iv) however, common sense advices to implement hybrid models, which could combine applications or tailored interventions, user navigation, collaborative filtering, as well as human-to-human interactions, considering the high popularity of the latter ones among the Internet users.

The results of the review of the literature allow us to draw a set of recommendations for the construction of a national evidence-based health website. In summary, it should:

1. contain information concerning the physiopathology of the human body, the most frequent diseases, risk behaviors, and health interventions of proven effectiveness, in order to improve the health literacy of citizens (cf. *Salute A–Z* in Figure 1);

2. include tailored interventions, with adaptation, personalization and feedback, aimed to promote healthy behaviors for disease prevention and compliance to secondary prevention programs of proven effectiveness (cf. the availability of applications on the portal, as showed later on in the portal description);
3. give accurate information on the organizational structure of the national NHS, in order to facilitate the access to the health care organizations and, at the same time, to promote the appropriate use of them (cf. *Esplora SSN* in Figure 1);
4. contain information about the performance of the different health care organizations (hospitals, etc.) and, possibly, of physicians;
5. include systems and tools able to endorse the participation of citizens, as well as the human-to-human interactions (cf. the use of Web 2.0 tools in the portal, as discussed in the portal description).

An Analysis of Italian Health Websites. An analysis of Italian health organizations' websites has been carried out in order to obtain useful indications for an effective online communication strategy. We have identified a set of indices and organized them in 4 dimensions, that explain the concept of "quality of online health information supply":

1. institutional identity and networking attitude: the possibility to easily identify the site with a specific authority, and also its attitude towards the development of thematic and operative networks with other public health administrations or professional associations or patients' associations working in the health service;
2. administrative transparency, based on the availability of online information regarding the organizational structure of the NHS, tasks and performance, and the availability of citizen protection mechanisms/tools;
3. availability and quality of the online services, concerning not only the quality of the website content and the use of Internet healthcare interventions, but also the quality (interactivity) of electronic forms and of facilities for on line booking of such services;
4. accessibility and ICT quality, referring to the different solutions for presenting and organizing the website content, regarding both accessibility for Internet users and technological criteria.

The analysis has considered the websites of all the Regional Governments (19 regions and 2 autonomous provinces) and the websites of a sample of 84 Local Health Authorities (out of 195 in total). Our analysis has evidenced a scenario revealing different online health communication strategies, with a strong local identity and with weak coordination by and towards the central institutional level (Ministry). In order to improve on line health communication, we suggest the following strategies:

- to recognize the centrality of citizens/patients, both in the phase of identifying and structuring the content of websites, and during the editing of the website and the organization of health services via the Internet;
- to strengthen the orientation and coordination role that the Ministry of Health should play in respect of other healthcare administrations as regards information and communication activities;

- to view online health information as the result of cooperative networking between the Ministry of Health and the other healthcare administrations (at national and local level);
- to build a network of health public administrations and professional associations and patients' associations working in the health services, aimed at exchanging information and strengthening mutual legitimacy, in order to reduce the fragmentation of information and to promote wider communication.

The Portal. Figure 1 shows the homepage of the proposed mock-up. The four main pillars of the proposal are *(i)* a navigation interface, on the top of the page, based on ”buttons” instead of the classical links, *(ii)* a large number of interactive applications, either accessible on the portal or downloadable on mobile devices, *(iii)* the possibility to customize the home page through the *MyPage* application and *(iv)* advanced search functionalities and page-to-page correlations based on taxonomies. In terms of contents, the (mock-up of the) portal respects the suggestions previously described.

The navigation menu based on buttons has the twofold aim of being aesthetically more catchy than the classical one, and ready for visualization on touch devices (e.g., iPads, touchpads, etc.), which represent the future of Web surfing. In order to enforce the citizen-oriented vocation of promoting good life habits, a lot of applications will be made available, e.g., for dietary calculation, alcohol abuse control, pregnancy check schedule, etc.; such applications, which enhance the level of interactivity of the portal and therefore attract users, can be mobile apps to be downloaded on devices, or Web applications accessible through the portal.

The possibility of customizing the page of the portal is an absolute novelty wrt existing public administrations/agencies, not only in Italy but, at the best of our knowledge, all over Europe. This has been obtained by including a MyPage application, which allows each registered user to personalize the information she wants to access.

The portal provides four main “channels”, i.e., sections specifically tailored to particular categories of information and possible interested users: citizens (corresponding to *La nostra salute*), health and administrative operators (*Attività e professioni*), users interested in institutional and organizational information (*Ministro e Ministero*), and users interested in *News e Media*. Each section, which is managed by a specific editorial board, has its own space in the home page, even if, in compliance with the user-centered approach previously described, the section dedicated to citizens is predominant.

Moreover, in order to satisfies suggestions coming from the review of the literature, a specific information system, collecting quality and performance data of healthcare organizations, will be developed and made accessible through the portal. It is also worth remarking that Web 2.0 tools are used throughout the portal (e.g., tag clouds, wiki and blogs, correlations with Facebook, Twitter, etc.), in order to promote user interaction.

To provide users with powerful and effective means to retrieve the information they are looking for on the portal, its contents are being classified according to a *taxonomy*, i.e., a classification scheme which organizes in a hierarchical structure the main categories of interest in the domain. By virtue of this classification, the user can query the portal by referring to the categories of the taxonomy, and get as reply those documents that belong to the same or related categories. Notably, the reference to the categories does not need to be explicit, i.e., the user is not required to a priori know the taxonomy to pose queries to the portal (see also below).

In order to simplify the process of definition, validation, and maintenance of the taxonomy, and to ease the possible integration in the portal of contents coming from external information sources, to realize our taxonomy we analyzed existing (de-facto) standards. Among various proposals for content classification and terminological representation in the biomedical domain (e.g., UMLS[2], ICD[3], SNOMED CT[4], GALEN[5]), in our project we referred to MeSH (Medical Subject Headings)[6] a taxonomy developed by the National Library of Medicine (NLM) of the United States. This choice has been motivated by the fact that MeSH is specifically tailored to information retrieval (and thus suited to our aims), contains also non-biomedical or clinical categories, and with respect to other proposals has more compact dimensions (it includes around 22000 terms), which makes it simpler to use.

In fact, to make easier the process of content classification, we operated a simplification of the MeSH, aimed at both reducing the number of categories and eliminating the most technical ones. However, to not loose the advantages of adopting a (de-facto) standard, we simply "cut" some branches of the "MeSH tree" so as to exclude too detailed categories. In this process, we have been helped by domain experts. The resulting taxonomy contains around 3500 MeSH terms.

Despite MeSH includes general purpose categories (e.g., the ones of the *Disciplines and Occupations* or *Phenomena and Processes* branches), we found out cases in which MeSH results insufficient in order to satisfactory classify some documents included in the portal. To overcome this problem, we decided to include in our taxonomy some of the categories used for article classification in (the Italian version of) Wikipedia[7], and in particular we selected the categories included in the *Human Activities* branch of the Wikipedia classification schema[8], which in fact substitutes in our taxonomy the (more limited) *Humanities* branch of MeSH. We choose the Wikipedia classification for two main reasons: (i) Wikipedia is one of the most popular portal on the Internet, and its contents are widely shared among several millions of users; (ii) its category tree is designed through a collaborative process aiming at including categories proposed by the users, and therefore particularly suited for information retrieval activities.

We finally observe that, to further support the process of content classification in the portal, we foreseen the development of tools helping the research of the categories in the taxonomy. In this respect several directions can be followed: (i) realization of keyword-based search mechanisms to directly access the categories of interest in the taxonomy (thus avoiding to manually navigate the taxonomy); (ii) use of a dictionary

[2] http://www.nlm.nih.gov/research/umls/index.html

[3] http://www.who.int/classifications/icd/en/

[4] http://www.ihtsdo.org/snomed-ct/

[5] http://www.opengalen.org

[6] http://www.nlm.nih.gov/mesh/meshhome.html

[7] http://www.wikipedia.org/

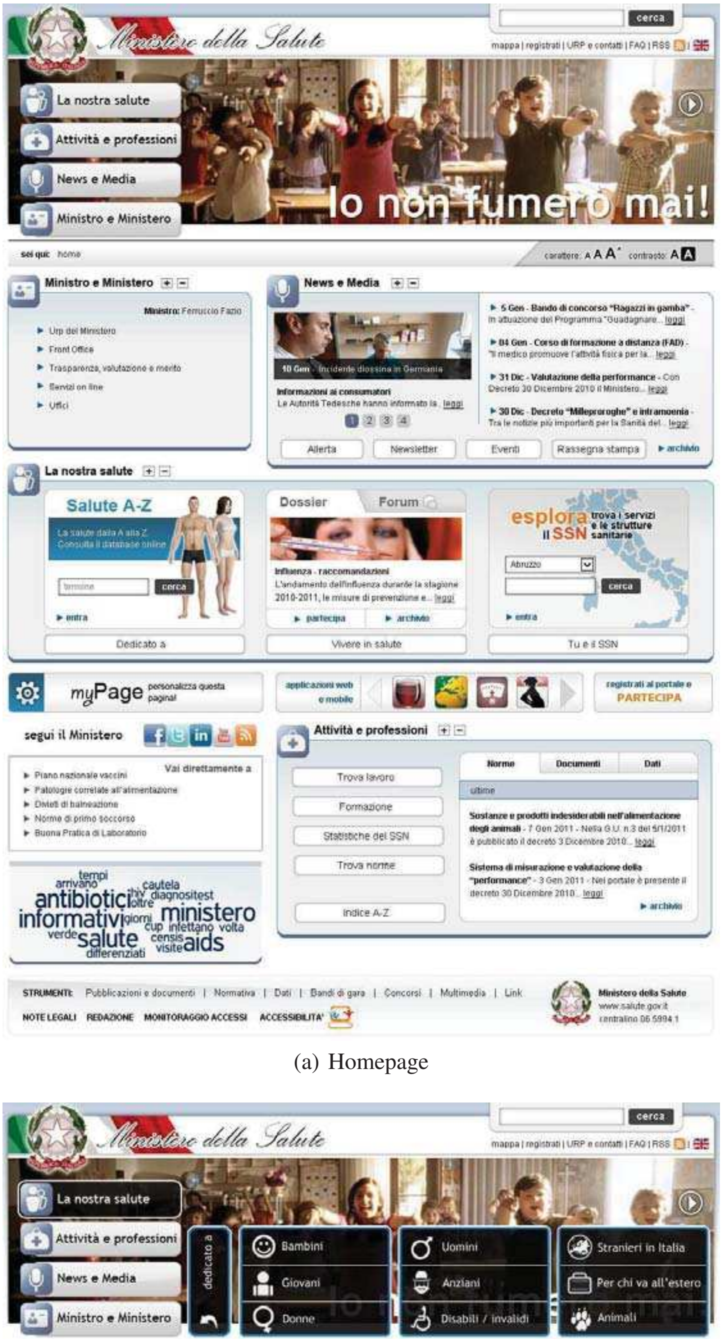
[8] http://it.wikipedia.org/w/index.php?title=Speciale:AlberoCategorie&target=attivitá+umane&mode=categories&dotree=Vai

(extending the one already available with MeSH) to include in the taxonomy also synonyms of categories, thus both expanding the lexicon of the taxonomy and including more terms in it, in a way transparent to the user; (iii) definition of techniques for automatic keyword extraction from text documents, in such a way that document classification could be done according to extracted terms, in the spirit of [3,9]. More details on the taxonomy and the classification process in the portal of the Italian Ministry of Health can be found in [5].

Conclusions. In this paper we have presented a project focused on the definition of guidelines for online communication on protection and promotion of the health in the Italian NHS (including regional health web portals, local health authorities websites, hospital websites, etc.), and the realization of a mock-up to serve as input for a redesign of the web portal of the Ministry of Health. The guidelines are published online and are currently in the process of being formally adopted by all the interested administrations. In a scenario in which all websites of the NHS organizations are realized in accordance with such guidelines, the portal of the Ministry of Health can be also able to act as a broker, in order to offer a centralized access to information and services of the NHS. An interesting future issue will be to consider how users will react, with respect to trust and privacy concerns, about the personalization features of the portal offered through the MyPage, as they may feel as their information access requests on the portal might be logged and analysed for potential medical information.

References

1. *Disparità e prossimità. Performance dei servizi, domanda di comunicazione e malattie oncologiche* (in Italian). Franco Angeli, 2007.
2. *Linee guida per la comunicazione on line in tema di tutela e promozione della salute* (in Italian). http://www.salute.gov.it/pubblicazioni/ppRisultato.jsp?id=1473, 2011.
3. Ananiadou, S., and McNaught, J. *Text mining for biology and biomedicine*. Artech House Books, 2006.
4. Andreassen, H., Bujnowska-Fedak, M., Chronaki, C., Dumitru, R., Pudule, I., Santana, S., Voss, H., and Wynn, R. European citizens' use of e-health services: a study of seven countries. *BMC Public Health 7*, 53 (2007).
5. Ausiello, S., De Angelis, A., Felli, P., Lembo, D., Mecella, M., and Russo, A. *Ministero della Salute: Ipotesi di progettazione e mock-up del nuovo portale orientato al cittadino e del relativo canale* (in Italian), 2010. A copy can be requested by email to the authors.
6. Fox, S., and Jones, S. The social life of health information. www.pewinternet.org/Reports/2009/8-The-Social-Life-of-Health-Information.aspx.
7. Krebs, P., Prochaska, J., and Rossi, J. A meta-analysis of computer-tailored interventions for health behavior change. *Prev. Med. 51*, (3–4) (2010).
8. Portnoy, D., Scott-Sheldon, L., Johnson, B., and Carey, M. Computer-delivered interventions for health promotion and behavioral risk reduction: a meta-analysis of 75 randomized controlled trials, 1988-2007. *Prev. Med. 47* (2008).
9. Rullo, P., Policicchio, V., Cumbo, C., and Iiritano, S. Olex: effective rule learning for text categorization. *IEEE Transaction on Knowledge and Data Engineering 21*, 8 (2009).
10. Strecher, V. Internet methods for delivering behavioral and health-related interventions (ehealth). *Annu. Rev. Clin. Psychol. 3* (2007).



10 Gen - Incidente diossina in Germania

Informazioni ai consumatori
Le Autorità Tedesche hanno informato la... leggi

5 Gen - Bando di concorso "Ragazzi in gamba" - in attuazione del Programma "Guadagnare... leggi

04 Gen - Corso di formazione a distanza (FAD) - "Il medico promuove l'attività fisica per la... leggi

31 Dic - Valutazione della performance - Con Decreto 30 Dicembre 2010 il Ministero... leggi

30 Dic - Decreto "Milleproroghe" e intramoenia - Tra le notizie più importanti per la Sanità del... leggi

Influenza - raccomandazioni
L'andamento dell'influenza durante la stagione 2010-2011, le misure di prevenzione e... leggi

Sostanze e prodotti indesiderabili nell'alimentazione degli animali - 7 Gen 2011 - Nella G.U. n.3 del 5/1/2011 è pubblicato il decreto 3 Dicembre 2010... leggi

Sistema di misurazione e valutazione della "performance" - 3 Gen 2011 - Nel portale è presente il decreto 30 Dicembre 2010... leggi

(a) Homepage

(b) Menu

Fig. 1. The mock-up

An approach to Content-Based Image Retrieval based on the Lucene search engine library[⋆] (Extended Abstract) [⋆⋆]

Claudio Gennaro, Giuseppe Amato, Paolo Bolettieri, Pasquale Savino
{claudio.gennaro, giuseppe.amato, paolo.bolettieri, pasquale.savino}@isti.cnr.it

ISTI - CNR, Pisa, Italy

Abstract. Content-based image retrieval is becoming a popular way for searching digital content as the amount of available multimedia data increases. However, the cost of developing from scratch a robust and reliable system with content-based image retrieval facilities for large databases is quite prohibitive.
In this paper, we propose to exploit an approach to perform approximate similarity search that is based on the observation that when two objects are very close one to each other they 'see' the world around them in the same way. Accordingly, we can use a measure of dissimilarity between the views of the world at different objects, in place of the distance function of the underlying metric space. To employ this idea the low level image features (such as colors and textures) are converted into a textual form and are indexed into the inverted index by means of the Lucene search engine library. The conversion of the features in textual form allows us to employ the Lucene's off-the-shelf indexing and searching abilities with a little implementation effort. In this way, we are able to set up a robust information retrieval system that combines full-text search with content-based image retrieval capabilities.

1 Introduction

The continuous reduction of the cost of multimedia devices such as cameras, camcorders, and smartphones, is driving the demand for content-based image and video retrieval tools for multimedia digital libraries. Several attempts are currently being made to provide these capabilities, for instance some commercial products like SnapTell (`http://www.snaptell.com`) and Google goggles (`http://www.google.com/mobile/goggles`) have been available for on-line visual search for smartphones. However, the cost of developing and deploying from scratch a robust and reliable system with content-based image retrieval facilities could not be within the range of possibilities for everyone.

[⋆] This work was partially supported by the ASSETS project, funded by the European Commission and by the VISITO project, funded by the Tuscany region of Italy.

[⋆⋆] This work is a short version of [7].

In this paper, we would like to approach the problem of similarity search by enhancing the full-text retrieval library Lucene[1] with content-based image retrieval facilities. Apache Lucene is a high-performance, full-featured text search engine library written entirely in Java that is suitable for nearly any application requiring full-text search abilities.

In particular, we use a technique for approximate similarity search when data are represented in generic metric spaces. The metric space approach to similarity search requires the similarity between objects of a database to be measured by means of a distance (dissimilarity) function, which satisfies the metric postulates: positivity, symmetry, identity, and triangle inequality. The advantage of the metric space approach to the data searching is its "extensibility", allowing us to potentially work for a large number of existing proximity measures as well as many others to be defined in the future. In contrast, many approaches need objects to be represented as vectors and cannot be applied to generic metric spaces.

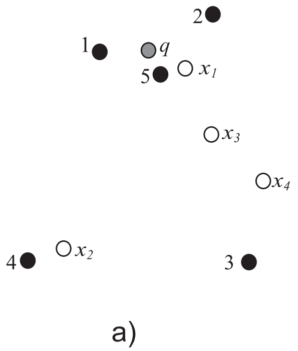
The basic idea exploited in our approach has been independently introduced by Amato et al [1] and Chavez et al. [4] and consists on observing that two objects x_1 and x_2 are very similar (which in metric spaces means that they are close one to each other), if their view of the surrounding world (their perspective) is similar as well. This implies that, if we take a set of objects from the database and we order them according to their similarity to x_1 and x_2, the obtained orderings are also similar. Therefore, we can approximatively judge the similarity between any two arbitrary objects x_1 and x_2, by comparing the ordering, according to their similarity to x_1 and x_2, of a group of reference objects, instead of using the actual distance function between the two objects.

Clearly, it is possible to find some special examples where very similar (or even identical) orderings correspond to very dissimilar objects. For instance, if reference points are all positioned on a line, two objects that are positioned on another line orthogonal to the first one will produce the same ordering of the reference points, independently of their actual position. However, as it has been proved in [1], even with a random selection of the reference points, the accuracy of this approach is very good.

Capitalizing on the work of Amato et al [1], we also use the inverted files in our research. Another similar approach, called MiPai [5], uses a compact prefix tree for estimating the real distance order of the indexed objects with respect to a query. All these above mentioned approaches make use of index methods completely designed and developed from scratch. Although the results of these systems are quite impressive[2], they probably will not easily move from research prototypes to commercial applications due to the strong effort required to maintain and support such information systems. Consider, for example, Lucene: at the time of this writing, Lucene's core team includes about half a dozen active developers. In addition to the official project developers, Lucene has a fairly

[1] http://lucene.apache.org

[2] http://mipai.esuli.it/
http://mi-file.isti.cnr.it/CophirSearch/



$$O(x_1)=(3,2,4,5,1)$$
$$O(x_2)=(4,5,2,1,3)$$
$$O(x_3)=(4,2,3,5,1)$$
$$O(x_4)=(4,3,1,5,2)$$
$$O(q)=(2,3,4,5,1)$$

b)

$$d_\rho(O(q),O(x_1))=\sqrt{(3-2)^2+(2-3)^2+(4-4)^2+(5-5)^2+(1-1)^2}=\sqrt{2}$$
$$d_\rho(O(q),O(x_3))=\sqrt{(4-2)^2+(2-3)^2+(3-4)^2+(5-5)^2+(1-1)^2}=\sqrt{6}$$
$$d_\rho(O(q),O(x_4))=\sqrt{(4-2)^2+(3-3)^2+(1-4)^2+(5-5)^2+(2-1)^2}=\sqrt{14}$$
$$d_\rho(O(q),O(x_2))=\sqrt{(4-2)^2+(5-3)^2+(2-4)^2+(1-5)^2+(3-1)^2}=\sqrt{32}$$

$$sim_{\cos}(O(q),O(x_1))=3\cdot2+2\cdot3+4\cdot4+5\cdot5+1\cdot1=54$$
$$sim_{\cos}(O(q),O(x_3))=4\cdot2+2\cdot3+3\cdot4+5\cdot5+1\cdot1=52$$
$$sim_{\cos}(O(q),O(x_4))=4\cdot2+3\cdot3+1\cdot4+5\cdot5+2\cdot1=48$$
$$sim_{\cos}(O(q),O(x_2))=4\cdot2+5\cdot3+2\cdot4+1\cdot5+3\cdot1=39$$

c)

Fig. 1. Example of perspective based space transformation. a) Black points are reference objects; white points are data objects; the gray point is a query. b) Encoding of the data objects in the transformed space. c) Distance d_ρ and similarity s in the transformed space.

large and active technical user community that frequently contributes patches, bug fixes, and new features.

Moreover, only the approach in [5] provides a full-text search on descriptive textual metadata, which is, however, not combined with the content-based similarity search. Our approach instead since it is built on top of Lucene provides complex query processing by combining similarity search with the full-text search.

The structure of the paper is as follows. Section 2 formalizes the idea of searching by using the perspective of the objects and shows how this idea can be efficiently supported by the use of the Lucene library. Section 3 proposes a preliminary performance evaluation of the proposed solution.

2 Perspective based space transformation

Let $\mathcal{D}$ be a domain of objects and $d : \mathcal{D} \times \mathcal{D} \to \mathbb{R}$ be a metric distance function between objects of $\mathcal{D}$. Let $R \in \mathcal{D}^m$, be a vector of m reference objects chosen from $\mathcal{D}$.

Given an object $x \in \mathcal{D}$, we represent it as the ordering of the reference objects R according to their distance d from x. More formally, an object $x \in \mathcal{D}$ is represented with $O(x)$, where $O(x)$ is the vector of ranks of all objects of R, ordered according to their distance d from x.

We denote the rank in $O(x)$ of a reference object $r_i \in R$ as $O_i(x)$. For example, if $O_4(x) = 3, r_4$ is the 3rd nearest object to x among those in R.

Figure 1 exemplifies the transformation process. Figure 1a) sketches a number of reference objects (black points), data objects (white points), and a query object (gray point). Figure 1b) shows the encoding of the data objects in the transformed space. We will use this as a running example throughout the remainder of the paper.

As we anticipated before, we assume that if two objects are very close one to each other, they have a similar view of the space. This means that also the orderings of the reference objects according to their distance from the two objects should be similar. There are several standard methods for comparing two ordered lists, such as as *Kendall's tau*, the *Spearman Footrule Distance*, and the *Spearman Rho Distance* [6]. In this paper, we concentrate our attention on the latter distance, which is also used in [4]. The reason of this choice (explained later on) is tied to the way standard search engines process the similarity between documents and query. Given two ordered lists $O(x)$ and $O(q)$ $(x,\, q \in \mathcal{D})$, containing the ranks of all objects of R, the Spearman Rho Distance d_ρ between $O(x)$ and $O(q)$ is computed as in the following:

$$d_\rho(O(x), O(q)) = \sqrt{\sum_{i=1}^{m} (O_i(x) - O_i(q))^2} \tag{1}$$

where m is the dimension of the vector R. This distance measures the degree in which rankings correspond with each other and it can be used in place of the metric distance d (see Figure 1c)).

In order to reduce the search cost and also, as we will see, the size of the index, it is convenient to take just the closest reference objects to represent any object that has to be indexed. Let $k_x \leq m$ be the number of reference objects used for representing the objects. Note that, in this case, different objects will be typically represented by different reference objects, given that different objects will have different neighbor reference objects. This idea can be extended also to the query, for which we can exploit a number $k_q \leq k_x$ of nearest reference objects. If we define two approximate version of the vectors $\widetilde{O}^k$, such that $\widetilde{O}_i^k(x) = k+1$ for all i such that $O_i(x) > k$ (with either $k = k_x$ or $k = k_q$), we can still use the distance (1), i.e:

$$d_\rho(\widetilde{O}^{k_x}(x), \widetilde{O}^{k_q}(q)) = \sqrt{\sum_{i=1}^{m} (\widetilde{O}^{k_x}(x) - \widetilde{O}^{k_q}(q))^2}. \tag{2}$$

In this case, we assume that x belongs to the dataset and q is the query. This is a generalization of the Spearman Rho Distance with location parameter for the special case $l = k_x = k_q$ [6], which evaluates the distance (or dissimilarity) of two top-k ranked lists.

Up to now, we have discussed how to compare two partial rankings of reference objects corresponding to objects and query. However, we did not say how to implement the proposed idea into a standard full-text search engine.

Most text search engine, including Lucene, use the Vector Space model to represent text. In this representation, a text document is represented as a vector of terms each associated with the number of occurrences of the term in the document. Therefore, we have to define a textual representation each metric object of the database so that the inverted index produced by Lucene looks like the one presented above and that its built-in similarity function behaves like the Spearman Similarity rank correlation used to compare ordered lists. This can be achieved in several ways, in the following we outline our solution.

First, we associate each element $r_i \in R$ with a unique alphanumeric keyword τ_i. Then we use the function $t^k(x)$, defined in the following, to obtain a space-separated concatenation of zero or more repetitions of τ_i words:

$$t^k(x) = \bigcup_{i=1}^{i} \bigcup_{j=1}^{k+1-O_i^k(x)} \tau_j$$

where, by abuse of notation, we denote the space-separated concatenation of words with the union operator $\bigcup$. The function $t^k(x)$ returns a text representation of x such that, if r_i appears in position p in the list of the k reference objects nearest to x, then the term τ_i is repeated $(k+1)-p$ times in the text. The function $t^k(x)$ is used to generate the textual representation of the object x to be used for both indexing and querying purposes. Specifically, we use $k = k_x$ for indexing and $k = k_q$ for querying.

In our case, this means that, if for instance term τ_i corresponding to the reference descriptor r_i $(1 \leq i \leq m)$ appears n times, the i-th element of the vector will contain the number n, and whenever τ_i does not appear it will contain 0. Let us refers to these vectors of size m as $\overline{O}^{k_x}(x)$ and $\overline{O}^{k_q}(q)$, which correspond to $t^{k_x}(x)$ and $t^{k_q}(q)$, respectively. The cosine similarity is typically adopted to determine the similarity of the query vector and a vector in the database of the text search engine, and it is defined as:

$$sim_{cos}(\overline{O}^{k_x}(x), \overline{Q}^{k_q}(q)) = \frac{\overline{O}^{k_x}(x) * \overline{Q}^{k_q}(q)}{\|\overline{O}^{k_x}(x)\| \| \overline{O}^{k_q}(q)\|},$$

where $*$ is the scalar product. sim_{cos} can be used as a function that evaluates the similarity of the two ranked lists in the same way as $d_\rho(x,q)$ defined in (2) does (although it is defined as a distance), and it is possible to prove that the first one is an order reversing monotonic transformation of the second one, and then that they are equivalent for practical aspects [3]. This means that if we use $d_\rho(\widetilde{O}^{k_x}(x), \widetilde{O}^{k_q}(q))$ and we take the first k nearest metric objects from dataset (i.e, from the shortest distance to the highest) we obtain exactly the same descriptors in the same order if we use $sim_{cos}(\overline{O}^{k_x}(x), \overline{Q}^{k_q}(q))$ and take the first k similar objects (i.e., the greater values to the smaller ones). This is illustrated in Figure 1c). The proof of this proposition is omitted due to space limitations of this paper but may be demonstrated using simple mathematical steps. To have an idea on how these textual representations look like, consider

[3] To be precise, it is possible to prove that $sim_{cos}(x,q)$ is an order reversing monotonic transformation of $d_\rho^2(x,q)$. However, since $d_\rho(x,q)$ is monotonous this does not affect the ordering.

the example reported in Figure 1, and let us assume $\tau_1 = \text{RO1}$, $\tau_2 = \text{RO2}$, etc. The function t will generate the following output

$t^5(x_1)$ = “RO5 RO5 RO5 RO5 RO5 RO2 RO2 RO2 RO2 RO1 RO1 RO1 RO3 RO3 RO4”
$t^5(x_2)$ = “RO4 RO4 RO4 RO4 RO4 RO3 RO3 RO3 RO3 RO5 RO5 RO5 RO1 RO1 RO2”
$t^5(x_3)$ = “RO5 RO5 RO5 RO5 RO5 RO2 RO2 RO2 RO2 RO3 RO3 RO3 RO1 RO1 RO4”
$t^5(x_4)$ = “RO3 RO3 RO3 RO3 RO3 RO5 RO5 RO5 RO5 RO2 RO2 RO2 RO1 RO1 RO4”

and for the query q:

$t^5(q)$ = “RO5 RO5 RO5 RO5 RO5 RO1 RO1 RO1 RO1 RO2 RO2 RO2 RO3 RO3 RO4”

If we exploit the idea of taking just the closest reference objects to represent any object that has to be indexed, and assuming, for instance, $k_x = 3$ (the number of reference objects used for indexing), and $k_q = 2$ (the number of reference objects used for generating the query), the textual representations become:

$t^3(x_1)$ = “RO5 RO5 RO5 RO2 RO2 RO1”
$t^3(x_2)$ = “RO4 RO4 RO4 RO3 RO3 RO5”
$t^3(x_3)$ = “RO5 RO5 RO5 RO2 RO2 RO3”
$t^3(x_4)$ = “RO3 RO3 RO3 RO5 RO5 RO2”

and for the query q:

$t^2(q)$ = “RO5 RO5 RO1”

This representation of an object will be clearly smaller than using all reference objects. In addition, this has also the effect of reducing the size of the inverted file. In fact, every object will be just inserted into k_x posting lists, by reducing their size and by also reducing the search cost.

3 A Real Application and Performance Evaluation

In this section, we report the results of an experimental evaluation of the proposed method. For both testing and demonstration, we developed a web user interface to perform image content based retrieval on the CoPhIR dataset [3], which consists of 106 millions images, taken from Flickr (`www.flickr.com`), described by MPEG-7 visual descriptors. Content based retrieval can be performed by using similarity functions of the visual descriptors associated with the images.

We have indexed the whole CoPhIR dataset and for each image, we created five Lucene fields which can be queried separately or in combination. The first field contains the unique identifier of the Flickr image. The second field maintains the textual information taken from title, and tags of the original Flickr image. The other three fields contain the content generated by the t function explained above for searching on three different pre-combined visual features. In particular, in order to support content based search, the CoPhIR project extracted several MPEG-7 visual descriptors from each image, three descriptors for describing the colors (SCD, CSD, and CLD) and two for describing textures (EHD and HTD). We have indexed three different aggregations of those descriptors, the first one

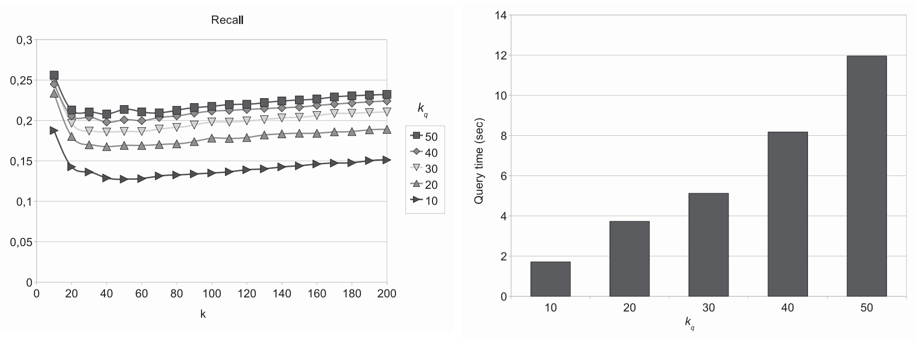


Fig. 2. Recall varying the number k for different values of k_q parameter (left), and query time for different values of k_q parameter (right).

combining the three color descriptors, the second one combining the two texture descriptors, and the third one combining all five descriptors. In this way we leave the possibility to the user to search for colors and textures independently or to search all the descriptors together. The weights used for aggregating the descriptors are the ones suggested in [2].

From page `http://lucignolo.isti.cnr.it/cophirUI` a demo web application of the developed search engine can be found. From that page it is possible to perform a full-text search, a similarity search starting from one of the random selected images. Besides the three types of visual similarities, thanks to the search functionality of Lucene, it also provides complex query processing by combining any of the three types of similarity search with the full-text search on descriptive metadata.

We conducted our experiments using the combination of all visual descriptors, with 20,000 reference objects and by setting $k_x = 50$ during the indexing. We used the measure of the recall to assess the accuracy of the method. Specifically, given a query object q, the recall is defined as $R = \frac{\#(S \cap S^A)}{\#S}$, where S and S^A are the ordering of the k closest objects to q found respectively by the exact similarity and by the proposed method. In practice, we compare the efficacy of our solution with an algorithm that exploits a sequential scan of the whole database. The comparison was made at the same conditions, using only the similarity obtained as combination of all five MPEG-7 descriptors, without exploiting the textual content. For this purpose 100 queries were randomly selected from the database. Results are shown in Figure 2. The graphs show the recall varying the number of items retrieved k for various options of the $k_q \leq k$. The performance are very similar to the one presented in [5], where the same dataset was used.

Figure 2 also shows the average query processing time as function of k_q. As expected, the search cost increases with the size of k_q, and become unacceptable for $k_q > 20$. This performance can be easily improved if the architecture consists of multiple Lucene indexes, since Lucene search framework includes parallel and multi-threads search facilities. Our index consists of ten separated Lucene indexes each one including about 1/10 of the whole dataset. If the indexes reside on different physical disks, we may obtain performance improvements; however, in our tests conducted with a single physical disk, the performance with multithread search was slightly better than with a single-thread search. The total space occupation of the Lucene indexes is 530GB, which means about 5.24KB for each image record.

4 Conclusions and future work

In this paper we presented an approach to approximate similarity search in metric spaces based on a space transformation that relies on the idea of perspective from a data point. We proved through a concrete implementation that the proposed approach has clear advantages over other methods existing in literature in terms of easiness in implementation. A major characteristic of the proposed technique is that it can be implemented by using inverted files, thus capitalizing on existing software investments. There are still some issues that are worth of investigations to further improve this technique. For instance, we can improve the quality of the approximation by reordering the first k objects of the result according to the actual distance d used in the original space of the object.

References

1. G. Amato and P. Savino. Approximate similarity search in metric spaces using inverted files. In *Proceedings of the 3rd international conference on Scalable information systems (InfoScale '08)*, pages 1–10. ICST, 2008.
2. M. Batko, P. Kohoutkova, and D. Novak. Cophir image collection under the microscope. *Similarity Search and Applications, International Workshop on*, 0:47–54, 2009.
3. P. Bolettieri, A. Esuli, F. Falchi, C. Lucchese, R. Perego, and F. Rabitti. Enabling content-based image retrieval in very large digital libraries. In *Second Workshop on Very Large Digital Libraries (VLDL 2009)*, pages 43–50. DELOS, 2009.
4. E. Chavez, K. Figueroa, and G. Navarro. Effective proximity retrieval by ordering permutations. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30:1647–1658, 2007.
5. A. Esuli. Pp-index: Using permutation prefixes for efficient and scalable approximate similarity search. In *Proceedings of the 7th Workshop on Large-Scale Distributed Systems for Information Retrieval (LSDS-IR09)*, pages 17–24, 2009.
6. R. Fagin, R. Kumar, and D. Sivakumar. Comparing top-k lists. *SIAM J. of Discrete Math.*, 17(1):134–160, 2003.
7. C. Gennaro, G. Amato, P. Bolettieri, and P. Savino. An approach to content-based image retrieval based on the lucene search engine library. In *Proceeding of the 14th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2010)*, LNCS.

A semantics-enabled registry for Web APIs recommendation (extended abstract)

Devis Bianchini, Valeria De Antonellis and Michele Melchiori

Dipartimento di Ingegneria dell'Informazione
Universita' degli Studi di Brescia, Via Branze, 38, 25123 Brescia
{bianchin|deantone|melchior}@ing.unibs.it

Abstract. Currently, Web applications can be quickly developed by combining existing APIs, independently provided by third parties, that make available ready-to-use functionalities and access to contents. This approach is gaining a lot of interest as an opportunity to integrate contents and application logics from independent sources with a limited effort. In this paper we present a semantics-enabled registry for Web API and we address the problem of supporting the retrieval and exploratory browsing of available Web APIs. Web APIs are semantically annotated and organized in the registry according to similarity and coupling criteria. The content of the registry is validated on the basis of collective knowledge available on the Web.

1 Introduction

With the advent of Web 2.0, users are more and more directly involved in the development of new Web applications, starting from many available Web APIs, independently provided by third parties. The roles of Web API provider and Web application designer are distinct in space and time: the former role is devoted to the creation of Web APIs, taking into account all technical details and technological issues, the latter mainly concentrates on Web APIs choice and composition, with a limited developing effort.

Several efforts have been devoted to the design of tools which support improved development of Web applications, often referred to as Web mashups [7]. The general problem is allowing a developer to explore and understand the space of pubblicly available Web APIs. Moreover, we aim at supporting the development process in a proactive way by suggesting the more suitable APIs at design time.

In [3], a faceted classification of unstructured Web APIs and a ranking algorithm have been applied to the ProgrammableWeb APIs repository to improve the search mechanism. The classification and searching solution is still based on IR techniques. The MatchUp system described in [4] addresses the problem of proactive suggestion of components, described according to a formal model based on Datalog rules, focusing in particular on the auto-completion of mashups: when the designer selects a component, the system suggests other components to be connected on the basis of recurrent patterns of components in the repository.

The iServe registry [8] provides a platform for publishing both RESTful and WS* services semantically annotated to better support their discovery. The focus is on semantic annotation and publishing. A Web-based interface which supports mashup of semantic-enriched Web APIs is proposed in sMash [5]. Possible mashups are shown as a graph, where each vertex represents an API and an edge between two APIs means that they are mashupable, that is, they can be used together in a mashup.

In this paper, main goal is to describe the design of a semantics-enabled Web API registry, where Web API descriptors are stored according to a conceptual model which abstracts from implementation aspects and includes semantic annotation with respect to domain ontologies. We highlight the relevance of semantic annotation to enable a semi-automatic composition of Web APIs. With respect to existing systems, we provide Web APIs selection by relying on similarity and coupling criteria and by taking into account the collective knowledge on past Web applications built starting from the Web APIs (see, for example, *www.programmableWeb.com* where about 3000 APIs and 5700 applications are published). More in general, our approach aims at building a framework for application development based on Web APIs that includes a high level phase of APIs selection and composition, separated from the low level activities of programming code writing to actually glue the selected APIs.

In the registry, Web API descriptors are organized by means of semantic links, set by computing similarity-based coefficients introduced in [2]. These coefficients are obtained through the application of matching techniques which rely on the semantic annotation of Web API descriptors. We distinguish two kinds of semantic links: functional similarity links between descriptors which provide similar functionalities, functional coupling links between descriptors whose inputs/outputs are coupled. The content of the registry is validated on the basis of collective knowledge available on the Web. Registry browsing, driven by semantic links, will provide an interactive and proactive support to the Web application designer. Registry browsing is implemented through a Web-based graphical user interface. The registry is intended to be used in an integrated way with tools and frameworks for the improved creation of Web applications based on available Web APIs provided by third parties.

The paper is organized as follows: Section 2 introduces our vision; the Web API model on which the registry is based and the registry organization are described in Section 3; Section 4 presents the Web-based interface to browse the registry contents; finally, Section 5 closes the paper and points out future work.

2 The approach

Consider for example a common estate Web application to search for a new home or to find mortgage rates. The Web application designer starts to build the application by choosing among available Web APIs and integrating their respective GUIs: Web APIs which allow to specify searching criteria for a mortgage, such as the property type or the desired minimum and maximum price; Web APIs to present estate agencies advertisements; a Google map that visualizes search

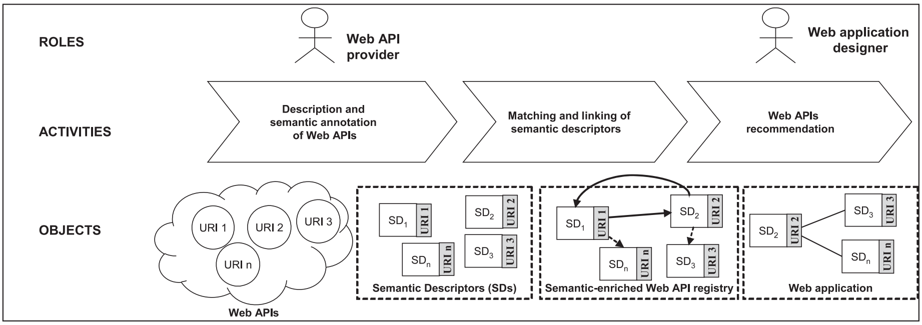

ROLES
Web API provider
Web application designer
ACTIVITIES
Description and semantic annotation of Web APIs
Matching and linking of semantic descriptors
Web APIs recommendation
OBJECTS
URI 1
URI 2
URI 3
URI n
Web APIs
SD1
SD2
SD3
SDn
Semantic Descriptors (SDs)
Semantic-enriched Web API registry
Web application

Fig. 1. The approach steps.

results on a map and also provides road markings. A set of common elements can be identified in Web APIs descriptions: (i) inputs and outputs; (ii) operations, usually associated with buttons or links on the Web API graphical interface; (iv) events, to model the interactions of the user with the Web API interface that trigger operations on the other Web APIs. However, the descriptions of APIs are featured by semantic heterogeneities that hinder their discovery as well as how they can be composes: as a very simple example, to identify a geographical position, the term `Coordinates` could be used in a Web API for estate advertisements, while the terms `Position` and `latLng` in the Google map API.

In this context, approaches and techniques to search and proactively suggest Web APIs to be combined in a Web application constitute important issues on which this paper aims at giving a contribution.

In our vision, the Web APIs are semantically annotated and made available to be assembled through the steps shown in Figure 1.

- *Semantic annotation.* Available Web APIs are semantically annotated by associating API elements (inputs/outputs/operations) to concepts defined in domain ontologies. The result of this phase is a collection of semantic descriptors, whose structure is described in Section 3.1.
- *Matching and linking of semantic descriptors.* Semantic-based matching techniques are applied to the semantic descriptors to organize them in a registry. The description of the registry and its use are the focus of this paper.
- *Web API recommendation.* Organization of Web APIs in the registry is exploited to provide: (i) proactive suggestion of Web APIs with respect to their similarity with the designer's requirements; (ii) interactive support to the designer for Web APIs composition, according to an exploratory perspective.

3 Semantics-enabled Web APIs organization

The registry we propose in this paper is composed of: (i) semantic-enriched descriptors of available Web APIs, that abstract from the implementation aspects and, specifically, include semantic annotation with respect to domain ontologies; (ii) semantic links to organize Web APIs, distinguishing between *similarity links*, set between descriptors that provide similar functionalities, and *coupling links*, set between descriptors that can be integrated in the same Web application.

3.1 Semantic descriptor for Web APIs

The description of Web APIs as found on the Web typically relies on non structured HTML documentation that makes difficult machine processing. Therefore techniques, formats and tools for semantic annotation of hRESTS Web APIs based on lightweight semantic models as MicroWSMO have been proposed in [6] to abstract from APIs heterogeneities. Semantic annotation of Web APIs requires that: (i) APIs are classified with respect to categories, that are taken from standard taxonomies available on the Web (identified by a taxonomy URI); (ii) operation names, inputs and outputs and event outputs are annotated with concepts, extracted from domain ontologies and identified by a concept URI. Domain ontologies are built by domain experts and can be designed ad-hoc for a particular application or can be made available on the Web for general purposes. If a suitable ontology is not available, it must be created first, for example using common editors (such as Protégé OWL). We do not commit any particular ontology formalism.

Semantic annotation of Web APIs is performed according to the following steps: (a) identification of elements (that is, operations, inputs, outputs) in the unstructured HTML document which represents the Web API, to produce an hRESTS description (using the SWEET tool [6]); (b) search for ontologies suitable for elements annotation and of taxonomies of categories for Web API classification; (c) annotation and classification of the Web API according to the MicroWSMO notation. The MicroWSMO description of the API, extended with semantically annotated events, is represented as a semantic descriptor [2]. Formally, we define a semantic descriptor SD_i as:

$$SD_i = \langle CAT_i, OP_i, EV_i \rangle \tag{1}$$

where: CAT_i is a set of categories, OP_i is a set of operations, EV_i is a set of events. Each operation $op_k \in OP_i$ is described by the operation name op_k, the operation inputs $IN(op_k)$ and the operation outputs $OUT(op_k)$, that are annotated with concepts taken from the reference domain ontologies. Each event $ev_h \in EV_i$ is described by the set of ontological concepts which annotate the event outputs $OUT_{ev}(ev_h)$ as well.

3.2 Similarity-based organization of Web APIs

Descriptors in the registry are related in two ways: (i) pairs of descriptors which perform the same or similar functionalities are connected through a functional similarity link; (ii) pairs of descriptors which provide complementary functionalities and can be composed in a Web application are connected through a functional coupling link. Functional similarity and coupling have been detailed in [2].

A *functional similarity link* between two semantic descriptors SD_i and SD_j is set if their categories are compatible (that is, at least one SD_j category is the same or less specific than one SD_i category) and is formally defined as:

$$\langle SD_i, SD_j, Sim_{IO}(SD_i, SD_j)\rangle \tag{2}$$

where $Sim_{IO}(SD_i, SD_j) {\geq} \delta$ is the functional similarity degree and $\delta \in [0, 1]$ is a threshold experimentally set. The functional similarity degree is computed to quantify how much SD_j provides at least the operations and I/Os required in SD_i; no matter if SD_j provides additional operations and I/Os. The building block of this expression is the Dice coefficient [9], used in Information Retrieval and the computation of a concept affinity $CAff()$ between pairs of, respectively, (i) operations, (ii) I/Os parameters, (iii) event outputs and operation inputs, used in the semantic descriptors to be matched. The coefficient $CAff \in [0..1]$ evaluates the similarity between two concepts. To this purpose, we rely on techniques such as those extensively defined in [1]. Here we simply state that $CAff$ is based on both a terminological (domain-independent) matching based on the use of WordNet and a semantic (domain dependent) matching based on ontology knowledge.

Functional coupling links between events EV_i raised by a semantic descriptor SD_i and operations OP_j caught by a semantic descriptor SD_j is formally defined as

$$\langle SD_i, SD_j, Coupl_{IO}(SD_i, SD_j)\rangle \tag{3}$$

where $Coupl_{IO}(SD_i, SD_j) {\geq} \theta$ is the coupling degree and $\theta \in [0, 1]$ is a threshold experimentally set. $Coupl_{IO}()$ is obtained by computing values of event-operation coupling coefficients, $CouplEvOp(ev_i, op_j)$, evaluated as the average $CAff()$ between the outputs of $ev_i {\in} EV_i$ and the inputs of $op_j {\in} OP_j$.

Let be $\mathcal{SD}$ the set of semantic descriptors, $OP(\mathcal{SD})$ (resp., $EV(\mathcal{SD})$) the overall set of operations (resp., events) for all the semantic descriptors in $\mathcal{SD}$, M^c the set of candidate correspondences between annotated event outputs and operation inputs automatically detected during the functional coupling evaluation. The registry is defined as a 3-uple $\langle \mathcal{SD}, \mathcal{SL}, \mathcal{CL}\rangle$, where $\mathcal{SL} {\subseteq} \mathcal{SD} {\times} \mathcal{SD} {\times} [0, 1]$ is the set of similarity links between descriptors and $\mathcal{CL} {\subseteq} \mathcal{SD} {\times} \mathcal{SD} {\times} [0, 1] {\times} M^c$ is the set of coupling links between event/operation pairs.

3.3 Tag-based validation of coupling links

Coupling links established between descriptors according to the techniques explained in the previous section are validated with reference to Web applications actually built and described in public repositories. To this purpose, we consider public repositories like *programmableWeb.com* that collect: (i) tagged Web APIs, (ii) Web applications built on these Web APIs. The validation occurs when the API is stored in the registry. In particular, joint occurrence within the same Web application of pairs of components tagged with tags associated with SD_i and SD_j is searched and took into account as a further source of evidence for the composition pattern represented by the coupling link between SD_i and SD_j. In our approach, for each descriptor SD_i in the registry, we maintain the set $TAG(SD_i)$ of its tags collected accessing *programmableWeb.com* through its

APIs[1]. Moreover, each coupling link from SD_i to SD_j is set as *strong* if exists at least a pair of tags, $t_{ih} \in TAG(SD_i)$ and $t_{jk} \in TAG(SD_j)$, such that the following conditions hold:

- $Support(t_{ih}, t_{jk}) \geq \gamma_{supp}$
- $Confidence(t_{ih}, t_{jk}) \geq \gamma_{conf}$

Otherwise, the link is set as *weak*. Values γ_{supp} γ_{conf} are thresholds that set the minimal acceptable values of confidence and support for validating a link. The coefficients $Support(t_{ih}, t_{jk})$ and $Confidence(t_{ih}, t_{jk})$, defined in Table 1, are the usual coefficients used in data mining to discover association rules between data items and are evaluated on the whole set of Web applications available on the public repository. Specifically, the first one measures the frequency of joint occurrences of t_{ih} and t_{jk} in the repository; the second one quantifies how strong is the following implication: the presence of t_{ih} in an application implies the presence of t_{jk} in the same application. The higher the values of support and confidence, the stronger the evidence that developers combine in the same application some Web APIs having the same tags featuring SD_i and SD_j.

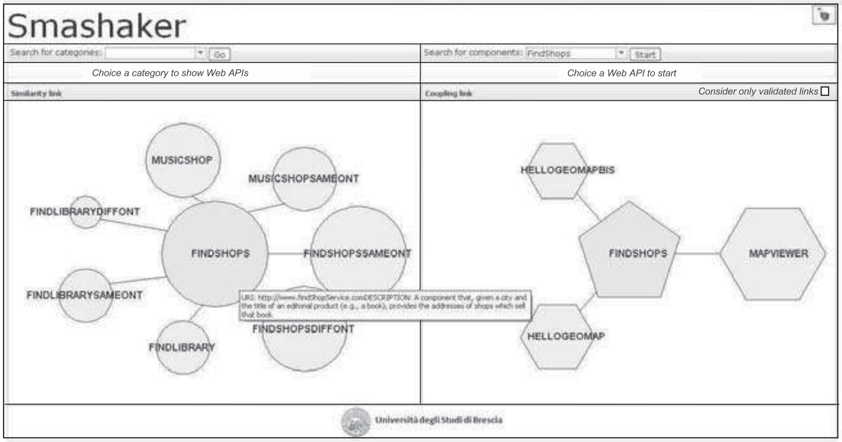
TAG-BASED SUPPORT AND CONFIDENCE
$support(t_{ih}, t_{jk}) = \frac{\#WA\ including\ comp.\ C1,\ C2,\ with\ t_{ih} \in TAG(C1)\ and\ t_{jk} \in TAG(C2)}{\#WA} \in [0, 1]$
$confidence(t_{ih}, t_{jk}) = \frac{\#WA\ including\ comp.\ C1,\ C2,\ with\ t_{ih} \in TAG(C1)\ and\ t_{jk} \in TAG(C2)}{\#WA\ including\ comp.\ C1\ with\ t_{ih} \in TAG(C1)} \in [0, 1]$
where #WA is the number Web Applications

Table 1. Support and confidence coefficients.

4 A Web-based interface for proactive suggestion of Web APIs

The Web API registry is meant to support an application design tool currently under development. The Graphical User Interface (GUI) presented in the following is part of this tool. In particular, the GUI shown in Figure 2 has been implemented to facilitate registry browsing. Descriptors and links are stored in a relational database. The interface has been implemented using the ZK open source framework, providing a library of AJAX components to implement different search functionalities on the registry:

[1] `http://api.programmableweb.com/`.



Choice a category to show Web APIs

Choice a Web API to start

URI: http://www.findShopService.comDESCRIPTION: A component that, given a city and the title of an editorial product (e.g., a book), provides the addresses of shops which sell that book.

Fig. 2. Graphical User Interface to show the results of the experimental validation.

- **search by category**, in the upper-left part of the interface, enables filtering of Web APIs according to their category; the category is selected in the "Search for categories" field;
- **search by keyword**, in the upper-right part of the interface, enables traditional keyword-based retrieval of components; the keywords are specified in the "Search for components" field and are matched against the names of the Web APIs, the names of their inputs/outputs or the names of their operations;
- **browse by similarity**, on the left, enables browsing according to similarity degree between Web API descriptors; the selected descriptor SD_i is highlighted as a circle in the center of the "Similarity link" panel (e.g., the `FindShops` descriptor in Figure 2); all the descriptors SD_j related to SD_i through a similarity link are displayed as circles around SD_i; the size of each circle is proportional to the $Sim_{IO}(SD_i, SD_j)$ value; for example, the `FindShops` descriptor is more similar to the `MusicShop` descriptor than to the `FindLibrary` one; by moving the mouse on a descriptor, a tooltip shows details about the descriptor itself;
- **browse by coupling**, on the right, enables browsing according to coupling degree between Web API descriptors; the selected descriptor SD_i is highlighted as a pentagon in the center of the "Coupling link" panel (see, for example, the `FindShops` descriptor in Figure 2); other descriptors SD_j coupled with SD_i are shown as hexagons around the pentagon; the size of each hexagon is proportional to the $Coupl_{IO}(SD_i, SD_j)$ value; for example, the `MapViewer` descriptor presents a $Coupl_{IO}(SD_i, SD_j)$ value that is higher then the `HelloGeoMap` descriptor. A filter is available in the registry GUI to restrict the result to show only strong links.

Planned experimentation and validation. We plan to experimentally evaluate the registry prototype according to effectiveness and efficacy. The effectiveness of the registry will be investigated according to the recall and precision of the search functionalities that we discussed above. In this test, a domain expert will manually define the set of Web APIs that are similar or coupled with respect to a given request. Then the number of relevant APIs identified through the search functionalities will be counted and compared with: (i) the total number of APIs returned by the system (precision); and (ii) the total number of relevant APIs manually identified in the dataset (recall). Then we intend to evaluate the efficiency of the registry by analyzing the response time of the system with respect to the number of Web APIs in the registry (population) for adding a new component to the registry and setting the similarity and coupling links.

5 Conclusions and future work

In this paper we presented a semantic-enabled registry of Web APIs. Web API are semantically annotated and organized in the registry according to similarity and coupling criteria. The registry is intended to be used in an integrated way with tools and frameworks for easy creation of Web applications starting from available Web APIs, implementing different kinds of search modalities based on the registry structure. Experimentation will be performed to evaluate the effectiveness of registry in terms of precision and recall for selection of coupled and similar Web APIs. Moreover, we plan to evaluate the scalability of the system to publish new Web APIs with respect to the registry size.

References

1. D. Bianchini, V. De Antonellis, and M. Melchiori. Flexible Semantic-based Service Matchmaking and Discovery. *World Wide Web Journal*, 11(2):227–251, 2008.
2. D. Bianchini, V. De Antonellis, and M. Melchiori. A recommendation system for semantic mashup design. In *Proc. of Ninth International Workshop on Web Semantics - WebS 2010 - DEXA Workshops*, pages 159–163, 2010.
3. K. Gomadam, A. Ranabahu, M. Nagarajan, A. Sheth, and K. Verma. A Faceted Classification Based Approach to Search and Rank Web APIs. In *ICWS*, pages 177–184, 2008.
4. O. Greenshpan, T. Milo, and N. Polyzotis. Autocompletion for Mashups. In *Proc. of the 35th Int. Conf. on Very Large DataBases (VLDB'09)*, pages 538–549, 2009.
5. Bin Lu, Zhaohui Wu, Yuan Ni, Guo Tong Xie, Chunying Zhou, and Huajun Chen. smash: semantic-based mashup navigation for data api network. In *18th International World Wide Web Conference (WWW2009)*, pages 1133–1134, 2009.
6. M. Maleshkova, C. Pedrinaci, and J. Domingue. Semantic annotation of Web APIs with SWEET. In *Proc. of 6th Workshop on Scripting and Development for the Semantic Web*, 2010.
7. Anne H. H. Ngu, Michael Pierre Carlson, Quan Z. Sheng, and Hye young Paik. Semantic-based mashup of composite applications. *IEEE Trans. on Services Computing*, 3(1):2–15, 2010.
8. Carlos Pedrinaci, Dong Liu, Maria Maleshkova, David Lambert, Jacek Kopecky, and John Domingue. iserve: a linked services publishing platform. In *Ontology Repositories and Editors for the Semantic Web (ORES2010)*, May 2010.
9. C. J. van Rijsbergen. *Information Retrieval.* Butterworth, 1979.

Link Prediction su Reti Multidimensionali

Giulio Rossetti Michele Berlingerio Fosca Giannotti
KDDLab, ISTI-CNR, Pisa, Italy
{name.surname}@isti.cnr.it

Sommario L'analisi di reti complesse è un campo di ricerca interdisciplinare, che vede coinvolti fisici, sociologi, matematici, economisti e informatici. In questo articolo estendiamo la formulazione classica del problema del Link Prediction allo scenario delle reti multidimensionali, ossia quelle reti che ammettono più di un link fra due entità. Introduciamo una nuova formulazione che tenga conto delle informazioni multidimensionali espresse dalle reti analizzate, e alcune famiglie di predittori progettati appositamente per sfruttare tali informazioni. Presentiamo infine una valutazione sperimentale dell'applicazione delle soluzioni proposte a reti multidimensionali reali. I risultati preliminari ottenuti sono incoraggianti, e spingono verso una ricerca più estensiva di soluzioni al problema del Link Prediction su reti multidimensionali.

1 Introduzione

Durante gli ultimi anni si è assistito al sorgere, nella comunità scientifica, di un grande interesse per l'analisi e l'estrazione di conoscenza dalle reti complesse che oggi dominano il mondo reale. Una delle direzioni di ricerca è legata all'analisi e la comprensione delle dinamiche evolutive che regolano nel tempo la struttura delle reti. Il tempo nelle reti può giocare un duplice ruolo: nel primo la struttura della rete evolve e si assiste all'apparizione di nuovi nodi o archi, mentre nel secondo, al passare del tempo, vengono compiute delle azioni da parte dei nodi (gli utenti di un social network si scambiano messaggi, ricercatori collaborano alla stesura di articoli, e via discorrendo). A causa della dinamicità delle reti, è comprensibile che l'interesse di numerosi lavori si sia incentrato sulla ricerca di pattern evolutivi che siano in grado di predire come queste evolvano nel tempo.

Recentemente, alcuni ricercatori si sono accorti che molte delle reti reali sono *multidimensionali*, ossia una coppia di nodi può essere connessa da più relazioni, che chiamiamo *dimensioni*. Differenti dimensioni possono rappresentare o tipi di relazione differenti (amicizia, parentela, ecc.), oppure diversi valori dello stesso tipo di relazione (come per esempio la collaborazione in diverse conferenze). Nel mondo reale esistono diversi esempi di reti multidimensionali, come la rete completa dei trasporti (dove treno, autobus, aereo, e nave sono quattro delle dimensioni possibili), le reti sociali (dove due persone possono essere connesse perché sono amiche, o giocano nella stessa squadra, o partecipano agli stessi eventi), o le reti di collaborazione (dove ogni conferenza è una dimensione diversa). Appare chiaro come questo tipo di reti, rispetto a quelle usualmente studiate in letteratura, presentino un ulteriore grado di libertà nella loro complessità, ossia le dimensioni. Diventa infatti interessante studiare le relazioni che intercorrono fra diverse dimensioni, l'importanza di una dimensione sulle altre, il fatto che due dimensioni si escludano a vicenda, e via discorrendo. In questo scenario, pensando al problema dell'analisi dell'evoluzione delle reti, una possibile domanda interessante da porsi è in quale o quali dimensioni apparirà più probabilmente un nuovo arco.

La multidimensionalità comporta la necessità di sostituire il modello usualmente utilizzato come base per l'analisi di reti, i grafi semplici, con uno più espressivo, i multigrafi. Come conseguenza di tale sostituzione, diviene necessario rivedere gli approcci algoritmici già studiati per i principali problemi di Data Mining in modo da tener conto delle ulteriori informazioni topologiche a disposizione per l'analisi. Congiuntamente alle informazioni inerenti la dimensionalità è interessante, soprattutto per gli studi incentrati sull'evoluzione delle reti, avere la possibilità di osservare, con il maggior dettaglio possibile, la cronistoria della rete oggetto di analisi. Una descrizione dettagliata della storia evolutiva di una rete consente, infatti, di raffinare i risultati ottenuti sfruttando una ancora più vasta fonte di informazioni.

In questo articolo ci occupiamo dell'evoluzione temporale di una rete multidimensionale introducendo un'estensione multidimensionale del problema del Link Prediction (la predizione di nuovi archi in una rete in evoluzione), e definiamo formalmente alcune classi di predittori che tengano conto delle informazioni di correlazione e anticorrelazione tra le dimensioni. Presentiamo quindi, brevemente, alcune reti multidimensionali, utilizzate per l'analisi sperimentale, e illustriamo i principali risultati preliminari ottenuti.

2 Definizione del problema

In [7], il problema del Link Prediction viene definito, data una rete sociale osservata ad un istante temporale t, come il problema di predire gli archi che si andranno ad aggiungere ad essa negli istanti successivi a t. Una volta introdotta la multidimensionalità tale formulazione del problema deve essere necessariamente estesa: le nuove informazioni fornite dalla rete introducono una maggiore complessità poiché i risultati proposti dai modelli predittivi devono riuscire a discriminare le dimensioni in cui compariranno gli archi predetti. Diamo di seguito una definizione formale del problema per l'ambito multidimensionale.

Definizione 1 (Link Prediction Multidimensionale) *Sia $G = (V, E, L, T)$ un multigrafo, non orientato, definito dagli insiemi finiti V dei nodi, E degli archi, L delle dimensioni e T degli istanti temporali associati agli archi.*

Il problema del Link Prediction, dato un multigrafo G ed un istante temporale $t' > max\{t : t \in T\}$ ha come obiettivo predire gli archi che entreranno a far parte del grafo originario in tale istante futuro e la specifica dimensione in cui essi si formeranno.

Per ogni arco predetto - identificato dalla tripla $(nodo, nodo, dimensione)$ - deve inoltre essere calcolato uno score *di confidenza della predizione.*

Come vediamo, quindi, la multidimensionalità aggiunge un grado di libertà al problema: non ci interessa solo sapere quali coppie di nodi si connetteranno in futuro, ma vogliamo anche sapere in quali dimensioni questo avverrà. In sezione 4 introduciamo alcune misure multidimensionali a supporto della soluzione al problema, mentre in sezione 5 introduciamo alcune famiglie di predittori.

3 Lavori Correlati

Molte pubblicazioni hanno trattato il problema del Link Prediction in reti monodimensionali e i principali modelli evolutivi sfruttati per effettuare un'analisi predittiva sulle reti sociali. Le soluzioni predittive analizzate in letteratura possono essere suddivise in *non supervisionate* e *supervisionate* a seconda che queste propongano formule chiuse, indipendenti dalla rete da analizzare, per effettuare la predizione, oppure guidino la fase predittiva tramite l'ausilio di informazioni topologiche estratte dalla rete stessa.

Fra i predittori non supervisionati, in [8] viene introdotta una soluzione basata sul principio del preferential attachment, mentre in [1] viene introdotto un modello basato invece sulle caratteristiche quantitative dei nodi comuni. Una buona rassegna dei modelli non supervisionati è [7], in cui gli autori confrontano empiricamente molti dei classici approcci di questo tipo. Nel nostro articolo, modifichiamo alcuni di questi predittori in modo da adattarli allo scenario multidimensionale.

Due predittori supervisionati sono invece [5] e [4], il primo dei quali, basato sull'estrazione di regole evolutive frequenti [2], non solo è anche in grado di predire nuovi nodi, ma utilizza tutta la storia temporale degli archi presenti nella rete.

In [6] gli autori presentano il problema di predire archi positivi (trust) e negativi (distrust). Sebbene questo possa sembrare una formulazione multidimensionale del Link Prediction, il problema affrontato è in realtà di classificazione, poichè viene predetta solo l'etichetta dell'arco.

4 Misure multidimensionali

Come abbiamo visto, nella letteratura del Link Prediction, molti predittori si basano su misure strutturali calcolate sulla rete. In analogia, i modelli che proponiamo in questa sezione sono basati su misure multidimensionali. Innanzitutto, quindi, vediamo qui alcune misure su reti multidimensionali su cui i nostri predittori sono basati. Per la natura preliminare di questo lavoro, presentiamo qui solo alcune delle misure utilizzate.

L'introduzione della multidimensionalità nel modello analizzato causa la necessità di rivedere parte delle misure utilizzate comunemente nell'analisi di reti: in particolare è necessario definire nuovamente i concetti di *neighbors* e di *degree* di un nodo. Mentre, infatti, in un grafo semplice non orientato, le due coincidono, ciò non accade più in un multigrafo.

Facendo riferimento a [3], in cui gli autori hanno introdotto un framework per l'analisi di reti multidimensionali, riportiamo una variante multidimensionale della misura di *neighbors*, una misura atta a cogliere l'importanza che una specifica dimensione può avere nella rete sulle altre, e due misure che contano la frazione di nodi e di archi presenti in una dimensione. Introduciamo, infine, una nuova misura multidimensionale, che estende il framework citato, mirata a pesare un arco in una dimensione relativamente alla sua storia temporale.

Definizione 2 (Neighbors$_{Xor}$) *Sia $v \in V$ un nodo di una rete G e $D \in L$ un insieme di dimensioni. La funzione $Neighbors_{Xor}$: $V \times P(L) \rightarrow N$ (dove $P(L)$ identifica l'insieme delle parti di L), definita come:*

$$Neighbors_{Xor}(v, D) = \sum_{u \in V} k_{uv}(D) \tag{1}$$

dove

$$k_{uv}(D) = \begin{cases} 1 \; se \; \forall (u, v, d) \in E : d \in D \\ 0 \qquad\qquad\qquad altrimenti \end{cases} \tag{2}$$

calcola il numero di vicini del nodo v raggiungibili tramite dimensioni in D, e non tramite archi etichettati con altre dimensioni.

Definizione 3 (Dimension Relevance Weighted) *Sia $v \in V$ un nodo in una rete G e $D \in L$ un insieme di dimensioni. La funzione Dimension Relevance Weighted: $V \times P(L) \rightarrow [0, 1]$, è definita come:*

$$DR_W(v, D) = \frac{\sum_{u \in N} Neighbors_{Set}(v, D) \frac{n_{uvd}}{n_{uv}}}{Neighbors(v, L)} \tag{3}$$

dove: $Neighbors_{Set}(v, D)$ identifica il numero dei nodi vici a v raggiungibili tramite archi etichettati da dimensioni appaetenenti all'insieme D, $Neighbors(v, L)$ identifica il numero dei nodi vicini a v raggiungibili tramite archi appartenenti ad una qualsiasi dimensione, n_{uvd} denota il numero di dimensinoni in cui compaiono archi tra i nodi u e v appartenenti a D, e n_{uv} denota il numero di dimensioni in cui compaiono archi tra u e v. Questa misura calcola la frazione di vicini direttamente raggiungibili dal nodo v seguendo archi appartenenti solo alle dimensioni incluse in D, pesata rispetto alle altre possibili dimensioni connettenti ciasun vicino.

Definizione 4 (*Ndd*) *Dati V l'insieme dei vertici, D l'insieme delle dimensioni (con $d \in D$) e E l'insieme degli archi della rete si definisce il coefficiente di Node Dimension Degree come:*

$$Ndd(d) = \frac{|\{u \in V | \exists v \in V : (u, v, d) \in E\}|}{|V|} \tag{4}$$

Definizione 5 (*Edd*) *Dati V l'insieme dei vertici, D l'insieme delle dimensioni (con $d \in D$) e E l'insieme degli archi della rete si definisce il coefficiente di Edge Dimension Degree come:*

$$Edd(d) = \frac{|\{(u, v, d) \in E | u, v \in V\}|}{|E|} \tag{5}$$

Definizione 6 (*$Wpres$*) *La funzione Wpres calcola il totale degli istanti - pesato in base all'ordine temporale - in cui un determinato arco è comparso nella rete nella dimensione specificata. Archi comparsi in istanti recenti hanno un peso maggiore.*

$$Wpres(u, v, d) = \sum_{\{t|(u,v,d,t) \in E\}} \Pi_t \tag{6}$$

dove Π_t indica il peso dell'istante temporale t.

5 Modelli Predittivi

Per affrontare il problema del Link Prediction sono stati proposti, in letteratura, molteplici modelli sia supervisionati che non supervisionati. La grande vastità e diversità dei modelli definiti deriva dall'aver ipotizzato, o estratto, differenti pattern evolutivi da eterogenee tipologie reti. In analogia, in questa sezione, introduciamo diverse classi di predittori da noi definiti. Per la natura preliminare di questo lavoro e per ragioni di spazio, introduciamo solo due famiglie di predittori. Facciamo notare, però, che è possibile combinare diversi predittori fra loro, oppure moltiplicarli per una o più delle misure sopra definite, in modo da catturare più segnali sovrapposti nell'evoluzione della rete. Questo è esattamente l'approcio seguito in [5], dove gli autori presentano diverse combinazioni fra il loro predittore basato sulle regole di evoluzione frequenti e alcuni predittori classici come Adamic-Adar e Common Neighbors.

Come prima soluzione, abbiamo modificato alcuni dei classici modelli non supervisionati (sono stati presi in considerazione Adamic-Adar, Common Neighbors e Jaccard) e li abbiamo adattati allo scenario multidimensionale. Per fare ciò, abbiamo semplicemente sostituito la misura $Neighbors$ da essi utilizzata, con la sua variante $Neighbors_{Xor}$ in modo da poter discriminare la dimensione degli archi predetti. In questo modo abbiamo creato una base sperimentale per il confronto dei successivi predittori, le cui performance sono state valutate rispetto a questi modelli.

Una soluzione più avanzata è stata quella di combinare i tre predittori base con le misure multidimensionali Ndd, Edd, e $Wpres$ (utilizzate come coefficienti moltiplicativi degli score) in modo da poter valutare se l'introduzione di ulteriori informazioni sulla struttura multidimensionale della rete riescono a garantire predizioni migliori.

L'ultima soluzione sperimentata è basata sulla misura DR_W introdotta in precedenza. Contrariamente a quanto fatto finora, non utilizziamo i modelli base nella predizione, ma ci affidiamo esclusivamente ad un modello ideato ad hoc e alle informazioni temporali fornite dalla rete.

Definizione 7 (*WDR*) *Siano $u, v \in V$ nodi della rete e $d \in D$ una dimensione: il predittore WeightedDimensionRelevance con informazioni di Weighted Presence è definito come:*

$$WDR(u, v, d) = DR_W(u, d) * DR_W(v, d) * (1 + Wpres(u, v, d)) \tag{7}$$

Durante la fase sperimentale abbiamo in realtà definito e provato diverse combinazioni di predittori, aggregati (non solo moltiplicazione), e coefficienti. Tuttavia, per ragioni di spazio e data la natura preliminare del lavoro, vediamo solo gli esperimenti relativi ai predittori sopra definiti.

6 Analisi Sperimentale

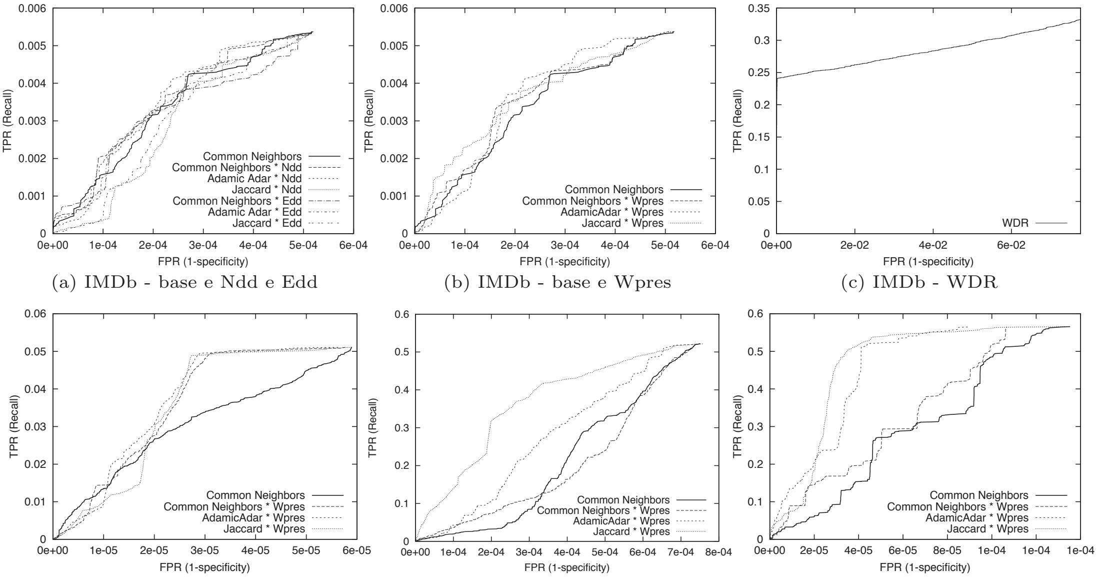
In questa sezione vediamo i risultati preliminari ottenuti applicando i nostri predittori ad alcune reti reali. Ciasuna rete è stata divisa in *training* e *test* set, e l'accuratezza dei predittori è stata misurata sul test set tramite curve ROC. Tale modalità di comparazione è stata scelta poiché consente una più facile lettura dei risultati ottenuti rispetto a quanto offerto dalle curve di Precision/Recall.

6.1 Reti Multidimensionali

Abbiamo utilizzato reti di diversa natura, ed in particolare:

- IMDb[1]: rete di attori cinematografici. Due attori sono connessi se hanno partecipato ad uno stesso film in uno stesso anno, e le dimensioni rappresentano la tipologia del film. Nodi: 43.867, dimensioni: 28. Il training set è composto da 216.544 archi appartenenti ad un arco temporale di 10 anni (1998-2007), il test set da 54.749 archi appartenenti all'anno successivo.
- DBLP[2]: rete di collaborazioni scientifiche. Due autori sono connessi se hanno partecipato alla stesura di un articolo in uno stesso anno, e le dimensioni sono definite dalle specifiche conferenze. Nodi: 30.176, dimensioni: 28. Il training set è composto da 78.956 archi appartenenti ad un arco temporale di 10 anni (1998-2007), il test set da 14.650 archi appartenenti all'anno successivo.
- GCD[3]: rete di fumettisti. Due fumettisti sono connessi se hanno partecipato alla stesura di uno stesso numero di un fumetto in un determinato arco temporale. Le dimensioni rappresentano la sezione del fumetto a cui hanno collaborato (ad es. copertina, storia). Nodi: 10.000, dimensioni: 6. Il training set è composto da 140.546 archi appartenenti all'arco temporale 1990-1999, il test set da 4.945 archi appartenenti all'anno successivo.
- GTD[4]: rete di gruppi terroristici. Due gruppi terroristici sono connessi se hanno pianificato un attentato nello stesso arco temporale in uno stesso stato. Gli stati rappresentano le dimensioni. Nodi: 2.755, dimensioni: 209. Il training set è composto da 25.200 archi, appartenenti all'arco temporale 1970-2007, il test set 2.572 archi appartenenti all'anno successivo.

[1] http://www.imdb.com
[2] http://dblp.uni-trier.de
[3] http://www.comics.org
[4] http://www.start.umd.edu/gtd



(a) IMDb - base e Ndd e Edd (b) IMDb - base e Wpres (c) IMDb - WDR

(d) DBLP - base e Wpres (e) GCD - base e Wpres (f) GTD - base e Wpres

Figura 1. Alcuni dei risultati ottenuti. Sull'asse delle ascisse è riportato i valori di FPR, mentre sulle ordinate i valori di TPR.

6.2 Risultati

In Figura 1 riportiamo le curve ROC calcolate sui risultati ottenuti sulle nostre reti. La prima riga mostra le differenze fra le varie famiglie di predittori sulla rete IMDb. In Figura 1(a) mostriamo l'andamento dei predittori di base moltiplicati per i coefficienti multidimensionali globali: utilizzando Common Neighbors come modello base di riferimento è possibile notare come gli incrementi nelle performance risultino contenuti seppur presenti. In Figura 1(b) abbiamo i predittori base estesi per l'analisi temporale tramite il moltiplicatore Wpres. In questo caso gli incrementi delle performance predittive risultano più sensibili. La Figura 1(c) riporta l'andamento del predittore WDR: in questo grafico si è omesso il confronto con i modelli base a causa della diversa scala utilizzata per la definizione del grafico. La seconda riga riporta invece i predittori base moltiplicati con Wpres, nelle altre tre reti. Come si può osservare, in queste reti la conoscenza multidimensionale e la storia temporale completa degli archi favorisce notevolmente l'accuratezza della predizione.

Un'analisi generale dei risultati suggerisce come l'andamento delle performance sia strettamente legato alla reale topologia della rete in esame. Questo risultato, per quanto non consenta la determinazione di un predittore *universale*, ossia di un modello con performance sempre superiori indipendentemente dalla rete analizzata, era attendibile data la tipologia dei modelli introdotti (non supervisionati basati su neighbors).

Quanto a Edge/Node Dimension Degree e Weighted Presence, sulle reti analizzate possiamo affermare che l'introduzione di moltiplicatori multidimensionali globali alla rete, e temporali locali ai singoli archi, consente di incrementare le performance dei modelli base rispettivamente nel 40-60%, per Ndd e Edd, e nel 60%, per $Wpres$, dei test eseguiti. Tale incremento varia a seconda del modello base a cui i coefficienti moltiplicativi sono applicati.

Tramite l'adozione del modello ad hoc WDR è possibile ottenere, per le reti analizzate, i migliori risultati sia per numero di predizioni corrette, sia per l'assegnazione degli score di confidenza ad ogni predizione.

7 Conclusioni e Lavori Futuri

Abbiamo presentato un'estensione del problema del Link Prediction allo scenario delle reti multidimensionali. Guidati da diverse misure multidimensionali topologiche e storiche, abbiamo definito diverse famiglie di predittori per queste reti. I risultati preliminari ottenuti sono incoraggianti, e sembrano confermare la validità dell'approccio, e l'effettiva potenza delle misure multidimensionali nel catturare fenomeni legati anche al modello evolutivo delle reti.

In futuro, ci proponiamo di estendere l'analisi sperimentale al fine di caratterizzare le reti secondo la loro prevedibilità temporale, tramite le misure e i predittori proposti. Inoltre, intendiamo verificare la fattibilità dell'introduzione di modelli supervisionati per risolvere il problema del Link Prediction su reti multidimensionali.

Riferimenti bibliografici

1. Lada A. Adamic and Eytan Adar. Friends and neighbors on the web. *Social Networks*, 25(3):211–230, 2003.
2. Michele Berlingerio, Francesco Bonchi, Björn Bringmann, and Aristides Gionis. Mining graph evolution rules. ECML PKDD '09, pages 115–130, Berlin, Heidelberg, 2009. Springer-Verlag.
3. Michele Berlingerio, Michele Coscia, Fosca Giannotti, Anna Monreale, and Dino Pedreschi. Foundations of multidimensional network analysis. Tech. Rep. 2010-TR-004. http://puma.isti.cnr.it/dfdownload.php?ident=/cnr.isti/2010-TR-004, 2010.
4. Mustafa Bilgic, Galileo Mark Namata, and Lise Getoor. Combining collective classification and link prediction. ICDMW '07, pages 381–386, Washington, DC, USA, 2007. IEEE Computer Society.
5. Björn Bringmann, Michele Berlingerio, Francesco Bonchi, and Arisitdes Gionis. Learning and predicting the evolution of social networks. *IEEE Intelligent Systems*, 25(4):26–35, 2010.
6. Jure Leskovec, Daniel P. Huttenlocher, and Jon M. Kleinberg. Predicting positive and negative links in online social networks. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *WWW*, pages 641–650. ACM, 2010.
7. David Liben-Nowell and Jon Kleinberg. The link prediction problem for social networks. CIKM '03, pages 556–559, New York, NY, USA, 2003. ACM.
8. David M. Pennock, Gary W. Flake, Steve Lawrence, Eric J. Glover, and C. Lee Giles. Winners don't take all: Characterizing the competition for links on the web. *PNAS*, 99(8):5207–5211, April 2002.

Spatio-temporal intersection of trajectories under uncertainties

Paolino Di Felice and Matteo Orsini

Department of Electrical and Information Engineering,
University of L'Aquila, L'Aquila (Italy)
paolino.difelice@ing.univaq.it, matteoorsini@yahoo.it

Abstract. The paper focuses on the spatio-temporal intersection of pairs of trajectories (i.e., time ordered sequences of points). It starts from the sketch of a linear algorithm already appeared in the literature working under the assumption that the exact position of the moving points is known at each time instant. Unfortunately, such an hypothesis is not appropriate in most real cases because of the many sources of uncertainty which undermine our knowledge about the actual position of the moving points. Hence, a detailed algorithm that abandons such an assumption is formalized, implemented and evaluated against synthetic data. A template trajectory database is adopted to prove the effectiveness of the intersection operator in real applications. Short conclusions end the paper.

1 Introduction

With the growing number of mobile applications, made possible by the wide-spread use of cheap devices that capture positions, data analysis on large sets of historical moving objects trajectories becomes increasingly important. Moving objects databases have been the subject of intensive research since the end of the nineties.

If only the position in space of an object is relevant, then the moving point (m-point, in the following) is the basic abstraction. Examples of m-points are vehicles, people, or animals moving on the earth ground. M-points databases follow into two categories: one representing current movements and the other representing histories of movements. In this paper, we refer to databases of the second kind, often called trajectory databases.

A lot of research about m-objects databases has been carried out by Güting and his colleagues (see, for instance, [1] and [2]). Their work, relevant both from the methodological and theoretical point of view, unfortunately does not help to face the needs common to most enterprises when they try to develop today real time applications. In fact, Güting and his colleagues at the implementation level concentrated their efforts on the development from scratch of the SECONDO system [3]. The design of this prototype ignores the provided spatial extensions of existing DBMSs, moreover SECONDO does not conform to the OGC standards as it does not follow any available data model and, as such, it is not embeddable into the DBMS infrastructure of an organization, where lot of static spatial data is already stored.

On the opposite, the ongoing HERMES project [4] aims at extending the capabilities of existing DBMSs in order to aiding designers and developers of spatio-temporal databases in managing every day applications about moving objects that change location, shape and size in time. Specifically, the spatio-temporal functionality of HERMES are provided as a data cartridge on top of Oracle. Our work shares the same vision and motivations at the basis of the HERMES project.

In [1] it was introduced the so called sliced representation to represent the trajectories drawn by m-points. The basic idea of such a modeling method is to decompose a trajectory into a set of temporal ordered units called slices such that within a slice the exact position of the m-point can be calculated at any time instant by some kind of "simple" function. This approach is perfect in a "perfect world". Unfortunately the real life is much more complex and, in particular, it is permeated by the uncertainty. The relevance of the uncertainty topic in the arena of m-points databases is proven by several papers (e.g., [5], [6], [7], [8], [9], [10] and [11]).

The contributions of our work are the following:

- we revisit the spatio-temporal intersection algorithm of pairs of trajectories originally sketched in [12] in such a way that its output somehow takes into account the uncertainty that usually limits our knowledge about the exact position of m-points at a generic instant time. To cope with such an uncertainty the proposed solution returns a set of time intervals instead of a set of specific timestamp values;
- the algorithm is implemented (as a user defined function) on top of the PostgreSQL open-source DBMS enhanced with the PostGIS spatial extension and evaluated by running experiments with synthetic data. The numerical results show that the proposed solution is extremely fast and, hence, of practical interest;
- as the final step, a template trajectory database is implemented to give evidence of the expressiveness of the intersection operator to carry out basic spatio-temporal analysis.

2 Background

As said in the Introduction, the background of the present paper goes back to the pioneer work made by Güting and colleagues. In particular in [1] they introduced the concept of sliced representation, the basic idea of which is to decompose the temporal development of a moving value into a set of temporal units called slices. To each slice is associated a unit defined as the pair {I, f(t)}, where I is a time interval and f(t) is some kind of "simple" function (e.g., linear) that models the movement of the m-point inside I.

Fig.1 shows an example of a m-point consisting of three temporally ordered units. Within each unit, the evolution is linear at constant speed and it is described by the function $f(t)=(x_0+x_1t,\ y_0+y_1t)$, where $x_0,x_1,y_0,y_1 \in \mathbb{R}$, that allows to trace the (x,y) coordinates taken by the m-point at any instant $t \in I$. Note that the units cannot temporally overlaps, yet gaps are possible (i.e. periods during which the position of the m-point is undefined).

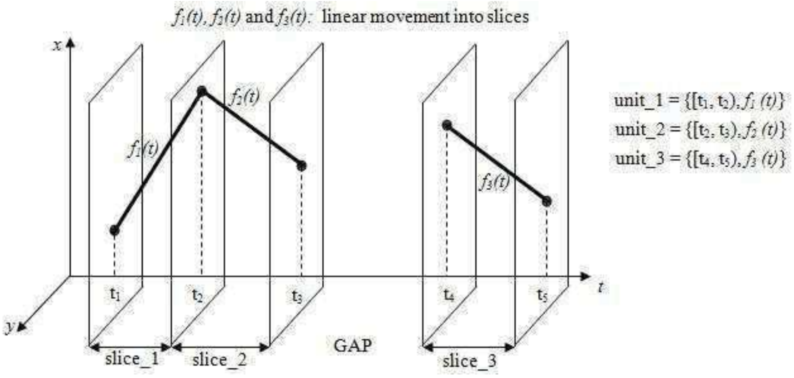

$f_1(t)$, $f_2(t)$ and $f_3(t)$: linear movement into slices
x
y
t
$f_1(t)$
$f_2(t)$
$f_3(t)$
unit_1 = {[t_1, t_2), f_1 (t)}
unit_2 = {[t_2, t_3), f_2 (t)}
unit_3 = {[t_4, t_5), f_3 (t)}
t_1
t_2
t_3
t_4
t_5
slice_1
slice_2
GAP
slice_3

Fig. 1. The sliced representation of a moving point

In [2] a large set of spatio-temporal operators was formally defined; here we focus on one of them (intersection) whose signature is:

moving(point) × moving(point) → moving(point),

where moving(point) is a data type. In [12] the authors sketched a possible algorithm of such an operator. In the following, we recall it.

The sequences of units that make up the two trajectories involved in the spatio-temporal intersection are preliminarily synchronized with an operation named refinement partition, which is obtained by breaking the units into other units that have the same value of f(t) but are defined on smaller time intervals, so that a resulting unit of the first argument and one of the second argument are defined either on the same time interval or on two disjoint time intervals. This means that through each point of any segment of both segment sequences we place a plane parallel to the x-y-axis obtaining a certain number of slices. As shown in Fig.2, for each slice two situations can happen: either a slice only contains one partial segment from one sequence (white zone) or two partial segments can be found (grey zone). In the former case there can be no intersection. In the latter case the segment intersection test is checked in constant time, using the well-known plane sweep algorithm [13]. If the two segments intersect at the point (x^*,y^*), then the unit $\{[t^*,t^*], f(t)=(x^*,y^*)\}$ is added to the result, where the interval I coincides with the instant t^* in which the m-points occupied the intersection point. If the two segments share a line described by the function $f^*(t)$, then the unit $\{[t_{ini},t_{end}), f(t)=f^*(t)\}$ is added to the result.

Let n and m be the number of consecutive segments of the two trajectories, the intersection computation traverses both sequences until the end of one sequence is reached. Each segment is considered exactly once since the time synchronization can be performed on the fly. This leads to a run-time cost of $O(n+m)$.

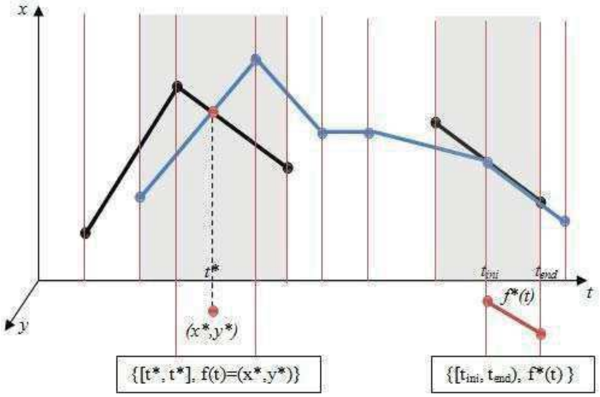


Fig. 2. A pictorial representation of the intersection algorithm

3 Sources of uncertainty about m-points

Working with m-points the following sources of uncertainty come into the picture:

- uncertainty about the knowledge of the position of the m-points over time. The main sources are: a) uncertainty in the knowledge of the m-point motion law (because of: traffic condition, weather condition, the kind of way on which the movement takes place - city street, provincial road with many bends, mountain road with lots of ups and downs, highway, and the speed limits to take care of); b) measurement errors, c) computational errors, and masking of the exact position of m-points due to privacy/anonymity reasons (e.g., [8]).
- Uncertainty in the reconstruction of the trajectory of the m-point.

Because of the manifold sources of uncertainties just listed, it is appropriate to state that is fond to assume the correctness of the where-when values returned by the computation of the spatio-temporal intersection of pairs of trajectories. We think to be more suitable to return an answer that admits a certain degree of flexibility. It is fair to remark that the fact that Güting and his colleagues model the m-points without taking into account the uncertainty must not surprise because their model dates back to the end of nineties, and at that time the uncertainty issue was not at the centre of the research agenda.

The choice taken in this paper is middle course: we assume that the where value is correct, while a temporal interval is returned to denote when the rendezvous between the two m-points may be occurred. Circumstance, this latter, not verifiable in any way, after renouncing to the assumptions at the basis of the sliced representation.

By assuming correct the geometric position of the rendezvous of two m-points is equivalent to ignore the uncertainty caused by the measurement errors, the computational errors, and the uncertainty in the reconstruction of their trajectories. On the other side, by adopting a time interval to denote the temporal window when the spatio-temporal intersection may be occurred is equivalent to take somehow into account the uncertainty caused by ignoring the m-points' motion law whose effects are definitely more vast.

We conclude this section by discussing a little bit more how the m-point's motion law issue impact on the spatio-temporal intersection problem. As an example, let us refer to Fig.3 and let us assume that the temporal relation existing among the points 1, 2, 3, and 4 is that of Fig.4. In other words, let us suppose that the interval [t_3..t_4] in which the m-point mpB moved from position 3 to position 4 partially overlaps the interval [t_1..t_2] in which mpA moved from 1 to 2. If we assume to know the motion law of mpA and mpB when they move from positions 1 to 2 and from 3 to 4, respectively, then it is possible to compute if they temporally met in Z or not, simply recurring to some Physics's law to be embedded into the unit function inside the corresponding interval I. Otherwise, we lose all the certainties and, consequently, we cannot state anything.

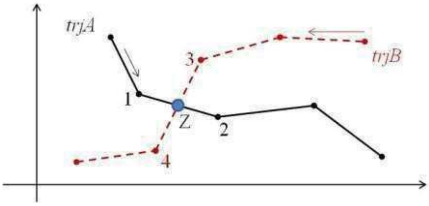


Fig. 3. Two geometrically intersecting trajectories (projected on the x-y Cartesian plane)

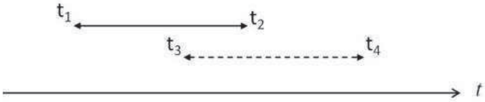


Fig. 4. A temporal relationship between m-points mpA and mpB

As said above, in this paper we renounce to the knowledge of the motion law of the m-points because in the reality it is mostly unpredictable. But, we adopt a computational method that allows to break the stalemate. In practical terms, with regard to Fig.3 and the hypothesis taken above (Fig.4), the answer to the "whether" issue is "yes", while the answer to the "when" issue is expressed in terms of the temporal interval [t_3..t_2] where such an event falls, if it happens, circumstance, this latter, not provable analytically any more.

If we take into consideration the fact that the timestamps linked to the points making up the trajectories in the database are those for which the m-point position is acquired, it follows that the extent of the interval [t_3..t_2] is less or equal to the extent of the acquisition interval. In practical terms, we can say that for most real applications this value is a matter of minutes and, hence, absolutely satisfactory.

4 Spatio-temporal intersection algorithms

The algorithm to calculate the spatio-temporal intersection of pairs of trajectories detailed below (and named time_meet) is a revised version of the algorithm sketched in Sec.2. The most important novelty we introduce here concerns the treatment of the intersection points (x*,y*), that are no longer associated with a single timestamp t* but with a time interval $[t_{ini},t_{end})$, specifying the period in which it may have taken place the rendezvous between the two m-points (in the sense and for the reasons given in Sec.3). On the opposite, the intersections whose geometry has dimension 1 are treated as in [12].

In the following, a generic trajectory consists of a time ordered sequence of points:

$\{<P_1, t_1>,<P_2, t_2>, \ldots, <P_{n+1}, t_{n+1}>\}$ (i.e. $t_1<t_2<\ldots<t_{n+1}$).

In turn, a generic point (P_i) is described by the pair $<x_i,y_i>$ denoting its geographic position expressed in a reference system (eg.: WGS84), while $<t_i=t(P_i)>$ is the corresponding timestamp (shortened as TS in the sequel). The t-value adds semantics to the knowledge of the pure geographic position of the m-point, offering a richer support to the decisions makers. As usual, we model the geometry of a trajectory as a linestring (shortened as LS in the sequel), that is, as a curve with linear interpolation between points. The generic pair of consecutive points P_iP_{i+1} defines a line segment, over the time interval $[t_i,t_{i+1})$.

The time_meet algorithm scans all the pairs of line segments whose intervals temporally overlap then, for each of them, it checks (by means of the spatial operators `st_intersects(geometry1,geometry2)` and `st_intersection (geometry1,geometry2)` - see [14]) whether and where the two participant segments spatially intersect. The first operator (`st_intersects()`) assesses if it takes place the intersection between two input geometries and returns true in the affirmative case, false otherwise. The second operator, instead, returns a geometry that represents the shared portion of the two geometries. Both such operators use the well-known plane sweep algorithm able to achieve in constant time the result, when the geometries are two line segments.

The time_meet algorithm follows.

Algorithm time_meet (in1 integer, in2 integer)

Input: in1 and in2 identify the pair of trajectories involved in the spatio-temporal test

Output: table result(IdTraj1 integer, IdTraj2 integer, intersection_geometry geometry, initial_time timestamp with time zone, final_time timestamp with time zone)

Method:

1. Let $lsA=\{A_1, A_2, \ldots, A_{n+1}\}$ and $lsB=\{B_1, B_2, \ldots, B_{m+1}\}$ be the LSs in the database identified by in1 e in2, respectively.
2. Let $tsA=\{t_k \mid t_k=t(A_k)$, where A_k belongs to lsA and $k=1, 2,\ldots, n+1\}$ and $tsB=\{s_j \mid s_j=t(B_j)$, where B_j belongs to lsB and $j=1, 2,\ldots, m+1\}$ be the set of TSs of the points of lsA and lsB, respectively.

```
3.  FOR EACH pair {[$t_i$, $t_{i+1}$), [$s_j$, $s_{j+1}$)} of overlapping time intervals detected in a
    synchronized scan of tsA and tsB DO
4.      IF (st_intersects($A_iA_{i+1}$, $B_jB_{j+1}$)) THEN
5.          geom = st_intersection($A_iA_{i+1}$, $B_jB_{j+1}$)
6.          $t_{ini}$ = max{$t_i$, $s_j$}
7.          $t_{end}$ = min{$t_{i+1}$, $s_{j+1}$}
8.          Add (in1, in2, geom, $t_{ini}$, $t_{end}$) to the table result
9.      END IF
10. END FOR
11. return result
12. END time_meet
```

Using the ordering of the line segments with respect to the time, the search of the not temporally disjoint pairs of segments can be carried out by a synchronized scan (row 3) of the two trajectories involved in the test. With regard to the example of Fig.5, the spatial intersection test (row 4) on the first temporally overlapping couple of segments ({A_4A_5, B_1B_2}) fails. To determine the next pair of segments to be tested (row 3), the algorithm checks which one of the two segments ended first and it advances on the trajectory it belongs to (in the example, we advance on trjA and test the pair {A_5A_6, B_1B_2}). When the segments spatially intersect (as in the case of {A_5A_6, B_3B_4} and {A_6A_7, B_5B_6}) the time_meet algorithm first calculates (row 5) the shared intersecting geometry (point (x*,y*) and line B_5A_7 of Fig.5), then it computes (rows 6-7) the temporal window [t_{ini},t_{end}) ([s_3,t_6) and [s_5,t_7) of Fig.5). The spatial-temporal intersection so determined is added to the result (row 8). The algorithm halts when the end of one of the two trajectories is reached and, hence, all possible pairs {[t_i, t_{i+1}), [s_j, s_{j+1})} have been taken into account.

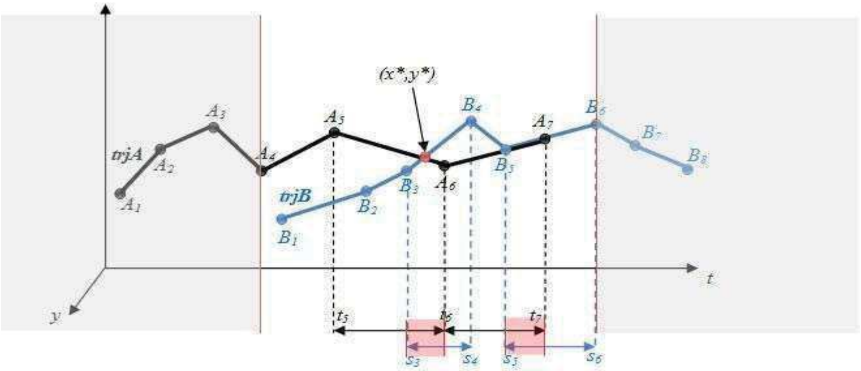


Fig. 5. Two trajectories involved in the time_meet test

The algorithm performs all the computations in a single scan of the two trajectories. Since the IF-THEN block of instructions can be executed in constant time (O(1) - the arguments of the `st_intersects()` and `st_intersection()` operators are line segments) and the time synchronization can be performed on the fly, the time_meet can be executed in O(n+m), where n and m indicate the number of line segments of the two input trajectories.

As final consideration, we note that the intersection computation algorithm can be easily modified to calculate the intersection test of pair of trajectories, that is to construct an algorithm (t_meet) that returns true if it is detected at least one spatial-temporal intersection, false otherwise.

5 Implementation and evaluation of the time_meet algorithm

For the purposes of carrying out the experiments reported in the following, we refer to the single-table database:

```
trajectory (
   Pkey: integer, Shape: geometry,
   TimeValues: timestamp with time zone ARRAY )
```

As implementation platform we chose the PostgreSQL open-source DBMS (vers. 8.4.3) enhanced with the PostGIS spatial extension (vers. 1.4.2). On such a platform, the creation of the previous table requires two steps: in the first step a no-spatial table is created, hence the spatial column is added through the invocation of the postGIS `addGeometryColumn` constructor.

In order to carry out the campaign of experiments, the `trajectory` table was populated with synthetic tuples generated by an “ad hoc” PL/pgSQL function which makes use of the well-known `random()` function to calculate the <x,y> coordinates of the points. The values of the TSs (in charge of the built-in function `now()`) start when the generating function begins the execution and, at each iteration, an increment of 10 minutes takes place. The function receives as input parameter an integer equal to the number of line segments of the trajectory to be generated.

The algorithm time_meet (as well as t_meet) has been implemented as a User Defined Function called `time_meet()` (`t_meet()`) on top of the postgreSQL/postGIS. The idea of implementing the proposed algorithms as UDFs to be added to the built-in UDFs of the system has the double benefit of making them available for being called from any queries as well as from external software applications that connect to the database.

Table 1 shows the response time of the function `time_meet()` for increasing values of n and m. The measurements were carried out on a Sony Vaio equipped with the Pentium Dual Core T4200 processor, 2Ghz and 4 GB RAM.

Table 1. Summary of the campaign of experiments

n	m	`time_meet()`
50	50	0.3 sec
50	500	2.8 sec
500	500	9 sec

As we can see from the table, the CPU-times are very low. A further improvement of the performances can be achieved by modifying the time_meet algorithm by preceding the parallel scan of the two trajectories with their time alignment. Such an operation consists in identifying those portions of the two trajectories that have no overlapping time intervals. Since the points are time ordered, such a research can be done in logarithmic time without impacting on the overall computational complexity of the `time_meet()` operator. With respect to Fig.5, the time alignment excludes from the intersection test the portion of trjA on the left of point A_4 and the portion of trjB on the right of point B_6, by reducing the number of line segments involved in the test, so saving CPU time.

6 A template database and example queries

This section gives evidence of the effectiveness of the proposed operators to conduct the spatio-temporal analysis. The section starts by defining a very simple database composed of three tables, followed by their feeding with few tuples, hence two queries are posed and their output is shown. The scripts that follow comply with the SQL of the postgreSQL/postGIS DBMS.

The database tables:

```
trajectory (…) (see Section 4);
pointOfInterest (
   ID: integer, Location: geometry, Description: varchar(200) );
areaOfInterest (
   ID: integer, Boundary: geometry, Description: varchar(200) )
```

Besides the first table (already introduced in Sec.5), the other two add further semantics to the "template" database enhancing the spatial analysis. In fact, in actual applications, `pointOfInterest` and `areaOfInterest` can be used to store, respectively, a set of relevant landmarks (e.g., petrol stations, churches, museums) and areas (e.g., downtown restricted areas, recreation areas, natural parks).

The SQL population scripts:

```
INSERT  INTO trajectory (Pkey, Shape, Timevalues)
VALUES  (100, 'LINESTRING(21 2,15 2, 10 6, 3 6, 1 11)'::GEOMETRY,
          '{"2010-10-21 09:00:00", "2010-10-21 09:10:00",
            "2010-10-21 09:20:00", "2010-10-21 09:30:00",
            "2010-10-21 09:40:00"}'::timestamp with time zone ARRAY),

        (101, 'LINESTRING(6 9, 8 3, 15 3, 9 9, 4 2)'::GEOMETRY,
          '{"2010-10-21 08:50:00","2010-10-21 09:00:00",
            "2010-10-21 09:10:00", "2010-10-21 09:20:00",
            "2010-10-21 09:30:00"}'::timestamp with time zone ARRAY)

INSERT INTO pointOfInterest(ID, Location, Description)
VALUES  (1,'POINT(10 2)', NULL), (2,'POINT(4 4)', NULL),
        (3,'POINT(12 10)',NULL), (4,'POINT(200 200)', NULL)

INSERT INTO areaOfInterest(ID, Boundary, Description)
VALUES  (22, 'POLYGON(5 5, 11 2, 13 8, 4 8, 5 5)'::GEOMETRY, NULL)
```

Fig.6 shows the geometry of "the scene" (projected on the x-y Cartesian plane) that reflects the content of the example database we refer to in this section. The point of interest (200 200) does not appear in the figure because it falls out of scale.

The proposed queries make use of `time_meet()` and `t_meet()` (besides the spatial operators `st_distance()` and `geometry_contain()` – [14]). Thanks to those operators, the SQL formulation is kept simple in both cases.

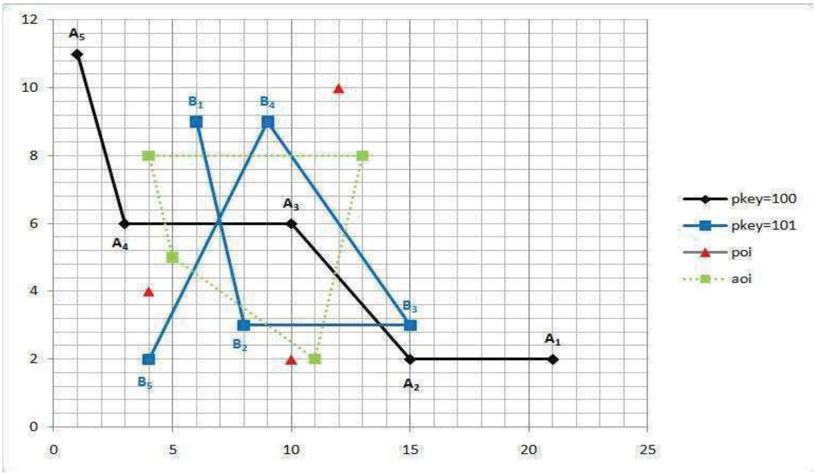
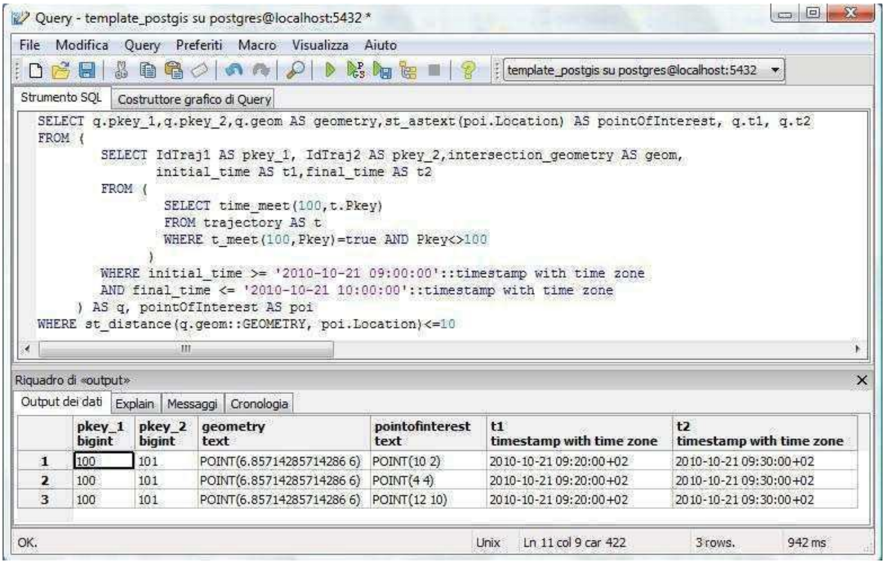


Fig. 6. The geometry of the reference "scene"

Q1

Given a trajectory (e.g., that identified by the p.k. 100), retrieve from the database the p.k. of the remaining ones that share, with the target trajectory, at least a spatio-temporal intersection in the interval bounded by the TSs 2010-10-21 09:00:00 and 2010-10-21 10:00:00. Display the spatio-temporal intersections falling within the given time interval, as well as the points of interest situated within a distance of 10m from those intersections.


Query - template_postgis su postgres@localhost:5432 *
File
Modifica
Query
Preferiti
Macro
Visualizza
Aiuto
template_postgis su postgres@localhost:5432
Strumento SQL
Costruttore grafico di Query
Riquadro di «output»
Output dei dati
Explain
Messaggi
Cronologia

```
SELECT q.pkey_1,q.pkey_2,q.geom AS geometry,st_astext(poi.Location) AS pointOfInterest, q.t1, q.t2
FROM (
        SELECT IdTraj1 AS pkey_1, IdTraj2 AS pkey_2,intersection_geometry AS geom,
               initial_time AS t1,final_time AS t2
        FROM (
                SELECT time_meet(100,t.Pkey)
                FROM trajectory AS t
                WHERE t_meet(100,Pkey)=true AND Pkey<>100
               )
        WHERE initial_time >= '2010-10-21 09:00:00'::timestamp with time zone
        AND final_time <= '2010-10-21 10:00:00'::timestamp with time zone
    ) AS q, pointOfInterest AS poi
WHERE st_distance(q.geom::GEOMETRY, poi.Location)<=10
```

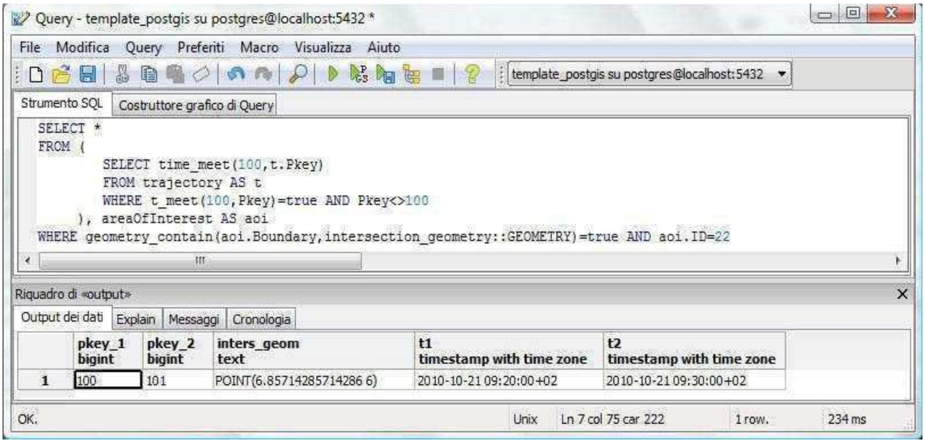
	pkey_1 bigint	pkey_2 bigint	geometry text	pointofinterest text	t1 timestamp with time zone	t2 timestamp with time zone
1	100	101	POINT(6.85714285714286 6)	POINT(10 2)	2010-10-21 09:20:00+02	2010-10-21 09:30:00+02
2	100	101	POINT(6.85714285714286 6)	POINT(4 4)	2010-10-21 09:20:00+02	2010-10-21 09:30:00+02
3	100	101	POINT(6.85714285714286 6)	POINT(12 10)	2010-10-21 09:20:00+02	2010-10-21 09:30:00+02

OK. Unix Ln 11 col 9 car 422 3 rows. 942 ms

The output of query Q1 shows that between the trajectory identified by the p.k. 100 and that identified by the p.k. 101 there exists a spatio-temporal intersection. The rendezvous between the two m-points could be occurred between the 09:20:00 and the 09:30:00 of the 21st of October 2010 at point of geometry (6.85714 6). Furthermore, we can see that there are three points of interest within a distance of 10m from the intersection point.

Q2

Given a trajectory (e.g., that identified by the p.k. 100), retrieve from the database the p.k. of the remaining ones that share, with the initial one, at least a spatio-temporal intersection inside a certain area of interest (let say that having ID=22 in the table `areaOfInterest`). Moreover, for each couple of trajectories, show where and when they met inside such an area.


Query - template_postgis su postgres@localhost:5432 *
File
Modifica
Query
Preferiti
Macro
Visualizza
Aiuto
template_postgis su postgres@localhost:5432
Strumento SQL
Costruttore grafico di Query
Riquadro di «output»
Output dei dati
Explain
Messaggi
Cronologia

```
SELECT *
FROM (
        SELECT time_meet(100,t.Pkey)
        FROM trajectory AS t
        WHERE t_meet(100,Pkey)=true AND Pkey<>100
     ), areaOfInterest AS aoi
WHERE geometry_contain(aoi.Boundary,intersection_geometry::GEOMETRY)=true AND aoi.ID=22
```

	pkey_1 bigint	pkey_2 bigint	inters_geom text	t1 timestamp with time zone	t2 timestamp with time zone
1	100	101	POINT(6.85714285714286 6)	2010-10-21 09:20:00+02	2010-10-21 09:30:00+02

```
OK.                    Unix   Ln 7 col 75 car 222   1 row.   234 ms
```

The interpretation of the output of query Q2 trivially follows from that of Q1.

7 Conclusions

The `time_meet()` and `t_meet()` spatio-temporal operators were defined, implemented and evaluated. Technically they were added as UDFs to the PostgreSQL DBMS equipped with the PostGIS spatial extender. `t_meet()/time_meet()` can play, in the context of m-points databases, the same role plaid by the `st_intersects()/ st_intersection()` (being part of OGC standard from many years now) in the context of applications on top of standard spatial databases.

The choice of the previous two operators, primarily motivated by their practical usefulness, offers also the chance to reaffirm the necessity to ensure adequate support, at the DBMS level, to spatio-temporal applications whose relevance is constantly grown in the last years (e.g., [15]). In absence of an adequate repertoire of operators about m-points, the negative repercussions at the application level are essentially two:

greater development difficulties and worst performances. Two shortcomings to be avoided.

References

1. Forlizzi, L., Güting, R. H., Nardelli, E., and Schneider, M.: A Data Model and Data Structures for Moving Objects Databases. In Proc. ACM SIGMOD International Conference on Management of Data: 319-330, 2000.
2. Güting, R. H., Bohlen, M. H., Erwig, M., Jensen, C. S., Lorentzos, N. A., Schneider, M., and Vazirgiannis, M.,: A foundation for representing and querying moving objects. ACM Transactions on Database Systems, 25(1), 1-42, 2000.
3. Güting, R. H., Behr, T., and Düntgen, C.: SECONDO: A platform for Moving Objects Database research and for publishing and integrating research implementations. IEEE Data Engineering Bulletin 33(2), 56-63, 2010.
4. Pelekis, N., Frentzos, E., Giatrakos, N., Theodoridis, Y.: HERMES: aggregative LBS via a trajectory DB engine. In Proc. of the ACM SIGMOD International Conference on Management of data, 2008.
5. Trajcevsky, G., Wolfson, O., Hinrichs, K., and Chamberlain, S.: Managing uncertainty in moving object databases. ACM Transactions on Database Systems, 29(3), 463-587, 2004.
6. Abul, O., Bonchi, F., and Nanni, M.: Never walk alone: uncertainty for anonymity in Moving Object Databases. In Proc. of the International Conference on Data Engineering, 2008.
7. Frentzos, E., Gratsias, K., and Theodoridis, Y.: On the Effect of Location Uncertainty in Spatial Querying. IEEE Transactions on Knowledge and Data Engineering, 21(3), 2009.
8. Giannotti, F. and Pedreschi, D.: Mobility, Data Mining and Privacy. Springer, 2008.
9. Trajcevski, G., Choudhary, A., Wolfson, O., Ye, L., and Li, G.: Uncertain Range Queries for Necklace. 11th IEEE International Conference On Mobile Data Management, 199-208, 2010.
10. Othman W.: Uncertainty Management in Spatio-temporal Databases, PhD Thesis in Theoretical Computer Science, Hasselt University, May 2009.
11. Kuijpers B. and Othman W.: Trajectory databases: Data models, uncertainty and complete query languages. Journal of Computer and System Sciences. 76(7), 538-560, 2010.
12. Cotelo Lema, J.A., Forlizzi, L., Güting, R. H., Nardelli, E., and Schneider, M.: Algorithms for Moving Object Databases. The Computer Journal, 46(6), 680-712, 2003.
13. Nievergelt, J., Preparata, F. P.: Plane-Sweep Algorithms for Intersecting Geometric Figures. Com. of the ACM, 25(10), 1982.
14. OpenGIS Implementation standard for geographic information. Simple feature Access, Part 2: SQL Option (ref. number: OGC 06-104r4), 2007.
15. Di Felice, P., Liguori, G., and Cestra, G.: Un’architettura software per lo sviluppo di applicazioni riguardanti dati spazio-tempo dipendenti. Congresso annuale AICA, L’Aquila, 2010 - ISBN/ISSN: 978-88-905406-0-8.

Poster Track

Assessing the effectiveness of "Precise" Activity Diagrams in the Context of Business Process Modeling

Francesco Di Cerbo[1], Gabriella Dodero[1] Gianna Reggio[2], Filippo Ricca[2], and Giuseppe Scanniello[3]

[1] CASE, Libera Università di Bolzano-Bozen, Italy,
francesco.dicerbo|gabriella.dodero@unibz.it
[2] DISI, Università di Genova, Italy,
filippo.ricca|gianna.reggio@disi.unige.it
[3] Dipartimento di Matematica e Informatica, Università della Basilicata, Italy,
giuseppe.scanniello@unibas.it

Abstract. UML activity diagrams are a commonly used notation for modeling business processes. In this paper, we present a novel precise style for this notation and a controlled experiment to assess its effectiveness. The context of the experiment is constituted of master students in Computer Science at the Free University of Bolzano-Bozen in Italy. The results indicate that the subjects achieved a significantly better comprehension level when business processes are modeled using the precise style with respect to a "lighter" variant, with no significant impact on the effort to accomplish the comprehension tasks.

Keywords: Business Process Modeling, UML activity diagrams, Controlled experiment, Precise and Ultra-light styles.

1 Introduction

To be competitive in the global market, many organizations have been changing their business processes [4]. In this context, modeling, management, and enactment of business processes are considered relevant to support organizations in their daily activities. Regarding the modeling of business processes, a number of process definition languages — such as BPMN [9], event-condition-action mechanisms [1], graph rewriting mechanism [5] and Petri Nets [2] — have been proposed in the literature.

More recently, the use of the Unified Modeling Language (UML) [11] has been suggested in the context of business process modeling. This trend is mainly due to the fact that UML has been conceived for the communication among people and so it may represent a natural choice for such a kind of modeling [8]. Further, in favor of the UML notation there is its flexibility that allows choosing the preferred degree of precision/abstractiveness to express models. For example, different options are available ranging from "light" styles, where nodes and arcs of the activity diagrams are decorated by natural language text, to more rigorous ones, where nodes and arcs might be expressed in a formal language. "Light" activity diagrams are simple to write/use but their inherent ambiguity could complicate the communication among participants. On the other hand, more precise/rigorous notations are more complex to use but limit ambiguity and they can be transformed into executable models (e.g., expressed in BPEL) more easily.

In this paper, we sketch our *precise* UML activity diagrams to model business processes. This style has been proposed and used in the context of the TECDOC project[1] together with other variants: *ultra-light*, *light*, and *conceptual precise*. In order to compare the proposed precise style with a lighter variant (precisely the *ultra-light* one) a controlled experiment with master students in Computer Science has been conducted and the results have been presented in the paper.

2 Process modeling with UML

2.1 Basic Terminology

The basic activities in a business process are the *basic task* of the process, while the *process objects* are the entities over which an activity of the process is performed. The active entities that perform the tasks are *process participants*, and whenever it will be relevant, we will distinguish the human participants from those corresponding to software and hardware systems.

2.2 Ultra-light style

The *ultra-light* style is the one often used in the industry for business process modeling, see e.g., [7]. Following the *ultra-light style* a process is modeled by a UML activity diagram, where nodes (activity and object) and guards on the arcs leaving the decision nodes are decorated by natural language text and usually they follow neither rules nor patterns.

Participants of the process may be modeled only by introducing swimlanes and titles of the various lanes. The objects produced and consumed by the activities of a business process may be optionally made explicit by using object nodes.

In Fig. 1, we present the *ultra-light* UML model of the business process Process Order, namely one of the experimental objects used in the empirical investigation presented here. It is a parametric activity diagram that receives as input the Requested Order (see the node at the boundary of the activity diagram) for an on-line shop. Tasks are represented in the model as rounded rectangles, while produced process objects are shown as rectangles. The activity diagram describes how the order is managed by the on-line shop. It is quite easy to understand and there is no need to further comment it.

[1] the TECDOC project, funded in the framework of research activities of Ligurian Technology District SIIT, aimed to define methodologies to efficiently schedule, coordinate, monitor and manage Complex Organizations.

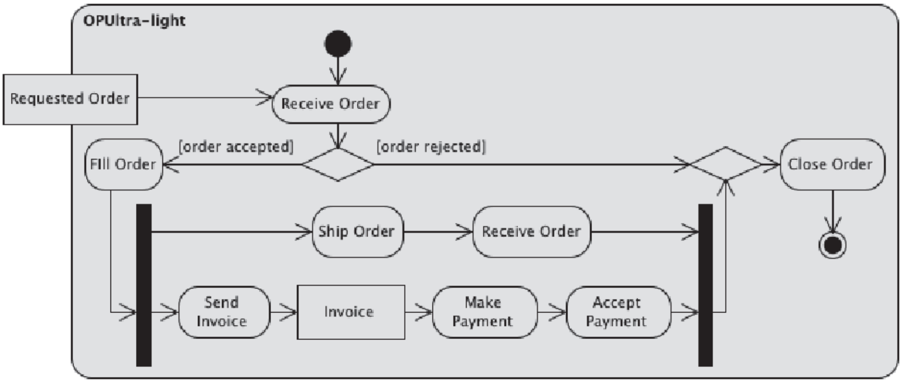


Fig. 1. Ultra-light model of Process Order

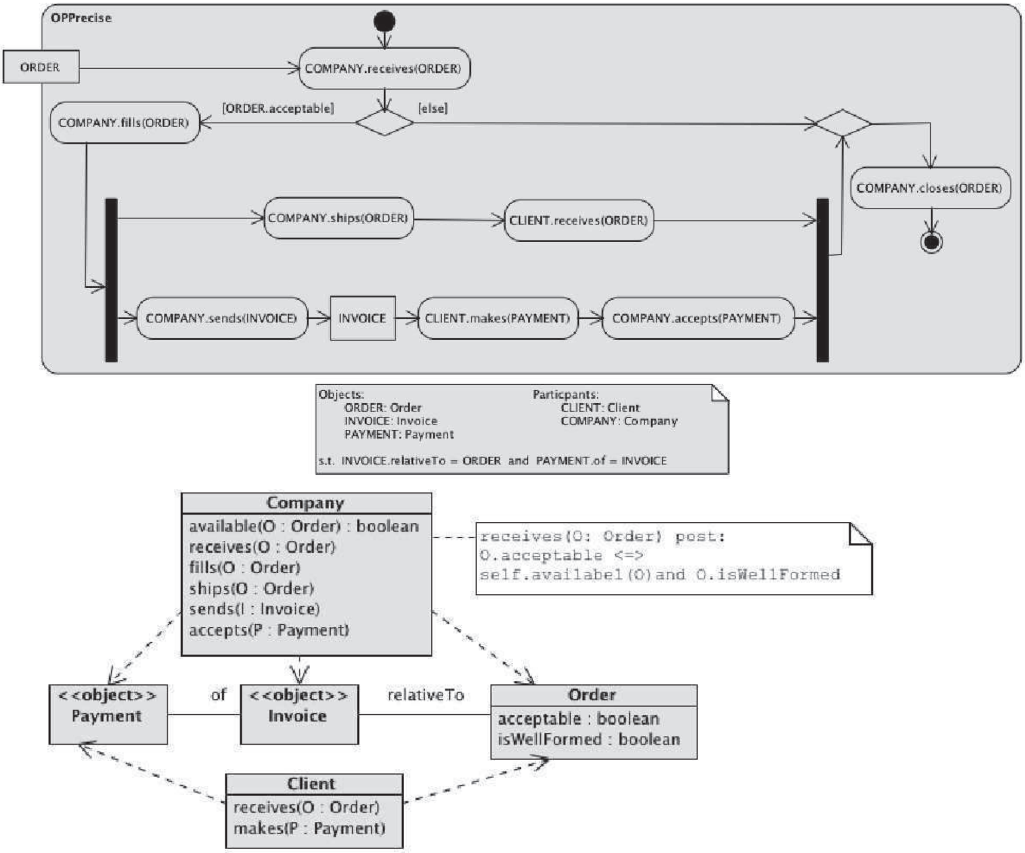


Fig. 2. Precise model of Process Order (activity and class diagram)

2.3 Precise Style

The participants and the objects of a business process modeled by the precise style are explicitly listed and precisely modeled with UML by means of classes. This represents the most remarkable difference with respect to the ultra-light style. Conversely, the behavioral view of the process is given by an activity diagram where actions and conditions will be written by using respectively: the language for the action of UML and OCL [10] (the textual language for boolean expressions part of UML). Whenever the object nodes will be used, they should be typed by UML classes and data types; and if swim-lanes are used, they should be given titles by participants.

Fig. 2 shows the precise model of the Process Order case. In the process we have two participants (both of kind human beings) the Client and the Company, and three business objects: ORDER, PAYMENT and INVOICE. The three objects are related among them as shown by the constraints in the participants/objects box (see the box on the bottom of Fig. 2). The flow of the business object ORDER is shown by using its name in the various action nodes, whereas the flow of INVOICE has been emphasized by using an object node.

The class diagram in Fig. 2 introduces the classes typing the participants and the objects with their relevant operations and attributes, together with their mutual relationships. For example we can see that a payment and an invoice are relative to exactly one order. The dashed arrows, i.e., the dependency relationships denote that the company may work on the payments, the invoices and the orders, whereas the client may manage only the payment and the order. Constraints may be used to finely describe the various classes. For example, the constraint on the operation receives of class Company (see the note in Fig. 2) expresses in OCL that an order is considered acceptable by the company if and only if it is well-formed and available.

3 Experimentation Setup

In this section we highlight the design of the experiment following the guidelines proposed by Wohlin *et al.* in [14]. For replication purposes, the experimental package (in English) and the raw data are made available on the Web[3].

3.1 Context and Hypotheses Formulation

The experiment was conducted with 26 master students in Computer Science at the Free University of Bolzano-Bozen. This experiment represented an optional educational activity of two Software Engineering courses: Infrastructures for Open Service Oriented Architectures and Requirements and Design of Software Systems. As mandatory laboratory activity of the former course, the students had previously individually developed Web services using specification documents that included UML models in terms of class, sequence, and activity

[3] www.scienzemfn.unisa.it/scanniello/BPM/

diagrams. Students of the course Requirements and Design of Software Systems had already made use of UML in the design of a non-trivial software system.

The perspective of this study is twofold. From the point of view of researchers, it is an investigation of the effectiveness of using precise activity diagrams in the specification of business processes; and from the point of view of project managers, it is an evaluation of the possibility of adopting this style. Accordingly, we have defined and tested the following null hypotheses:
- H_{lo}: The use of precise activity diagrams **does not significantly improve** the comprehension level of the subjects to perform a task.
- H_{to}: There **is no significant difference** in terms of effort when using precise or ultra-light activity diagrams to perform a comprehension task.

The objective of the statistical analysis is to reject the null hypotheses, thus accepting the corresponding alternative ones that can be easily derived. H_{l0} is one tailed [14] since we expect a positive effect of the precise style on the process comprehension, while H_{t0} is two-tailed since we cannot postulate an expectation.

3.2 Design

We adopted a counterbalanced design [14] as shown in Table 1. We considered four groups: A, B, C, and D. Each group was formed by subjects randomly selected (precisely: 7 subjects for groups A and D; 6 for groups B and C). Each subject worked on two comprehension *Tasks* (i.e., Task 1 and Task 2) on the following two experimental *Objects*: Process Order (PO) and Document Management Process (DMP). Each time subjects used the precise or the ultra-light activity diagrams.

The selected business processes refer to application domains on which the subjects were familiar with. PO is for processing orders in an on-line shop (see Fig. 1). Conversely, DMP is in charge of managing the review process of any kind of document (e.g., recipes for culinary dishes).

3.3 Selected Variables

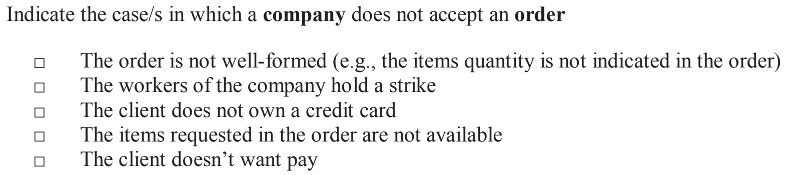
In this experiment, the only independent variable is *Treatment*, which is a nominal and admits two values: *Precise* and *Ultra-light*. Further, we selected the following dependent variables to test the null hypotheses: *comprehension level* and *comprehension effort*.

The *comprehension level* dependent variable is used to measure the comprehension of the subjects on each business process model. To this end, we asked the subjects to answer a comprehension questionnaire (one for each experimental object) composed of multiple choice questions. Fig. 3 shows a sample question (Question 8) concerning the comprehension questionnaire of PO.

The provided answers have been measured using an information retrieval based approach. In particular, the correctness and the completeness of the answers have been measured, similarly to [13], using *precision* and *recall*, respectively. In order to get a single value representing a balance between correctness and completeness, we used the F-measure, i.e., the harmonic mean of precision

	A	B	C	D
Task 1	PO Precise	PO Ultra-light	DMP Precise	DMP Ultra-light
Task 2	DMP Ultra-light	DMP Precise	PO Ultra-light	PO Precise

Table 1. Experiment design



Indicate the case/s in which a **company** does not accept an **order**

- □ The order is not well-formed (e.g., the items quantity is not indicated in the order)
- □ The workers of the company hold a strike
- □ The client does not own a credit card
- □ The items requested in the order are not available
- □ The client doesn't want pay

Fig. 3. A sample question of the comprehension questionnaire for PO

and recall values. For example, if a student had answered to Question 8 of the PO task (Fig. 3) picking the first, second and fifth answer, where the correct answers are only the first and the fourth ones, the precision value will be 0.33 (three answers given and only one correct) while the recall value will be 0.5 (one correct answer out of two). The F-measure value will be then 0.39.

The overall comprehension level achieved by each subject, which assumes value ranging from 0 to 1, has been computed using the overall average of the F-Measure values of all the questions. A value close to 1 indicates a very good understanding of the business process, while a value close to 0 indicates a very bad comprehension level.

The *comprehension effort* variable measures the time, expressed in minutes, that each subject spends to accomplish a task. We got the comprehension effort values using the start and stop times the subjects were asked to record.

3.4 Execution and Experimental Material

We asked the subjects to use the following procedure to execute both the comprehension tasks: *(i)* specify name and start-time in the comprehension questionnaire; *(ii)* answer independently the questions by consulting the provided material; *(iii)* mark the end-time of the task.

To perform the experiment the subjects were provided with the following hard copy material: *(i)* a summary of the modeled business process, *(ii)* the comprehension questionnaires and the models of the business processes, *(iii)* a unique post-experiment questionnaire[2] to be filled in after the two tasks.

4 Analysis and Results

In this section, the results of the data analysis are sketched quickly with respect to the defined null hypotheses. In all the performed statistical tests, we decided

[2] For space reasons, the results of the post-experiment questionnaire are not presented.

	Precise			Ultra-Light				
Object	mean	med	σ	mean	med	σ	MW test	Wilcoxon test
All	0.79	0.84	0.11	0.62	0.66	0.14	< **0.001**	<**0.001**
DMP	0.76	0.74	0.10	0.64	0.64	0.10	**0.005**	-
PO	0.80	0.84	0.11	0.58	0.69	0.19	**0.003**	-

Table 2. Descriptive statistics of comprehension level and statistical test results.

(as it is customary) to accept a probability of 5% of committing Type-I-error [14].

4.1 Comprehension level

Table 2 reports some descriptive statistics (i.e., mean, median, and standard deviation) of comprehension level and the results of statistical analysis conducted on the gathered data. Because of the sample size (26 subjects) and mostly non-normality of the data we adopted non-parametric tests to test the first null hypothesis. In particular we selected Mann-Whitney test for unpaired analysis and Wilcoxon test for paired analysis. We used these tests since they are very robust and sensitive [14].

The one-way unpaired Mann-Whitney test ($p - value < 0.001$) and the one-way paired Wilcoxon test ($p - value < 0.001$) provide evidence that there exists a significant difference in terms of comprehension level between the two treatments. Therefore, in general, we can reject the null hypothesis H_{l0}. The mean comprehension level improvement achieved with precise diagrams is of 17 points (see means of the "All" row in Table 2), i.e., 27,41%. Similar results can be observed for the primary measures. For space reasons we report only results of Mann-Whitney tests: precision ($p - value < 0.001$) and recall ($p - value < 0.001$) and for both objects, DMP ($p - value = 0.005$) and PO ($p - value = 0.003$).

4.2 Comprehension effort

Students with precise diagrams employed more time that students with ultra-light diagrams. Means and medians are respectively: 22'16" and 20 minutes for precise diagrams; 22'12" and 19'50" minutes for ultra-light diagrams. A two-tailed unpaired Mann-Whitney test returned 0.9 as $p - value$. A similar value is returned by paired Wilcoxon test ($p - value = 0.6$). Therefore we cannot reject the overall null hypothesis H_{t0}. Even analyzing the two objects separately no significant difference was found. The results of the unpaired two-tailed Mann-Whitney test were 0.56 and 0.57 for DMP and PO, respectively.

5 Conclusion

UML activity diagrams provide an intuitive and easy way to express business processes models [3], [8], [6]. However, their effectiveness is mostly not assessed by means of controlled experiments. Indeed, only a few other studies perform empirical evaluations in business process formalisms comparisons. An example is [12], where a comparison between UML and BPMN is presented.

In this paper, we have proposed a precise style for the UML activity diagrams in the context of business process modeling. The effectiveness of this style has been investigated with respect to a less rigorous style by using a controlled experiment. The results of this investigation indicated a significant effect of the precise style on the comprehension (+27.61%), whith no impact on the effort.

References

1. L. Aversano, A. De Lucia, M. Gaeta, P. Ritrovato, S. Stefanucci, and M. L. Villani. Managing coordination and cooperation in distributed software processes: the genesis environment. *Software Process: Improvement and Practice*, 9(4):239–263, 2004.
2. S. Bandinelli, E. Di Nitto, and A. Fuggetta. Supporting cooperation in the spade-1 environment. *IEEE Trans. Softw. Eng.*, 22:841–865, December 1996.
3. A. De Lucia, R. Francese, and G. Tortora. Deriving workflow enactment rules from uml activity diagrams: a case study. *Symposium on Human-Centric Computing Languages and Environments*, 0:211–218, 2003.
4. H. E. Eriksson and M. Penker. *Business Modelling with UML*. Wiley Computing Publishing, 2000.
5. P. Heimann, G. Joeris, C. Krapp, and B. Westfechtel. Dynamite: Dynamic task nets for software process management. In *Proceedings of the International Conference on Software Engineering*, pages 331–341, 1996.
6. S. Jurack, L. Lambers, K. Mehner, G. Taentzer, and G. Wierse. Object flow definition for refined activity diagrams. In *Proceedings of the 12th International Conference on Fundamental Approaches to Software Engineering*, pages 49–63, Berlin, Heidelberg, 2009. Springer-Verlag.
7. R. Monfared, A. West, R. Harrison, and R. Weston. An implementation of the business process modelling approach in the automotive industry. *Journal of Engineering Manufacture*, 216(11):1413–1428, 2002.
8. E. D. Nitto, L. Lavazza, M. Schiavoni, E. Tracanella, and M. Trombetta. Deriving executable process descriptions from UML. In *Proceedings of the 22rd International Conference on Software Engineering*, pages 155–165, 2002.
9. OMG. Business process model and notation (BPMN) Version 2.0. *OMG Final Adopted Specification, Object Management Group*, 2006.
10. OMG. Object constraint language (OCL) specification, version 2.2. Technical report, Object Management Group, February 2010.
11. OMG. Unified modeling language (OMG UML) specification, version 2.3. Technical report, Object Management Group, May 2010.
12. D. Peixoto, V. Batista, A. Atayde, E. Borges, R. Resende, and C. Pádua. A Comparison of BPMN and UML 2.0 Activity Diagrams. In *VII Simposio Brasileiro de Qualidade de Software*, 2008.
13. F. Ricca, M. Di Penta, M. Torchiano, P. Tonella, and M. Ceccato. The role of experience and ability in comprehension tasks supported by uml stereotypes. In *29th International Conference on Software Engineering (ICSE 2007), Minneapolis, MN, USA, May 20-26, 2007*, pages 375–384, 2007.
14. C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell, and A. Wesslén. *Experimentation in Software Engineering - An Introduction.* Kluwer, 2000.

Repositories of conceptual schemas: concepts, constructs, methods and quality dimensions[1]

Carlo Batini, Marco Comerio, Enrica Pasqua, and Gianluigi Viscusi
[batini, comerio, pasqua, viscusi] @ disco.unimib.it

Dipartimento di Informatica, Sistemistica e Comunicazione (DISCo),
Università di Milano-Bicocca, Milan, Italy

Abstract. Repositories of conceptual schemas have been proposed in literature to represent data managed in complex large scale information systems. In this paper we discuss concepts, constructs, methods and quality dimensions for repositories of conceptual schemas. In particular, the paper considers the two main state of the art paradigms to organize huge amounts of heterogeneous data: integration, to catch the resolution of heterogeneities among data, and abstraction, to enable representation of data at different levels of synthesis. Furthermore, in this paper we provide a formalization of all these issues, and investigate several quality dimensions of a repository.

1 Introduction

In a modern organization there can be operating hundreds and even thousands of databases developed and managed in different times and by different teams, resulting in an extremely fragmented view of the overall information content managed in the organization. In this paper we discuss concepts, constructs, methods and quality dimensions proposal for repositories of conceptual schemas. The discussion is grounded on the framework for repository design and development initially investigated in [1], where the central idea was to iteratively make use of abstraction/refinement primitives for bottom-up/top-down generation of schemas, together with integration/diversification primitives for unification/diversification of schemas characterized by different types of heterogeneities.

The paper is structured as follows. In Section 2 we revisit and compare previous and present researches and experiences on repositories of schemas that, for what concern the authors of the paper go back to twenty years ago. In Section 3 we provide a new formalization of the concepts involved in repositories, namely abstract and integrated schema, refinement/abstraction and integration/diversification primitives. Section 4 introduces several quality dimensions and metrics that can be adopted to

[1] This work is partially supported by the SAS Institute srl (Grant Carlo Grandi) and by the Italian MoSeForAgroFood project, funded by the Lombardy region.

represent and analyze the quality of a repository. We conclude the paper discussing future works.

2 Related work

Large organizations with a long history and a consequent large set of legacy information systems need instruments to face conceptual change and schema evolution management and maintenance of very large set of conceptual schemas. At the state of the art, these challenges have been considered in terms of i) clustering of schemas [2], ii) hierarchical structuring of schemas [3], iii) methods an techniques for conceptual schema analysis and classification [4], and iv) application of integration-abstraction primitives for building dictionaries and repositories of schemas [1, 5].

As to clustering, it can take different names such as typology, numerical taxonomy, and partitioning [6], and applying to different domains such as epidemiology, cellular manufacturing, linguistics, data mining, and economics. Furthermore, at the state of the art Entity-Relationship clustering algorithms and approaches are still mainly semiautomatics, asking for "experts" involvement [7]. As a consequence, there is still a poor clustering assistance in diagramming tool.

Primitives for schema integration are introduced in [8], where a methodology for schema integration in the Entity Relationship model is presented. Integration and abstraction primitives have been introduced in [1], where several properties of the repositories adopting such primitives have been formally modelled and investigated. Abstractions in conceptual modelling have been studied to support database design, database comprehension and schema summarization [4], formal characterizations of generic relationships [9], and, recently, theories of ontology granularity [10].

Different approaches to repository management have been proposed in the literature: remarkably, a system to manage conceptual models represented in different languages, among which UML, and Web ontologies represented in RDFS and OWL-DL, is presented in [11]. Finally, heuristic methodologies and tools for efficient production of repositories reusing other repositories are presented in [12]. Usage of repositories in eGovernment initiatives is discussed in [5].

3 Concepts, constructs and methods for repositories of schemas

A *conceptual schema S* is a set of entities, relationships, generalizations, IS-A relationships in the Entity Relationship model with usual meanings [13]. Without loss of generality we will consider only binary relationships; an n-ary relationships R_n defined among entities $E_1, .., E_n$ can be transformed in a new entity R_n connected with n binary relationships to $E_1, .., E_n$. We will not consider cardinalities and other constraints, and identifiers.

A *repository of conceptual schemas* is a set of schemas, representing at conceptual level the whole information content of the databases of an organization. A schema may be basic, abstract, or integrated.

A *basic schema* is a schema resulting from the conceptualization of a database of the organization. Schemas in the repository are placed in *levels*. Basic schemas constitute the level 0 of the repository. Besides basic schemas, other schemas in a repository can be generated by two types of transformation primitives, namely i) *abstraction*, that has in input a single schema and generates in output a single schema and ii) *integration*, that has in input *n* schemas and generates in output a single schema, under the constraint that each schema in the repository is in input to only one transformation and there is only one schema that is in input to no transformation (the repository is a tree). Both abstraction and integration have associated inverse operations (namely, *refinement* and *diversification*), in which the roles of input and output schema(s) are exchanged.

An *abstract schema* AS = abstraction (S) is a schema resulting from the application of *abstraction primitives* to a schema S in the repository. Inversely, a *refined schema* RS = refinement (S) is a schema resulting from the application of refinement primitives to a more abstract one. If the level of schema *S* is *i,* the level of a corresponding abstract schema AS is *i+1*.

An *integrated schema* IS = integration ($S_1, S_2, ..., S_n$) is a schema resulting from the *integration* of *n* schemas $S_1, S_2, ...,S_n$. The integration process, deeply investigated in the literature [13, 14], can be expressed in terms of two activities, namely: i) discovering heterogeneities in schema matching, and ii) applying *integration primitives*, namely transformations performed on schemas to be integrated in order to produce the reconciled integrated schema.

We assume that all concepts of $S_1, S_2, ...,S_n$ are represented in *S*. Similarly to abstraction, also integration has an inverse conceptual operation, named *diversification*, that transforms a schema S into a set of schemas $S_1, S_2, ..., S_n$ with a process that makes use of *diversification primitives*, that have a complementary role with respect to integration primitives. In order to simplify the structure of repositories, we will assume in the following that one schema in the repository can be involved in only one abstraction or integration. Concerning the level of integrated schemas, if the levels of schemas $S_1, S_2, ...,S_n$ are *level1, level2, ..., leveln*, the level of IS is equal to *max (level1, level2, ..., leveln) + 1*.

Two types of structures can appear in the repository: *refinement chains* (or, inversely, *abstraction chains,* we prefer from now on to adopt the refinement instead of abstraction as the relevant transformation), that can be made of one to n *refinement* (*abstraction*) *steps* among schemas, and *integration (diversification) trees* and *steps,* with similar meanings.

Due to the above organization, the repository can be described at two levels of detail, we may distinguish the *high level structure* and the *full structure* of a repository. The high level structure is made of (i) refinement chains and schemas involved, and (ii) integration trees and schemas involved. The full structure includes refinement primitives used in refinement steps and concepts involved in refinement primitives. Codes are assigned to schemas, refinements chains and integration trees according to the following rules:

1. Refinement chains are identified by a code R_i placed on top of the chain; schemas in refinement chains of length *n* are assigned codes R_{i1}, R_{i2}, …, R_{in}.

2. Integration steps are identified by a code I_k placed on the bottom of the integrated schema in the step; the *n* source schemas involved in the integration step are assigned codes I_{k1}, I_{k2}, I_{kn}.
3. Double codes are assigned to schemas that are both the top schema in a chain and a source schema in an integration step.

The above rules for codes allow us to associate levels to the set of refinement chains and the set of integration steps, considered as independent structures in the schema; we adopt a top-down enumeration. E.g. when only refinement chains are considered, refinement chain R1 has level 1 and refinement chain R2 has level 2.

We have now to make more precise the concept of abstraction/refinement primitive. Here we prefer to proceed top-down, starting our formalization with the concept of refinement primitive (*r-primitive* in brief); the formalization of the inverse concept of *abstraction primitive* is straightforward.

A *refinement primitive* is made of two parts, called respectively the *source schema SS* and the *target schema TS*.

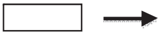
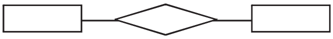
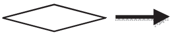
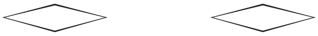
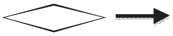
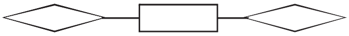
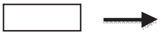
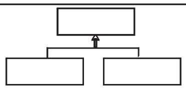
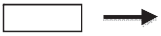
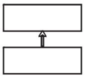
Code	Type of primitive	Source schema	Target schema
rp_1	Entity expanded into a Relationship between two (n) Entities		
rp_2	Relationship expanded into two (n) Relationships		
rp_3	Relationship expanded into an Entity and two Relationships		
rp_4	Entity expanded into a Generalization between two (n) entities		
rp_5	Entity expanded into an IS-A relation between two Entities		

Figure 1: Basic refinement primitives

We assume in the following that *SS* is made of a unique concept, entity or relationship. When an abstraction primitive is applied to a schema *S*, the concepts of *S* that match *SS* are substituted by the concepts in *TS*; the primitive has to specify which concepts in *TS* inherit the links of *SS* in *S*, and the names of concepts in *TS*. Among all possible types of primitives, we distinguish a subset characterized by a simple structure, and we call them *basic r- primitives*. They are represented in Figure 1. Notice that relationships in primitives rp_1, $rp_{2,}$ rp_3 may be n-ary.

Complex refinement primitives are refinement primitives whose structure in terms of left hand schema and/or right hand schema may be general. We consider and use a subset of complex primitives, those ones that have in the left hand side schema a single concept, entity or relationship.

3.1 Methods for repository generation

As for single conceptual schemas, there are basic design patterns also for repository design. We analyze three design patterns: bottom-up, top down, and mixed.

A *bottom up process* proceeds as shown in Figure 2, with a divide and conquer strategy, from granular low-level schemas to high-level compact and homogenised schemas made of abstract concepts. Notice that here we use the term bottom-up for the whole process of repository generation.

```
Given a set of basic schemas $S = [S_1, S_2, ..., S_n]$
1. Perform
  •  Cluster schemas in $S$ in $m$ groups
  •  For each group produce an integrated schema $IS$
  •  For each integrated schema, produce an abstract schema
Until it is feasible to integrate schemas in one step
2. Produce the final integrated schema
3. Perform
     •  Produce an abstract schema
Until the abstract schema is considered adequately compact
```

Figure 2: Bottom-up process for repository design

In a *top-down process* (see Figure 3) a fully integrated abstract view of all concepts in basic schemas is ideally available at the beginning of the process: the process proceeds with refinement or diversifications until all basic schemas are reached.

```
Given a set of basic schemas $S = [S_1, S_2, ..., S_n]$
Generate a high level schema with less than ten macro concepts
Perform
  •  For each schema that is a leaf of the generation process, decide whether to
     refine it or else to diversify it into n schemas.
  •  If refine then
          •  Using r-primitives produce a refined schema
          Else
          •  Choose n, and using diversification primitives produce a set of $n$
             diversified schemas
Until all basic schemas are leafs of the generation process.
```

Figure 3: Top-down process for repository design

A *mixed process* proceeds alternatively with top-down and bottom-up steps. The issue of choosing the "best" process for a specific organization and its basic schemas $S_1, S_2, ..., S_n$ is a relevant one, and needs the specification of the concept of "best" through suitable quality dimensions and metrics.

4 Qualities of repositories

A repository of conceptual schemas may be characterized in terms of several quality dimensions and related metrics. Single schema dimensions have been investigated in the literature, see [15, 16]. The most investigated qualities of schemas are i) *readability*, the property of the schema to express the meaning of the reality represented in a clear way for its intended use; ii) *correctness*, the correct representation of requirements in terms of the model categories; iii) *minimality*, every aspect of requirements is represented only once in the schema; iv) *completeness*, the extent to which a schema includes all the concepts in the requirements; v) *maintainability*, the property of the schema of being easily changed when requirements change.

When we move from one schema to a repository of schemas, we may use the above quality dimensions for single schemas, while we have to introduce new qualities that characterize the structure of the repository and the generation process of schemas in the repository by means of abstractions (refinements) or integrations (diversification). Such aspects of structure and of generation processes may concern consecutive schemas in refinement chains, or else repositories and their views. We define now quality dimensions of a repository, and classify them according to the distinction we made between qualities referred to the high level structure of the repository and qualities referred to the full structure.

Qualities pertaining the high level structure of the repository are *global schema balancing*, *schema refinement balancing, local integration balancing*, and *global integration balancing*.

Global schema balancing is a property of the high level structure of the repository, and requests that schemas different from basic schemas have a similar size, where the size can be expressed by the number of concepts (entities, relationships, etc.) in the schema. Basic schemas have sizes that depend historically on the relevance and evolution of the corresponding databases. When moving from basic schemas to higher level schemas, this property says that we should proceed toward achieving homogeneity in the complexity of schemas generated, expressed by their size. *Schema refinement balancing* refers to the homogeneous disposition of schemas in refinement chains. *Local integration balancing* is a property of integration steps in the high level structure, and requests that the schemas involved in the integration step are of similar size. It can be seen as a subproperty of global schema balancing. *Global integration balancing* is a property of integration chains, and requests that the schemas in the different integration steps are of similar number. Global integration balancing captures a different quality of the high level structure of the repository than previous qualities.

We now discuss qualities referred to the full structure of the repository, and especially on the distribution and complexity of refinement (abstraction) and integration (diversification) balancing; they are: *concept refinement balancing*, *global and local refinement balancing*, and *understandability*.

Concept refinement balancing concerns the homogeneous refinement of high level concepts in refinement chains, from abstract schemas to the concrete ones. While concept refinement balancing makes reference to the balancing of refinement chains of the whole repository, *global and local refinement balancing* make reference to the repository seen in its relationship with the views that have been produced referring to single concepts or else single sectors of the organization. A view associated to a concept, e.g., in the upper level schema, shows the full set of refinements and diversification primitives that the concept undergoes.

Finally, *understandability* can be defined as the effort required to the users of the repository to understand the generative process, where low effort corresponds to high understandability.

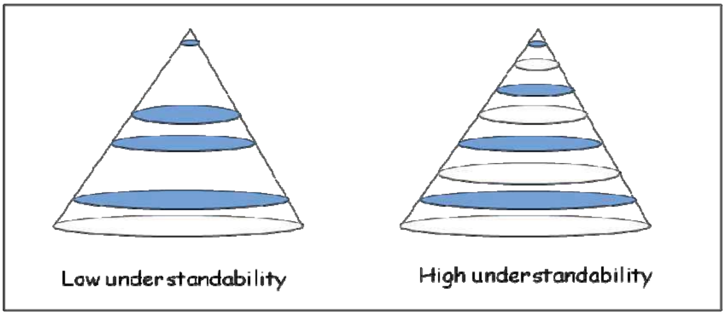


Figure 4: Two cases of low and high understandability

In Figure 4 we see two cases of low and high understandability. Here the intuition is that in the left hand side refinement chain has a discontinuous refinement process, and in some cases proceeds with simple refinements, in other with complex ones.

5 Conclusion and Future work

In this paper we have discussed concepts, methods and qualities of repositories of conceptual schemas under a unified perspective. Conceptual schemas are a relevant asset for organizations facing the today inedited large amount of available data, allowing to have a representation of the managed concepts and their relevance for the organization activities. Nevertheless, the amount of conceptual schemas is often related to the number of databases in an organization and their changes in time.

Due to these issues, an organization having hundreds and even thousands of databases, and willing to build an integrated representation of the conceptual content of the whole set of databases, needs both for abstracted and integrated representations of all concepts in databases, due to the complexity involved in the matching, homogenisation and merging activities of large amount of schemas. Thus, in this paper we point out the relevance of repository of conceptual schemas for these issues, whereas a systematization of state of art concepts and methods is required.

The major contribution of the paper concerns, on the one hand, the systematization and extension of state of the art concepts and methods; on the other hand, the introduction and discussion of quality dimensions of repositories.

In future work the discussed apparatus consisting in refinement/abstraction and integration/diversification primitives will be considered all together to model new types of quality dimensions of repositories and metrics for the ones presented in this paper. Furthermore, future work will be also devoted to analyze the important issue of the costs of repository generation.

References

1. Batini, C., Di Battista, G., Santucci, G.: Structuring primitives for a dictionary of entity relationship data schemas. Software Engineering, IEEE Transactions on 19, 344-365 (1993)
2. Akoka, J., Comyn-Wattiau, I.: Entity-relationship and object-oriented model automatic clustering. Data & Knowledge Engineering 20, 87-117 (1996)
3. Shoval, P., Danoch, R., Balabam, M.: Hierarchical entity-relationship diagrams: the model, method of creation and experimental evaluation. Requirements Eng. 9, 217-228 (2004)
4. Castano, S., De Antonellis, V.D., Fugini, M.G., Pernici, B.: Conceptual schema analysis: techniques and applications. ACM Trans. Database Syst. 23, 286-333 (1998)
5. Viscusi, G., Batini, C., Mecella, M.: Information Systems for eGovernment: a quality of service perspective. Springer, Berlin-Heidelberg (2010)
6. Tavana, M., Joglekar, P., Redmond, M.A.: An automated entity-relationship clustering algorithm for conceptual database design. Information Systems 32, 773-792 (2007)
7. Simperl, E., Sure, Y.: The Business View: Ontology Engineering Costs. In: Hepp, M., Leenheer, P., Moor, A., Sure, Y. (eds.) Ontology Management, vol. 7, pp. 207-225. Springer US (2008)
8. Batini, C., Lenzerini, M.: A Methodology for Data Schema Integration in the Entity Relationship Model. Software Engineering, IEEE Transactions on SE-10, 650-664 (1984)
9. Dahchour, M., Pirotte, A., Zimányi, E.: Generic Relationships in Information Modeling. In: Spaccapietra, S. (ed.) Journal on Data Semantics IV, vol. 3730, pp. 1-34. Springer Berlin / Heidelberg (2005)
10. Keet, C.M.: Enhancing comprehension of ontologies and conceptual models through abstractions. Springer-Verlag, Berlin Heidelberg (2007)
11.Hauch, R., Miller, A., Cardwell, R.: Information intelligence: metadata for information discovery, access, and integration. SIGMOD ’05: Proceedings of the 2005 ACM SIGMOD international conference of Management of data, New York, NY, USA, pp. 793–798. ACM
12. Batini, C., Grosso, R., Longobardi, G.: Design of Repositories of Conceptual Schemas in the small and in the large. Proceedings of the eGovernment Workshop '05 (eGOV05) (2005)
13. Batini, C., Ceri, S., Navathe, S.B.: Conceptual Database Design: An Entity-Relationship Approach. Benjamin Cummings/ Addison Wesley, Palo Alto, California, USA (1991)
14. Batini, C., Lenzerini, M., Navathe, S.B.: A comparative analysis of methodologies for database schema integration. ACM Comput. Surv. 18, 323-364 (1986)
15. Moody, D.L.: Theoretical and practical issues in evaluating the quality of conceptual models: current state and future directions. Data & Knowledge Eng. 55, 243-276 (2005)
16. Akoka, J., Comyn-Wattiau, I., Cherfi, S.S.S.: Quality of conceptual schemas an experimental comparison. In: Research Challenges in Information Science, 2008. RCIS 2008, pp. 197-208. (2008)

A conceptual framework and an underlying model for community detection and management in a Social Internetworking scenario

Pasquale De Meo[1], Antonino Nocera[2], Giovanni Quattrone[3], and Domenico Ursino[2]

[1] Dipartimento di Fisica, Sezione di Informatica, Università di Messina
Viale F. Stagno D'Alcontres, 31, 98166 Messina, Italy
[2] DIMET, Università Mediterranea di Reggio Calabria,
Via Graziella, Località Feo di Vito, 89122 Reggio Calabria, Italy
[3] Department of Computer Science, University College of London
Malet Place Engineering Building, London, WC1E 6BT, United Kingdom
pdemeo@unime.it,a.nocera@unirc.it,
g.quattrone@cs.ucl.ac.uk,ursino@unirc.it

Abstract In this paper we propose a conceptual framework, and an underlying model, to handle community detection, characterization and membership in a Social Internetworking scenario. In order to face these problems, our framework must preliminarily handle a further one, i.e. user similarity detection.

1 Introduction

In the last years social networks have become one of the key actors in the World Wide Web context. Currently, the most famous ones (e.g., Facebook and Twitter) have largely exceeded one hundred million members. Many of the current social networks are centered around a main topic, such as work, research, school, sport, etc. For this reason users interested in many topics have often to subscribe to multiple social networks.

This trend leads to a fragmentation of both the contents and the profile associated with each user inside different social networks. In order to face this issue, the concept of *Social Internetworking* has been recently proposed. This term indicates a scenario in which more, possibly heterogeneous, social networks cooperate to form a sort of federated system. Currently, there exist some commercial attempts to provide Social Internetworking functionalities; among them we cite Google Open Social, Power.com, Gathera and Friendfeed. All these systems are characterized by an underlying commercial philosophy; as a consequence, they provide only some *basic* functionalities that, however, work *very well* from an efficiency and effectiveness point of view.

As for the scientific standpoint, to the best of our knowledge, only some attempts to propose a Social Internetworking System have been presented in the literature [12,4,5]. Actually, Social Internetworking is a very challenging issue and the problems that can be investigated in this context are numerous and extremely fascinating (see [4] for some of them).

One of the most interesting problems in Social Internetworking is *community detection*. It aims at constructing very cohesive and loosely coupled communities of users possibly belonging to different social networks. In the literature, the detection of user communities inside a single social network has been highly investigated [6,7,9,11,13]. However, the corresponding solutions cannot be directly extended to the Social Internetworking context because the information to handle in this case is much more complex; for instance, the information that two users are both members of a social network whereas only one of them is a member of another social network is extremely important in a Social Internetworking scenario whereas it is not relevant if a single social network is considered.

Two other challenging problems strictly related to the previous one are *community characterization* and *community membership*. The former aims at detecting the main features that best represent a given community. The latter aims at determining the membership degree of a given user to one or more communities.

In order to face all these problems it is necessary to consider a preliminary one, i.e. *the detection of similarities among users* of a Social Internetworking System, possibly registered to different social networks. With regard to this problem, it is worth pointing out that many methods to compute similarities among users have been proposed. A first one compares their profiles; in this way the *semantic* similarity among them is determined. An alternative method considers the actions performed by them. Interestingly enough, these relationships could connect also users who do not know each other.

The four problems considered above are strictly related to each other and, therefore, it would be interesting to simultaneously face all of them. In our opinion one of the best ways to perform this task consists in the definition of a framework capable of handling all of them.

All the ideas expressed above represent the basis of this paper; in fact, in it we propose a conceptual framework, following the Component-Based Development paradigm and the layers architectural pattern, and an underlying model, for community detection, characterization and membership in a Social Internetworking System (hereafter, SIS).

This paper is organized as follows: Section 2 illustrates the proposed model for the representation of a SIS. Section 3 is devoted to present the proposed framework and to discuss its main characteristics. A comparison between the proposed framework and some related approaches can be found in Section 4. Finally, in Section 5, we draw our conclusions.

2 The proposed SIS model

A SIS consists of a set of social networks, a set of users operating on them and a set of resources stored therein.

A user can join more social networks of a SIS; in this case, analogously to what happens in other Social Internetworking scenarios (see, for instance, the reference scenario of Google Social Graph) we assume that it is possible (in case with the support of the user himself) to know all the accounts adopted by him in the social networks of the SIS. As a consequence, a unique identifier can be associated with a user in the whole SIS.

A set of tags can be associated with a user; they express his main activities and interests. Social networks allow users to post resources inside them. The identifier of a resource is unique only for a given user and does not depend on the social network where he posts it. Each resource is described by means of a set of tags. Inside the SIS each user can access a resource and can evaluate it by means of a numerical score. Resources of interest can be detected by submitting suitable queries; a query consists of a set of tags which specify the information needs of the user submitting it. Finally, if a mutual agreement exists, two users can declare their friendship within a given social network of the SIS.

From the previous description it is possible to observe that, in a SIS with the characteristics described above, a user u_i can perform four kinds of action, namely: *(i) membership*: it indicates that u_i has joined a social network s_k; *(ii) friendship*: it denotes that u_i has become friend of a user u_l in s_k; *(iii) post*: it indicates that u_i has posted a resource r_j in s_k; *(iv) evaluation*: it denotes that u_i has evaluated r_j in s_k; the corresponding evaluation is a number belonging to the real interval $[0, 1]$.

In order to define a SIS model with the features specified above we must preliminarily introduce the concepts of user profile and resource profile.

Specifically, the profile of a user u_i is a tuple $P_{u_i} = \langle ID_{u_i}, T_{u_i} \rangle$, where: *(i)* ID_{u_i} is the identifier of u_i; as previously pointed out, ID_{u_i} is unique within a SIS; *(ii)* $T_{u_i} = \{t_1, t_2, \ldots, t_p\}$ is a set of tags specifying the interests of u_i; these are the tags mostly used by him in the past for labeling or searching for the resources of his interests.

The profile of a resource r_j is a tuple $P_{r_j} = \langle ID_{r_j}, UserID_{r_j}, T_{r_j} \rangle$, where: *(i)* ID_{r_j} and $UserID_{r_j}$ are the identifiers of r_j and of the user who posted it, respectively; in our SIS model, a resource r_j is univocally identified for the user who posted it; as a consequence, it is univocally identified in the SIS by the pair $\langle ID_{r_j}, UserID_{r_j} \rangle$; *(ii)* $T_{r_j} = \{t_1, t_2, \ldots, t_m\}$ is the set of the tags labeling r_j and specifying its content.

We are now able to introduce our model for representing a SIS.

Definition 1. Let $\mathcal{U}$ be a set of users, let $\mathcal{R}$ be a set of resources and let $\mathcal{S}$ be a set of social networks. A Social Internetworking System SIS can be represented by a hypergraph $\mathcal{H} = \langle NSet, ESet \rangle$ where:

- $NSet = NSet_U \cup NSet_R \cup NSet_S$ is the set of the *nodes* of $\mathcal{H}$. There is a node $un_i \in NSet_U$ for each user $u_i \in \mathcal{U}$; a label $\langle U, P_{u_i} \rangle$, where P_{u_i} is the profile of u_i, is associated with this node. There is a node $rn_j \in NSet_R$ for each resource $r_j \in \mathcal{R}$; a label $\langle R, P_{r_j} \rangle$, where P_{r_j} is the profile of r_j, is associated with this node. There is a node $sn_k \in NSet_S$ for each social network $s_k \in \mathcal{S}$; a label $\langle S, ID_{s_k} \rangle$, where ID_{s_k} is the identifier of s_k, is associated with this node.

- $ESet = ESet_M \cup ESet_F \cup ESet_P \cup ESet_V$ is the set of the *hyperedges* of $\mathcal{H}$. Each hyperedge is associated with an action carried out by a user; specifically: *(i)* there is a hyperedge $e_M(un_i, sn_k) \in ESet_M$ if u_i has joined s_k; a label $\langle M \rangle$ is associated with e_M; *(ii)* there is a hyperedge $e_F(un_i, un_l, sn_k) \in ESet_F$ if u_i and u_l have specified that they are friends in s_k; a label $\langle F \rangle$ is associated with e_F; *(iii)* there is a hyperedge $e_P(un_i, rn_j, sn_k) \in ESet_P$ if u_i has posted r_j in s_k; a label $\langle P \rangle$ is associated with e_P; *(iv)* there is a hyperedge $e_V(un_i, rn_j, sn_k) \in ESet_V$ if u_i has evaluated r_j in s_k; a label $\langle V, v \rangle$ is associated with e_V; here v indicates the value that u_i assigned to r_j in s_k. □

It is worth pointing out that, in the scenario considered in Definition 1, users may be involved in four action types (i.e., membership to a social network, friendship, resource post and resource evaluation). However, besides these common action types, our approach can be easily extended to include other ones possibly performed inside social networks.

3 The proposed conceptual framework

The architecture of the proposed conceptual framework consists of four layers. In its turn, each layer consists of a set of components. The four layers of our framework are:

- *The User Similarity Modeling Layer*; it handles a mathematical model to compute the similarity of each pair of SIS users on the basis of the actions performed by them or of the content of their profiles.
- *The Clustering Layer*; it exploits the user similarities computed by the User Similarity Modeling Layer and groups users into virtual communities by means of suitable clustering algorithms.
- *The Community Characterization Layer*; it characterizes the virtual communities returned by the Clustering Layer on the basis of the information provided by the User Similarity Modeling Layer.
- *The Membership Layer*; for each SIS user, it determines the degree of his membership to each virtual community detected by the Clustering Layer.

The components currently defined for the *User Similarity Modeling Layer* are:

- The ARM (Action Relationship Modeling) component; it builds a matrix whose generic element represents the similarity degree of a pair of users; for this purpose it considers all the actions of a given type performed by them.
- The RA (Relationship Aggregation) component family; each RA receives a set of matrixes, one for each action type, and returns a matrix obtained by suitably aggregating them.
- The SRM (Semantic Relationship Modeling) component; it builds a matrix whose generic element represents the semantic similarity degree of a pair of users; for this purpose it exploits the corresponding user profiles.

The *Clustering Layer* currently contains only one component, called SC, which constructs virtual communities.

The components currently defined for the *Community Characterization Layer* are:

- The $CDBCC$ (Centrality Detection Based Community Characterization) component; for each community detected by the Clustering Layer on the basis of the actions performed by SIS users, it identifies the κ most representative users who characterize the community in the whole.
- The $SBCC$ (Semantic Based Community Characterization) component; for each community detected by the Clustering Layer on the basis of user profiles, it builds a profile which characterizes the community itself from the semantic point of view.

The components currently defined for the *Membership Layer* are:

- The $RUBM$ (Representative User Based Membership) component; given a SIS user, it determines the degree of his membership to each virtual community detected by the Clustering Layer on the basis of the actions performed by the SIS users; for this purpose it exploits the corresponding representative users derived by the $CDBCC$ component.
- The NBM (Neighborhood Based Membership) component; it solves the same problem solved by $RUBM$ but by means of a different strategy based on the analysis of the neighborhood of the considered user.
- The SBM (Semantic Based Membership) component; given a SIS user, it determines the degree of his membership to each virtual community detected by the Clustering Layer on the basis of user profiles; for this purpose it compares the profiles of both the considered user and the virtual communities.

It is worth pointing out that all the matrixes handled by the framework components are very sparse. As a consequence, suitable techniques for the management of very sparse matrixes are exploited to manage them.

Due to space limitations we cannot illustrate the components of the proposed framework in detail. However, in the following, we shall investigate its main features.

The main characteristic of our conceptual framework is that it operates on a SIS, instead of on a single social network. Even if the modeling of a SIS can be seen as an extension of the one of a social network (for instance, in both cases, it could be possible to use graphs or hypergraphs), the algorithms for solving the problems of interest are much more complex. For instance, they must consider that some social networks of a SIS could handle complementary information that, if suitably aggregated in the SIS, could provide some value added. On the other side, some social networks in a SIS could use different ways to represent the same information; in this last case, a preliminary homogeneity task appears necessary. Think, for instance, of the concepts of friendship, trustworthiness and reliability in different social networks. Think, also, that a user could have different profiles, different reputations, etc. in different social networks because these last ones can use different policies to handle and evaluate these properties and/or because a user could have different behaviors in different social networks. The social networks themselves could have different trusts and reputations, and this last information appears extremely useful to handle. All these heterogeneity issues must not be handled when operating on a single social network, whereas they become one of the main focuses when operating on a Social Internetworking scenario. Clearly, the algorithms underlying the components of the proposed framework handle all these problems and, therefore, are much more complex than the ones operating on a single social network.

As a further characteristic of our framework consider that, in the context of social networks, the simplest way to compute user similarities consists of computing the corresponding *Jaccard* coefficient. However, this coefficient is often not suited in Social Networking and Social Internetworking because it considers only the acquaintances of a user (and, therefore, local information), whereas it does not take global information into account. Our framework exploits information stored in the whole SIS to compute user similarities. In particular, it examines the whole set of paths connecting the nodes associated with two users, rather than only the direct links existing between them.

Furthermore, the approaches operating on single social networks are often based on the analysis of the structural properties of the involved networks. In order to handle the information heterogeneity problem possibly characterizing social networks in a SIS, our approach considers also the behaviors and the semantic information concerning involved users. Behaviors are derived from the examination of the actions performed by users; semantic information is extracted from the analysis of their profiles.

These are the main characteristics of our conceptual framework taken as a whole. The algorithms underlying each of its components have also other specificities and novelties that we cannot describe here due space limitations. We provide only some highlights about them in the next section.

4 Related work

In this section we compare our approach with other related ones already presented in the literature. Before carrying out this comparison, we point out that, to the best of our knowledge, no conceptual framework conceived to *jointly* handle user similarity detection as well as community detection, characterization and membership *in a Social Internetworking scenario* has been already proposed in the literature. As a matter of fact, all the previous approaches to handle user similarity detection or community detection, characterization and membership have been conceived to operate on a single social network.

In [1] the problem of computing user similarities in single social networks is formalized as an optimization problem. Other approaches compute similarities by exploiting *matrix based methods*. For instance, the approach of [10] uses a modified version of the Katz coefficient, *SimRank* [8] provides an iterative fixpoint method and the approach of [2] operates on directed graphs and uses an iterative approach relying on their spectral properties. Our approach, like the last ones mentioned above, uses information stored in the whole network (in our case in a SIS) to compute user similarities. However, there are some strong differences between the approaches described above and ours. First of all, the main goal of our approach is the construction and the management of virtual communities and, in this context, user similarity detection is just a step of this procedure. Moreover, generally, the approaches described above are based on the analysis of the structural properties of the involved networks. By contrast, our approach considers also the behaviors and the semantic information of involved users.

A large number of community detection algorithms is based on the concept of *network modularity.* This task is NP-hard and, then, suitably heuristics must be considered [6,3,7]. Many of the approaches described above return disjoint communities, i.e. they assign each user to only one community. Our approach follows a different perspective; in fact, it first constructs virtual communities and, then, for each pair $\langle u_i, VComm_x \rangle$, where u_i is a user and $VComm_x$ is a detected community, it computes a coefficient stating the membership degree of u_i to $VComm_x$. As for a second difference, many of the approaches described above detect communities by *directly* analyzing the graph associated with the social network which they operate on; in this analysis, they consider *the information associated with the edges* of this graph. By contrast, our approach detects communities by constructing support graphs starting from the hypergraph associated with the SIS which it operates on; these *support graphs* encode similarity relationships among users and are derived on the basis of the information associated with *both the nodes and the hyperedges* of the hypergraph representing the SIS. The third difference regards the features considered to characterize detected communities. In fact, many approaches described above do not focus on this problem; other ones perform this task by considering the behavior of the involved users. By contrast, our approach associates a great relevance with this issue and, in fact, implements different possible ways to perform this task.

5 Conclusions

In this paper we have proposed a conceptual framework to handle community detection, characterization and membership in a SIS. In our opinion the proposed framework should not be considered as an “end-point” but as a “starting-point”. As a matter of fact, it could be the precursor of a framework for the management of a SIS. This framework could consist of a central core (handling both the hypergraph representing the SIS and a catalogue of its users, resources and social networks) to which a large variety of plugins could be added; this way of proceeding is typical of many high-success software frameworks (think, for instance, of Eclipse and Thunderbird). In this way, it could be possible to realize a catalogue of plugins specialized to solve the various problems typical of Social Internetworking. As a further effort, it could be possible to design an intelligent wizard, based on an underlying recommender system, which suggests to a user the plugins that he should add in the framework to satisfy his needs, and guides him in their installation.

References

1. V. Batagelj, P. Doreian, and A. Ferligoj. An optimizational approach to regular equivalence. *Social Networks*, 14(1-2):121–135, 1992.
2. V.D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. van Dooren. A Measure of Similarity between Graph Vertices: Applications to Synonym Extraction and Web Searching. *SIAM Review*, 46(4):647–666, 2004.
3. A. Clauset, M. Newman, and C. Moore. Finding community structure in very large networks. *Physical Review Part E*, 70(6):066111, 2004.
4. P. De Meo, A. Nocera, D. Rosaci, and D. Ursino. Recommendation of reliable users, social networks and high-quality resources in a Social Internetworking System. *AI Communications*, 24(1):31–50, 2011.
5. P. De Meo, A. Nocera, G. Terracina, and D. Ursino. Recommendation of similar users, resources and social networks in a Social Internetworking Scenario. *Information Sciences*, 181(7):1285–1305, 2011.
6. M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Science of the United States of America*, 99(12):7821–7826, 2002.
7. R. Guimera and L. Nunes Amaral. Functional cartography of complex metabolic networks. *Nature*, 433(7028):895–900, 2005.
8. G. Jeh and J. Widom. SimRank: a measure of structural-context similarity. In *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'02)*, pages 538–543, Edmonton, Alberta, Canada, 2002. ACM Press.
9. M.S. Kim and J. Han. CHRONICLE: A Two-Stage Density-Based Clustering Algorithm for Dynamic Networks. In *Proc. of the International Conference on Discovery Science (DS'09)*, pages 152–167, Porto, Portugal, 2009. Lecture Notes in Computer Science. Springer.
10. E.A. Leicht, P. Holme, and M. E. J. Newman. Vertex similarity in networks. *Physical Review Part E*, 73(2):026120, 2006.
11. M. Newman and E.A. Leicht. Mixture models and exploratory analysis in networks. *Proceedings of the National Academy of Sciences of the United States of America*, 104:9564–9, 2007.
12. Y. Okada, K. Masui, and Y. Kadobayashi. Proposal of social internetworking. In *Web and Communication Technologies and Internet-Related Social Issues - HSI 2005*, volume 3597 of *Lecture Notes in Computer Science*, pages 114–124. Springer Berlin / Heidelberg, 2005.
13. G. Palla, I. Derenyi, I. Farkas, and T. Vicsek. Uncovering the overlapping community structure of complex networks in nature and society. *Nature*, 435(7043):814–818, 2005.

Groupware Mail Messages Analysis for Mining Collaborative Processes (poster paper)

Claudio Di Ciccio[1], Massimo Mecella[1]
Monica Scannapieco[2], and Diego Zardetto[2]

[1] Sapienza – Università di Roma
Dipartimento di Informatica e Sistemistica Antonio Ruberti
Via Ariosto 25, Roma, Italy
{cdc,mecella}@dis.uniroma1.it
[2] Istituto Nazionale di Statistica
Via Balbo 16, Roma, Italy
{scannapi,zardetto}@istat.it

Nowadays, the most of the research related to workflows has considered the management of formal business processes. There has been some discussion of informal processes, often under names such as "artful business processes", e.g., [1]: informal processes are typically carried out by those people whose work is mental rather than physical (managers, professors, researchers, etc.), the so called "knowledge workers" [2]. With their skills, experience and knowledge, they are used to perform difficult tasks, which require complex, rapid decisions among multiple possible strategies, in order to fulfill specific goals. In contrast to business processes that are formal and standardized, often informal processes are not even written down, let alone defined formally, and can vary from person to person even when those involved are pursuing the same objective. Knowledge workers create informal processes "on the fly" to cope with many of the situations that arise in their daily work. While informal processes are frequently repeated, since they are not written down, they are not exactly reproducible, even by their originators, nor can they be easily shared. Their outcome releases and their information exchanges are very often done by means of e-mail conversations, which are a fast, reliable, permanent way of keeping track of the activities that they fulfill.

The objective of the research proposed in this position document is to automatically build, on top of a collection of e-mails, a set of workflow models that represent the artful processes which lay behind the knowledge workers activities.

A company can take many advantages out of this work. First of all, the unspecified agile processes that are autonomously used become formalized: since such models are not defined *a priori* by experts but rather inferred from real-life scenarios that actually took place, they are guaranteed to respect the true executions and not reflect the expected ones (often Business Process Management tools are used to show the discrepancy between the theoretical workflows and the concrete ones). Secondly, such models can be shared and compared, so that the best practices might be put in evidence from the community of knowledge workers, to the whole business benefit. Moreover, without any further computational cost, an analysis over such processes can be done, so that bottlenecks and delays in actual executions can be found out. We remark here that all of these utilities come out with almost no effort for workers, due to the automated nature of the envisioned approach.

The approach we would like to pursue, in order to retrieve a collection of process models out of an initial set of e-mail messages, involves many research fields at a time, each concerning a sequential phase of the overall processing. For first, we exploit *text categorization* techniques to filter the set of e-mails of interest out of the whole provided collection. Then, we make use of *object matching* algorithms to obtain clusters of related e-mail conversations, from the previous extracted subset. Every cluster is subsequently treated by *text mining information extraction* procedures, in order to find out which tasks e-mail messages are about. Finally, *process mining* is used to abstract process models representing the workflows, which the sets of subsumed tasks were considered traces of. We aim in the future to release a prototype realization of our approach, named MailOfMine.

Background and state of the art. *Process Mining*, also referred to as *Workflow Mining* (see [3]), is the set of techniques that allow the extraction of structured process descriptions, stemming from a set of recorded real executions. Such executions are intended to be stored in so called *event log*s, i.e., textual representations of a temporarily ordered linear sequence of tasks. There, each recorded *event* reports the execution of a *task* (i.e., a well-defined step in the workflow) in a *case* (i.e., a workflow instance). Beware that events are always recorded sequentially, even though tasks could be executed in parallel: it is up to the algorithm to infer the actual structure of the workflow that they are traces of, identifying the causal dependencies between tasks (*condition*s).

The idea to apply process mining in the context of workflow management systems was introduced in [4]. There, processes were modeled as directed graphs where vertices represented individual activities and edges stood for dependencies between them. Cook and Wolf, at the same time, investigated similar issues in the context of software engineering processes. In [5] they described three methods for process discovery: one using neural networks, another with a purely algorithmic technique, the last adopting a Markovian approach. The authors consider the latter two the most promising approaches. The purely algorithmic approach builds a finite state machine where states are fused if their futures (in terms of possible behavior in the next k steps) are identical. The Markovian approach uses a mixture of algorithmic and statistical methods and is able to deal with noise. Note that the results presented in [5] are limited to sequential behavior.

Most of the nowadays mainstream process mining tools model processes as Workflow Nets (WFNs – see [6]), explicitly designed to represent the control-flow dimension of a workflow. From [4] onwards, many techniques have been proposed, in order to address specific issues: pure algorithmic (e.g., α algorithm, drawn in [7] and its evolution [8], α^{++}), heuristic (e.g., [9]), genetic (e.g., [10]). Indeed, heuristic and genetic algorithms have been introduced to cope with noise, that the pure algorithmic techniques were not able to manage.

A very smart extension to the previous research work has been recently achieved by the two-steps algorithm proposed in [11]. Differently from previous works, which typically provide a single process mining step, it splits the computation into two phases: the first builds a Transition System (TS) that represents the process behavior and the tasks causal dependencies; the second makes use of the state-based "theory of regions" ([12], [13], [14]) to construct a WFN which is bisimilar to the TS. The first phase is made "tunable", so that it can be either more strictly adhering or more permissive to the analyzed log traces behavior, i.e., the expert can decide a balance between "overfitting" and "underfitting". Recall, indeed, that event logs are not the whole universe of possible traces that may run: hence, on one hand, the extracted process model should be valid for future unpredictable cases; on the other hand, it should be checked whether such a process model actually adheres to the behavior that most of the gathered traces reflected in the past (we say "most" here to emphasize that a little percentage of the whole log may represent erroneous deviations from the natural flow of tasks). The second phase has no parameter to set, since its only aim is synthesizing the TS into an equivalent WFN. Thus, it is fixed, while the former step can be realized exploiting one among many of the previously proposed "one-step" algorithms (for instance, [9] seems to integrate well).

[2] describes the "ACTIVE" EU collaborative project, coordinated by British Telecom, currently ongoing (due date is February 2011). Such project addresses the need for greater knowledge worker productivity by providing more effective and efficient tools. Among the main objectives, it aims at helping users to share and reuse informal processes, even by learning those processes from the user's behavior.

Object Matching (OM) is the problem of identifying pairs of data objects coming from different sources and representing the same real world object. The aim of OM techniques is typically related to the cleaning of large data sets from erroneous duplicates. Such duplicates usually derive from misspellings, abbreviations, lack of standard formats, or any combination of these factors. Analogous techniques can be used to achieve a *reference reconciliation* (see [15]), namely the identification of related references in complex information spaces where data corresponding to the same reference can be structurally heterogeneous (e.g., e-mail contacts, documents, spreadsheets). If data objects are records, the problem is known in literature as Record Linkage (RL) [16]. Integration of different data sources and improvement of the quality of single sources are only some of the real application scenarios that need to solve the OM problem. The RL version of the OM problem has received considerable attention by the scientific community. Most of the works (e.g. [17], [18]) focus on solving the problem within a relational DBMS: often, in the real world, entities have two or more representations in databases. Duplicate records do not share a common key and/or they contain errors that make duplicate matching a difficult task.

Some techniques for discovering strong entity associations in semi-structured data, such as document meta-data, are also known (e.g., see [19]). Here, the attention is moved towards a novel technique, proposed in [20]. Such a methodology differs from the others in the field, indeed, since *(i)* it is able to treat not only records but every kind of objects, provided that it is possible to define a distance for them, *(ii)* it is focused on effectiveness rather than on the ability to manage huge amount of data, *(iii)* its nature is completely automated. Regarding the third point, there were new unsupervised techniques (such as [17]) already proposed, but none of them, to the best of our knowledge, were fully automated. Indeed, though not requiring exactly a clerically prepared training set, such techniques still depend critically on some external inputs (e.g., human intervention is needed to set crucial parameters for the algorithms in [17]). It achieves such a result by applying a two-phases algorithm. It makes use of statistical models that allow to represent a probability distribution as a convex combination of two distinct probability distributions: the one stemming from the sub-population of Matches ($\mathcal{M}$) and the other from that of Unmatches ($\mathcal{U}$). Thus, the two consecutive tasks are: *(i)* estimating mixture parameters by fitting the model to the observed distance measures between pairs; *(ii)* then, obtaining a probabilistic clustering of the pairs into Matches and Unmatches, by exploiting the fitted model. In the clustering step the fitted mixture model is used to search an optimal classification rule such that each pair can be assigned, based on its observed distance value, either to the $\mathcal{M}$ or to the $\mathcal{U}$ class. Such this constrained optimization problem is solved by means of a purposefully designed evolutionary algorithm [21].

Text Mining, or Knowledge Discovery from Text (KDT) deals with the machine supported analysis of text: indeed, it refers generally to the process of extracting interesting information and knowledge from unstructured text. It is a field in the intersection of related areas such as information retrieval, machine learning, statistics, computational linguistics and, especially, data mining.

Natural language text contains much information that is not directly suitable for automatic analysis by a computer. However, computers can be used to sift through large amounts of text and extract useful information from single words, phrases or passages. Therefore, text mining can be interpreted in the sense of an information extraction activity, i.e., as a restricted form of full natural language understanding, where we know in advance what kind of semantic information we are looking for.

Text mining covers many other topics that are out of the scope of this paper: for a comprehensive survey on it, please refer to [22] or, for a more extended explanation, [23]. A particular discipline of interest that belongs to it is *Text Categorization* (TC, also known as *Text Classification* or *Topic Spotting*), namely, the activity of assigning *categories* (symbolic labels), from a given set, to natural language texts, on the basis of *endogenous* knowledge only (i.e., knowledge is extracted from the documents only and not from other possible external sources). The categories in the given set can be two (*Binary* TC, i.e., a document can belong to a category or its complement) or more. TC is applied in many contexts, such as document filtering and automated metadata generation. For a comprehensive survey on Machine Learning in Automated Text Categorization, the reader can refer to [24].

As a successful example of application, we want to report here a case that deeply relates with the research proposal of this paper: [25] proposes a method employing text mining techniques to analyze e-mails collected at a customer center. The method uses two kinds of domain-dependent knowledge: one is a key concept dictionary manually provided by human experts and the other is a concept relation dictionary automatically acquired by a fuzzy inductive learning algorithm. Based upon the work exposed in [26], the depicted method takes as input the subject and the body of an e-mail, decides a text class for the e-mail, extracts key concepts from e-mails and finally presents their statistical information as well.

The MailOfMine proposed approach. As depicted in Figure 1, MailOfMine is intended to be divided into layers that act in series like in a waterfall model. Each layer is built to achieve a specific task (e-mail reading, e-mail filtering, e-mail clustering, tasks extraction, workflow mining). The initial input is a file storing a whole repository of e-mail messages. The final output is a set of process models inferred from the given input. In the middle, every layer is drawn to receive as input the result that comes from the upper one, starting from the raw e-mails, and in turn take the output to the lower one, down to the final process models.

The very first task is accomplished by a plug-in based system, that must be able to extract a common format for e-mail messages, stemmed from heterogeneous sources: for example, Microsoft Outlook, Mozilla Thunderbird, Evolution Mail files use different storage formats, but the system must be able to manage them all.

In Figure 1, such task corresponds to the layer 0 (*Multiple E-Mail Storage Formats Extraction*).

From that point on, the proposed approach follows this pattern, founded on a double analysis/synthesis passage:

1. (*Mail Filtering*) *analysis* on the set of heterogeneous e-mail messages to filter irrelevant messages out (Text Categorization, Information Retrieval);
2. (*Mail Clustering*) *synthesis* on the set of relevant messages, to cluster related ones into homogeneous[3] sets (Object Matching);
3. (*Tasks Inference*) *analysis* on each cluster, to extract the tasks out (Information Extraction);
4. (*Workflows Inference*) *synthesis* on tasks, to build a process model that conforms with the subsumed trace (Process Mining).

Conclusions. In this paper, the MailOfMine approach and its basic idea of inferring artful business process models, i.e., agile workflows formal representations, from knowledge workers e-mail storage files, have been proposed. The

[3] Here “homogeneous” has to be intended with respect to the activities to serve for the purpose of task inference.

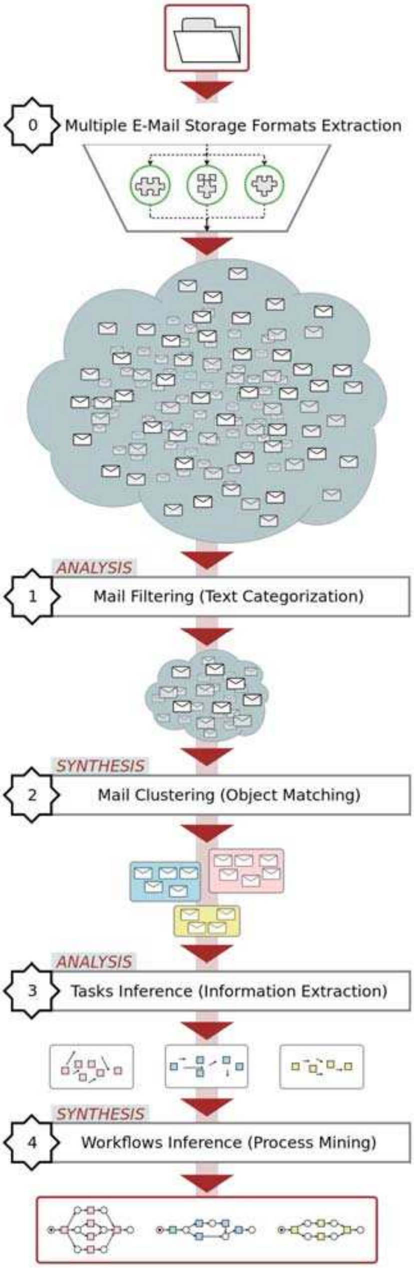

0
Multiple E-Mail Storage Formats Extraction
ANALYSIS
1
Mail Filtering (Text Categorization)
SYNTHESIS
2
Mail Clustering (Object Matching)
ANALYSIS
3
Tasks Inference (Information Extraction)
SYNTHESIS
4
Workflows Inference (Process Mining)

Fig. 1. The MailOfMine architecture

approach can take advantage of many previous works that succeeded in the heterogeneous research fields that are involved in this context (Text Mining, Object Matching, Process Mining). However, new research is indeed needed: the challenge is not only in the integration of various techniques, but also in applying process mining techniques on top of more traditional text mining ones. Currently we are studying the details of all the different layers/steps, and then we will proceed to an effective validation, by applying the approach and the prototype tool to a corpus of about 10 GByte emails, collected over 10 year of work of some authors to Italian and European research projects.

In future work, we want also to address other interesting issues. A challenge is the one of cooperative activities: they may involve many knowledge workers at a time, and it can happen that a task, say *DoIt*, that Mr A. Bloggs demands to Mrs B. Doe, is in turn redirected to Mr C. Smith. Thus, Mr C. Smith fulfills a series of tasks, in collaboration with Mrs B. Doe, which Mr A. Bloggs is unaware of. It could be interesting, then, to investigate on how to relate these activities that are traced by separated e-mail sets (one belonging to Mr A. Bloggs, the other to Mrs B. Doe), so to unify them under a single workflow model.

Another point to cope with is the question of privacy: e-mail messages may contain sensible private information that the single knowledge worker, or the company, might not want to be processed. At this initial stage of our research, we are supposing that treated data are public at least to the company that the knowledge worker works for, since we consider just the company mailbox and not her personal one, and the inferred information would not be presented to other people than who the company wants to involve. But how to include privacy concerns is surely a challenge to be addressed in future work.

References

1. C. Hill, R. Yates, C. Jones, and S. L. Kogan, "Beyond predictable workflows: Enhancing productivity in artful business processes," *IBM Systems Journal*, vol. 45, no. 4, pp. 663–682, 2006.
2. P. Warren, N. Kings, I. Thurlow, J. Davies, T. Buerger, E. Simperl, C. Ruiz, J. M. Gomez-Perez, V. Ermolayev, R. Ghani, M. Tilly, T. Bösser, and A. Imtiaz, "Improving knowledge worker productivity - the active integrated approach," *BT Technology Journal*, vol. 26, no. 2, pp. 165–176, 2009.
3. W. M. P. van der Aalst, "The application of petri nets to workflow management," *Journal of Circuits, Systems, and Computers*, vol. 8, no. 1, pp. 21–66, 1998.
4. R. Agrawal, D. Gunopulos, and F. Leymann, "Mining process models from workflow logs," in *Advances in Database Technology – EDBT'98.*
5. J. E. Cook and A. L. Wolf, "Discovering models of software processes from event-based data," *ACM Trans. Softw. Eng. Methodol.*, vol. 7, no. 3, pp. 215–249, 1998.
6. W. M. P. van der Aalst, "Verification of workflow nets," in *ICATPN*, ser. Lecture Notes in Computer Science, P. Azéma and G. Balbo, Eds., vol. 1248. Springer, 1997, pp. 407–426.
7. W. M. P. van der Aalst, T. Weijters, and L. Maruster, "Workflow mining: Discovering process models from event logs," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 9, pp. 1128–1142, 2004.

8. L. Wen, W. M. P. van der Aalst, J. Wang, and J. Sun, “Mining process models with non-free-choice constructs,” *Data Min. Knowl. Discov.*, vol. 15, no. 2, pp. 145–180, 2007.
9. A. Weijters and W. van der Aalst, “Rediscovering workflow models from event-based data using little thumb,” *Integrated Computer-Aided Engineering*, vol. 10, p. 2003, 2001.
10. A. K. Medeiros, A. J. Weijters, and W. M. Aalst, “Genetic process mining: an experimental evaluation,” *Data Min. Knowl. Discov.*, vol. 14, no. 2, pp. 245–304, 2007.
11. W. van der Aalst, V. Rubin, H. Verbeek, B. van Dongen, E. Kindler, and C. Günther, “Process mining: a two-step approach to balance between underfitting and overfitting,” *Software and Systems Modeling*, vol. 9, pp. 87–111, 2010.
12. J. Cortadella, M. Kishinevsky, L. Lavagno, and A. Yakovlev, “Synthesizing petri nets from state-based models,” in *Computer-Aided Design, 1995. ICCAD-95. Digest of Technical Papers, 1995 IEEE/ACM International Conference on*, nov. 1995, pp. 164 –171.
13. ——, “Deriving petri nets from finite transition systems,” *Computers, IEEE Transactions on*, vol. 47, no. 8, pp. 859 –882, aug. 1998.
14. J. Desel and W. Reisig, “The synthesis problem of petri nets,” *Acta Informatica*, vol. 33, pp. 297–315, 1996.
15. X. Dong, A. Y. Halevy, and J. Madhavan, “Reference reconciliation in complex information spaces,” in *SIGMOD Conference*, ACM, 2005, pp. 85–96.
16. A. K. Elmagarmid, P. G. Ipeirotis, and V. S. Verykios, “Duplicate record detection: A survey,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, pp. 1–16, 2007.
17. S. Chaudhuri, V. Ganti, and R. Motwani, “Robust identification of fuzzy duplicates,” in *Data Engineering, 2005. ICDE 2005. Proceedings. 21st International Conference on*, apr. 2005, pp. 865 – 876.
18. S. Guha, N. Koudas, A. Marathe, and D. Srivastava, “Merging the results of approximate match operations,” in *VLDB ’04: Proceedings of the Thirtieth international conference on Very large data bases*, pp. 636–647.
19. N. Sarkas, A. Angel, N. Koudas, and D. Srivastava, “Efficient identification of coupled entities in document collections,” in *ICDE 2010*, pp. 769–772.
20. D. Zardetto, M. Scannapieco, and T. Catarci, “Effective automated object matching,” in *ICDE 2010*, pp. 757–768.
21. Z. Michalewicz, *Genetic algorithms + data structures = evolution programs (2nd, extended ed.)*. New York, NY, USA: Springer-Verlag New York, Inc., 1994.
22. A. Hotho, A. Nürnberger, and G. Paaß, “A brief survey of text mining,” *LDV Forum - GLDV Journal for Computational Linguistics and Language Technology*, vol. 20, no. 1, pp. 19–62, May 2005.
23. M. W. Berry and M. Castellanos, *Survey of Text Mining II: Clustering, Classification, and Retrieval*, M. W. Berry and M. Castellanos, Eds. Springer, September 2007.
24. F. Sebastiani, “Machine learning in automated text categorization,” *ACM Comput. Surv.*, vol. 34, no. 1, pp. 1–47, 2002.
25. S. Sakurai and A. Suyama, “An e-mail analysis method based on text mining techniques,” *Appl. Soft Comput.*, vol. 6, no. 1, pp. 62–71, 2005.
26. S. Sakurai, Y. Ichimura, A. Suyama, and R. Orihara, “Acquisition of a knowledge dictionary for a text mining system using an inductive learning method,” in *Proceedings of IJCAI 2001 Workshop on Text Learning: Beyond Supervision*, 2001, pp. 45–52.

Computing Privacy Preserving OLAP Aggregations on Data Cubes: A Constraint-Based Approach

Alfredo Cuzzocrea[1], Domenico Saccà[2], Vincenzo Rodinò[2]

[1] ICAR-CNR and University of Calabria
87036 Cosenza, Italy
cuzzocrea@si.deis.unical.it
[2] DEIS Dept, University of Calabria
87036 Cosenza, Italy
{sacca, vrodino}@deis.unical.it

Abstract. A *constraint-based framework for computing privacy preserving OLAP aggregations on data cubes* is proposed and experimentally assessed in this paper. Our framework introduces a *novel privacy OLAP notion*, which, following consolidated paradigms of OLAP research, *looks at the privacy of aggregate patterns defined on multidimensional ranges rather than the privacy of individual tuples/data-cells*, like similar efforts in privacy preserving database and data-cube research. We complete our main theoretical contribution by means of an experimental evaluation and analysis of the effectiveness of our proposed framework on synthetic, benchmark and real-life data cubes.

1 Introduction

Given a multidimensional range R of an OLAP data cube A [5], an aggregate pattern over R is defined as an aggregate value extracted from R that is able of providing a "description" of data stored in R. In order to capture the privacy of aggregate patterns, in this paper we introduce a novel notion of privacy OLAP. According to this novel notion, given a data cube A the privacy preservation of A is modeled in terms of the privacy preservation of aggregate patterns defined on multidimensional data stored in A. Therefore, we say that a data cube A is privacy preserving iff aggregate patterns extracted from A are privacy preserving. Contrary to our innovative privacy OLAP notion above, previous privacy preserving OLAP proposals (e.g., [1,6,9]) totally neglect this even-relevant theoretical aspect, and, inspired by well-established techniques that focus on the privacy preservation of relational tuples [10,7], mostly focus on the privacy preservation of data cells (e.g., [9]) accordingly. Despite this, OLAP deals with aggregate data, and neglects individual information. Therefore, it makes more sense to deal with the privacy preservation of aggregate patterns rather than the privacy preservation of data cube cells.

In this paper we propose an innovative framework based on *flexible sampling-based data cube compression techniques for computing privacy preserving OLAP aggregations on data cubes while allowing approximate answers to be efficiently evaluated over such aggregations*. This framework addresses an application scenario where, given a multidimensional data cube *A* stored in a *producer* Data Warehouse server, a collection of multidimensional portions of *A* defined by a given (range) *query-workload QWL* of interest must be published online for *consumer* OLAP client applications. Moreover, after published, the collection of multidimensional portions is no longer connected to the Data Warehouse server, and updates are handled from the scratch at each new online data delivery. The query-workload *QWL* is cooperatively determined by the Data Warehouse server and OLAP client applications, mostly depending on OLAP analysis goals of client applications, and other parameters such as business processes and requirements, frequency of accesses, and locality. OLAP client applications wish for retrieving summarized knowledge from *A* via adopting a *complex multi-resolution query model* whose components are (*i*) queries of *QWL* and, for each query *Q* of *QWL*, (*ii*) *sub-queries* of *Q* (i.e., in a multi-resolution fashion). To this end, for each query *Q* of *QWL*, an *accuracy grid* $\mathcal{G}(Q)$, whose cells model sub-queries of interest, is defined. While aggregations of (authorized) queries and (authorized) sub-queries in *QWL* are disclosed to OLAP client applications, it must be avoided that, by meaningfully combining aggregate patterns extracted from multidimensional ranges associated to queries and sub-queries in *QWL*, malicious users could infer sensitive knowledge about other multidimensional portions of *A* that, due to privacy reasons, are hidden to unauthorized users. Furthermore, in our reference application scenario, target data cubes are also massive in size, so that data compression techniques are needed in order to efficiently evaluate queries, yet introducing *approximate answers* having a certain *degree of approximation* that, however, is perfectly tolerable for OLAP analysis goals [2]. In our proposal, the described application scenario with accuracy and privacy features is accomplished by means of the so-called *accuracy/privacy contract*, which determines the *accuracy/privacy constraint* under which client applications must access and process multidimensional data. In this contract, the Data Warehouse server and client OLAP applications play the role of mutual subscribers, respectively.

2 Constraint-Based Formalization of the Privacy Preserving OLAP Problem

2.1. Fundamentals. A *data cube A* defined over a relational data source *S* is a tuple $A = \langle D,\mathcal{F},\mathcal{H},\mathcal{M}\rangle$, such that: (*i*) *D* is the data domain of *A* containing (OLAP) data cells, which are the basic aggregations of *A* computed over relational tuples stored in *S*; (*ii*) $\mathcal{F}$ is the set of *dimensions* of *A*; (*iii*) $\mathcal{H}$ is the set of *hierarchies* related to the dimensions of *A*; (*iv*) $\mathcal{M}$ is the set of *measures* of *A*.

Given an $|\mathcal{F}|$-dimensional data cube *A*, an *m-dimensional range-query Q* against *A*, with $m \le |\mathcal{F}|$, is a tuple $Q = \langle R_{k_0}, R_{k_1}, \ldots, R_{k_{m-1}}, \mathcal{A}\rangle$, such that: (*i*) R_{k_i} denotes a *contiguous* range defined on the dimension d_{k_i} of *A*, with k_i belonging to the range $[0, |\mathcal{F}|-1]$, and (*ii*) $\mathcal{A}$ is a SQL aggregation operator.

Given a query *Q* against a data cube *A*, the *query region* of *Q*, denoted by *R*(*Q*), is defined as the sub-domain of *A* bounded by the ranges $R_{k_0}, R_{k_1}, \ldots, R_{k_{m-1}}$ of *Q*.

Given an *m*-dimensional query *Q*, the accuracy grid $\mathcal{G}(Q)$ of *Q* is a tuple $\mathcal{G}(Q) = \langle \Delta\ell_{k_0}, \Delta\ell_{k_1}, \ldots, \Delta\ell_{k_{m-1}}\rangle$, such that $\Delta\ell_{k_i}$ denotes the range partitioning *Q* along the dimension d_{k_i} of *A*, with k_i belonging to $[0, |\mathcal{F}|-1]$, in a $\Delta\ell_{k_i}$-based (one-dimensional) partition. By combining the one-dimensional partitions along *all* the dimensions of *Q*, we finally obtain $\mathcal{G}(Q)$ as a *regular multidimensional partition* of *R*(*Q*). From Section 1, recall that the elementary cell of the accuracy grid $\mathcal{G}(Q)$ is implicitly defined by sub-queries of *Q* belonging to the query-workload *QWL* against the target data cube.

Based on the latter definitions, in our framework we consider the broader concept of *extended range-query* Q^+, defined as a tuple $Q^+ = \langle Q, \mathcal{G}(Q)\rangle$, such that (*i*) *Q* is a "classical" range-query, $Q = \langle R_{k_0}, R_{k_1}, \ldots, R_{k_{m-1}}, \mathcal{A}\rangle$, and (*ii*) $\mathcal{G}(Q)$ is the accuracy grid associated to *Q*, $\mathcal{G}(Q) = \langle \Delta\ell_{k_0}, \Delta\ell_{k_1}, \ldots, \Delta\ell_{k_{m-1}}\rangle$, with the condition that each interval $\Delta\ell_{k_i}$ is defined on the *corresponding* range R_{k_i} of the dimension d_{k_i} of *Q*. For the sake of simplicity, we assume $Q \equiv Q^+$.

Given an *n*-dimensional data domain *D*, we introduce the *volume* of *D*, denoted by $\|D\|$, as follows: $\|D\| = |d_0| \times |d_1| \times \ldots \times |d_{n-1}|$, such that $|d_i|$ is the cardinality of the dimension d_i of *D*. This definition can also be extended to a multidimensional data cube *A*, thus introducing the volume of *A*, $\|A\|$, and to a multidimensional range-query *Q*, thus introducing the volume of *Q*, $\|Q\|$.

Given a data cube *A*, a range query-workload *QWL* against *A* is defined as a *collection* of (range) queries against *A*, as follows: $QWL = \{Q_0, Q_1, \ldots, Q_{|QWL|-1}\}$, with $R(Q_k) \subseteq R(A)\ \forall\ Q_k \in QWL$.

Given a query-workload $QWL = \{Q_0, Q_1, \ldots, Q_{|QWL|-1}\}$, we say that QWL is *non-overlapping* if there not exist two queries Q_i and Q_j belonging to QWL such that $R(Q_i) \cap R(Q_j) \neq \emptyset$. Given a query-workload $QWL = \{Q_0, Q_1, \ldots, Q_{|QWL|-1}\}$, we say that QWL is *overlapping* if there exist two queries Q_i and Q_j belonging to QWL such that $R(Q_i) \cap R(Q_j) \neq \emptyset$. Given a query-workload $QWL = \{Q_0, Q_1, \ldots, Q_{|QWL|-1}\}$, the *region set* of QWL, denoted by $R(QWL)$, is defined as the *collection* of regions of queries belonging to QWL, as follows: $R(QWL) = \{R(Q_0), R(Q_1), \ldots, R(Q_{|QWL|-1})\}$.

2.2. Accuracy Metrics. As accuracy metrics for answers to queries of the target query-workload QWL, we make use of the *relative query error* between exact and approximate answers, which is a well-recognized-in-literature measure of quality for approximate query answering techniques in OLAP (e.g., see [2]).

Formally, given a query Q_k of QWL, we denote as $A(Q_k)$ the exact answer to Q_k (i.e., the answer to Q_k evaluated over the original data cube A), and as $\tilde{A}(Q_k)$ the approximate answer to Q_k (i.e., the answer to Q_k evaluated over the synopsis data cube A'). Therefore, the relative query error $E_Q(Q_k)$ between $A(Q_k)$ and $\tilde{A}(Q_k)$ is defined as follows: $E_Q(Q_k) = \frac{|A(Q_k) - \tilde{A}(Q_k)|}{\max\{A(Q_k), 1\}}$.

$E_Q(Q_k)$ can be extended to the whole query-workload QWL, thus introducing the *average relative query error* $\bar{E}_Q(QWL)$ that takes into account the contributions of relative query errors of all the queries Q_k in QWL, each of them weighted by the volume of the query, $\|Q_k\|$, with respect to the whole volume of queries in QWL, i.e. the *volume of QWL*, $\|QWL\|$. $\|QWL\|$ is defined as follows: $\|QWL\| = \sum_{k=0}^{|QWL'|-1} \|Q_k\|, Q_k \in QWL$.

Based on the previous definition of $\|QWL\|$, the average relative query error $\bar{E}_Q(QWL)$ for a given query-workload QWL can be expressed as a *weighted linear combination* of relative query errors $E_Q(Q_k)$ of all the queries Q_k in QWL, as follows: $\bar{E}_Q(QWL) = \sum_{k=0}^{|QWL|-1} \frac{\|Q_k\|}{\|QWL\|} \cdot E_Q(Q_k)$, i.e.: $\bar{E}_Q(QWL) = \sum_{k=0}^{|QWL|-1} \frac{\|Q_k\|}{\sum_{j=0}^{|QWL|-1} \|Q_j\|} \cdot \frac{|A(Q_k) - \tilde{A}(Q_k)|}{\max\{A(Q_k), 1\}}$.

2.3 Privacy Metrics. Since we deal with the problem of ensuring the privacy preservation of OLAP aggregations, our privacy metrics takes into consideration how sensitive knowledge can be discovered from aggregate data, and tries to limit this possibility. On a theoretical plane, this is modeled by the privacy OLAP notion introduced in Section 1.

To this end, we first study how sensitive aggregations can be discovered from the target data cube A. Starting from the knowledge about A (e.g., range sizes, OLAP hierarchies etc), and the knowledge about a given query Q_k belonging to the query-workload QWL (i.e., the volume of Q_k, $\|Q_k\|$, and the exact answer to Q_k, $A(Q_k)$), it is possible to infer knowledge about sensitive ranges of data contained within $R(Q_k)$. For instance, it is possible to derive the average value of the contribution throughout which each basic data cell of A within $R(Q_k)$ contributes to $A(Q_k)$, which we name as *singleton aggregation* $I(Q_k)$. $I(Q_k)$ is defined as follows: $I(Q_k) = \frac{A(Q_k)}{\|Q_k\|}$.

It is easy to understand that, starting from the knowledge about $I(Q_k)$, it is possible to *progressively* discover aggregations of larger range of data within $R(Q_k)$, rather than the one stored within the basic data cell, thus inferring even-more-useful sensitive knowledge. Also, by exploiting OLAP hierarchies and the well-known roll-up operator, it is possible to discover aggregations of ranges of data at higher degrees of such hierarchies.

Secondly, we study how OLAP client applications can discover sensitive aggregations from the knowledge about approximate answers, and, similarly to the previous case, from the knowledge about data cube and query metadata. Starting from the knowledge about the synopsis data cube *A'*, and the knowledge about the answer to a given query Q_k belonging to the query-workload *QWL*, it is possible to derive an *estimation* on $I(Q_k)$, denoted by $\tilde{I}(Q_k)$, as follows: $\tilde{I}(Q_k) = \frac{\tilde{A}(Q_k)}{S(Q_k)}$, such that $S(Q_k)$ is the *number of samples* effectively extracted from $R(Q_k)$ to compute *A'* (note that $S(Q_k) < \|Q_k\|$). The relative difference between $I(Q_k)$ and $\tilde{I}(Q_k)$, named as *relative inference error* and denoted by $E_I(Q_k)$, gives us a metrics for the privacy of $\tilde{A}(Q_k)$, which is defined as follows: $E_I(Q_k) = \frac{|I(Q_k) - \tilde{I}(Q_k)|}{\max\{I(Q_k),1\}}$.

Similarly to what done for the average relative query error $\overline{E}_Q(QWL)$, we model $\overline{E}_I(QWL)$ as follows: $\overline{E}_I(QWL) = \sum_{k=0}^{|QWL|-1} \frac{\|Q_k\|}{\|QWL\|} \cdot E_I(Q_k)$, i.e.: $\overline{E}_I(QWL) = \sum_{k=0}^{|QWL|-1} \frac{\|Q_k\|}{\sum_{j=0}^{|QWL|-1} \|Q_j\|} \cdot \frac{|I(Q_k) - \tilde{I}_U(Q_k)|}{\max\{I(Q_k),1\}}$

2.4 Thresholds. Similarly to related proposals appeared in literature recently [9], in our framework we introduce the accuracy threshold Φ_Q and the privacy threshold Φ_I. Φ_Q and Φ_I give us an *upper bound* for the average relative query error $\overline{E}_Q(QWL)$ and a *lower bound* for the average relative inference error $\overline{E}_I(QWL)$ of a given query-workload *QWL* against the synopsis data cube *A'*, respectively. As stated in Section 1, Φ_Q and Φ_I allow us to meaningfully model and treat the accuracy/privacy constraint by means of rigorous mathematical/statistical models. In our application scenario, Φ_Q and Φ_I are cooperatively negotiated by the Data Warehouse server and OLAP client applications, based on specific applicative requirements and constraints, *by following a best-effort approach*.

3 Computing Privacy Preserving OLAP Data Cubes via Constrained-based Sampling

3.1. Accuracy-Grid-Constrained Sampling of OLAP Data Cubes. Computing the synopsis data cube *A'* via sampling the input data cube *A* is the most relevant task of the privacy preserving OLAP framework we propose. To this end, we adopt the *strategy of sampling query regions according to the partitioned representation defined by their accuracy grids*. This strategy is named as *accuracy-grid-constrained sampling*. On the basis of this strategy, *samples are extracted from cells of accuracy grids*, according to *a vision that considers the elementary cell of accuracy grids as the atomic unit of our reasoning*. This assumption is well-founded under the evidence of noticing that, given a query Q_k of *QWL*, and the collection of its sub-queries $q_{k,0}$, $q_{k,1}$, …, $q_{k,m-1}$ defined by the accuracy grid $\mathcal{G}(Q_k)$ of Q_k, sampling the (sub-)query regions $R(q_{k,0})$, $R(q_{k,1})$, …, $R(q_{k,m-1})$ of $R(Q_k)$ allows us to (*i*) efficiently answer sub-queries $q_{k,0}$, $q_{k,1}$, …, $q_{k,m-1}$, as sampling is accuracy-grid-constrained, and, at the same time, (*ii*) efficiently answer the super-query Q_k, being the answer to Q_k given by the summation of the answers to $q_{k,0}$, $q_{k,1}$, …, $q_{k,m-1}$ (recall that we consider range-SUM queries). It is a matter of fact to note that the alternative solution of sampling the super-query Q_k directly, which we name as *region-constrained sampling*, would expose us to the flaw of being unable to efficiently answer the sub-queries $q_{k,0}$, $q_{k,1}$, …, $q_{k,m-1}$ of Q_k, since there could exist the risk of having (sub-)regions of Q_k characterized by *high density* of samples, and (sub-)regions of Q_k characterized by *low density* of samples.

3.2. The Allocation Phase. Given the input data cube *A*, the target query-workload *QWL*, and the storage space *B*, in order to compute the synopsis data cube *A' the first issue to be considered is how to allocate B across query regions of QWL*. Given a query region $R(Q_k)$, allocating an amount of storage space to $R(Q_k)$, denoted by $B(Q_k)$, corresponds to assign to $R(Q_k)$ a certain number of samples that can be extracted from $R(Q_k)$, denoted by $N(Q_k)$. To this end, during the allocation phase of our technique, we *assign more samples to those query regions of QWL having skewed* (*i.e., irregular and asymmetric*) *data distributions* (*e.g., Zipf*)*, and less samples to those query regions having Uniform data distributions*. The idea underlying such an approach is that few samples are enough to "describe" Uniform query regions as data distributions of such regions are "regular", whereas we need more samples to "describe" skewed query regions as data distributions of such regions are, contrary to the previous case, not "regular". Specifically, we face-off the deriving allocation problem by means of a *proportional storage space allocation scheme*, which allows us to efficiently allocate *B* across query regions of *QWL* via assigning a *fraction* of *B* to each region. This allocation scheme has been preliminarily proposed in [3] for the different context of approximate query answering techniques for two-dimensional OLAP data cubes, and, in this work, it is extended as to deal with multidimensional data cubes and (query) regions.

First, if *QWL* is overlapping (see Section 2.1), we compute its *equivalent non-overlapping query-workload*, denoted by *QWL'*, by adequately projecting query ranges. Hence, in both cases (i.e., *QWL* is overlapping or not) a set of regions $R(QWL) = \{R(Q_0), R(Q_1), \ldots, R(Q_{|QWL|-1})\}$ is obtained. Let $R(Q_k)$ be a region belonging to $R(QWL)$, the amount of storage space allocated to $R(Q_k)$, $B(Q_k)$, is determined according to a proportional approach that considers (*i*) the nature of the data distribution of $R(Q_k)$ and geometrical issues of $R(Q_k)$, and (*ii*) the latter parameters of $R(Q_k)$ in proportional comparison with the same parameters of all the regions in $R(QWL)$, as follows: $B(Q_k) = \left\lfloor \frac{\varphi(R(Q_k)) + \Psi(R(Q_k)) \cdot \xi(R(Q_k))}{\sum_{h=0}^{|QWL|-1} \varphi(R(Q_k)) + \sum_{h=0}^{|QWL|-1} \Psi(R(Q_k)) \cdot \xi(R(Q_k))} \cdot B \right\rfloor$, such that [3]: (*i*) $\Psi(R)$ is a Boolean *characteristic function* that, given a region *R*, allows us to decide if data in *R* are Uniform or skewed; (*ii*) $\varphi(R)$ is a factor that captures the *skewness* and the *variance* of *R* in a combined manner; (*iii*) $\xi(R)$ is a factor that provides the ratio between the skewness of *R* and its standard deviation, which, according to [8], allows us to estimate the *skewness degree* of the data distribution of *R*. Previous formula can be extended as to handle the overall allocation of *B* across regions of *QWL*, thus achieving the formal definition of our proportional storage space allocation scheme, denoted by $\mathcal{W}(A,R(Q_0),R(Q_1),\ldots,R(Q_{|QWL|-1}),B)$, via the following system:

$$\begin{cases} B(Q_0) = \left\lfloor \dfrac{\varphi(R(Q_0)) + \Psi(R(Q_0)) \cdot \xi(R(Q_0))}{\sum_{k=0}^{|QWL|-1} \varphi(R(Q_k)) + \sum_{k=0}^{|QWL|-1} \Psi(R(Q_k)) \cdot \xi(R(Q_k))} \cdot B \right\rfloor \\ \ldots \\ B(Q_{|QWL|-1}) = \left\lfloor \dfrac{\varphi(R(Q_{|QWL|-1})) + \Psi(R(Q_{|QWL|-1})) \cdot \xi(R(Q_{|QWL|-1}))}{\sum_{k=0}^{|QWL|-1} \varphi(R(Q_k)) + \sum_{k=0}^{|QWL|-1} \Psi(R(Q_k)) \cdot \xi(R(Q_k))} \cdot B \right\rfloor \\ \sum_{k=0}^{|QWL|-1} B(Q_k) \leq B \end{cases} \qquad \textbf{(1)}$$

3.3 The Sampling Phase. Given an instance of our proportional allocation scheme (1), $\mathcal{W}$, during the second phase of our technique, we sample the input data cube *A* in order to obtain the synopsis data cube *A'*, in such a way as to satisfy the accuracy/privacy constraint with respect to the target query-workload *QWL*. To this end, we apply a different strategy in dependence on the fact that query regions characterized by Uniform or skewed distributions are handled, according to similar insights that have inspired our allocation technique (see Section 3.1). Specifically, for a skewed region $R(q_{k,i})$, given the maximum number of samples that can be extracted from $R(q_{k,i})$, $N(q_{k,i})$, we *sample the* $N(q_{k,i})$ *outliers of* $q_{k,i}$. It is worthy to notice that, for skewed regions, *sum of outliers represents an accurate estimation of the sum of all the data cells contained within such regions*. Also, it should be noted that this approach allows us to gain advantages with respect to approximate query answering as well as the privacy preservation of sensitive ranges of multidimensional data of skewed regions. Contrary to this, for a Uniform region $R(q_{k,i})$, given the maximum number of samples that can be extracted from $R(q_{k,i})$, $N(q_{k,i})$, let (*i*) $\overline{C}_{R(q_{k,i})}$ be the average of values of data cells contained within $R(q_{k,i})$, (*ii*) $\mathcal{U}(R(q_{k,i}),\overline{C}_{R(q_{k,i})})$ be the set of data cells C in $R(q_{k,i})$ such that $value(C) > \overline{C}_{R(q_{k,i})}$, where $value(C)$ denotes the value of C, and (*iii*) $\overline{C}^{\uparrow}_{R(q_{k,i})}$ be the average of values of data cells in $\mathcal{U}(R(q_{i,k}),$ $\overline{C}_{R(q_{k,i})})$, we adopt the strategy of extracting $N(q_{k,i})$ samples from $R(q_{k,i})$ by selecting them as the $N(q_{k,i})$ *closer-to-*$\overline{C}^{\uparrow}_{R(q_{k,i})}$ *data cells* C *in* $R(q_{k,i})$ *such that* $value(C) > \overline{C}_{R(q_{k,i})}$.

In order to satisfy the accuracy/privacy constraint, the sampling phase aims at accomplishing (decomposed) accuracy and privacy constraints *separately*, based on a two-step approach Given a query region $R(Q_k)$, we *first* sample $R(Q_k)$ in such a way as to satisfy the accuracy constraint, and, *then*, we check if samples extracted from $R(Q_k)$ *also* satisfy, beyond the accuracy one, the privacy constraint. Moreover, our sampling strategy aims at obtaining a *tunable* representation of the synopsis data cube *A'*, which can be *progressively refined* until the accuracy/privacy constraint is satisfied as much as possible. This means that, given the input data cube *A*, we first sample *A* in order to obtain the *current* representation of *A'*. If such a representation satisfies the accuracy/privacy constraint, then the *final* representation of *A'* is achieved, and used at query time to answer queries instead of *A*. Otherwise, if the current representation of *A'* does not satisfy the accuracy/privacy constraint, then we perform "corrections" on the current representation of *A'*, thus refining such representation in order to obtain a final representation that satisfies the constraint, on the basis of a best-effort approach. What we call the *refinement process* (described in Section 3.3) is based on a greedy approach that *"moves" samples from regions of QWL whose queries satisfy the accuracy/privacy constraint to regions of QWL whose queries do not satisfy the constraint, yet ensuring that the former do not violate the constraint*.

Given a query Q_k of the target query-workload *QWL*, we say that Q_k satisfies the accuracy/privacy constraint iff the following inequalities simultaneously hold: $\begin{cases} E_Q(Q_k) \leq \Phi_Q \\ E_I(Q_k) \geq \Phi_I \end{cases}$.

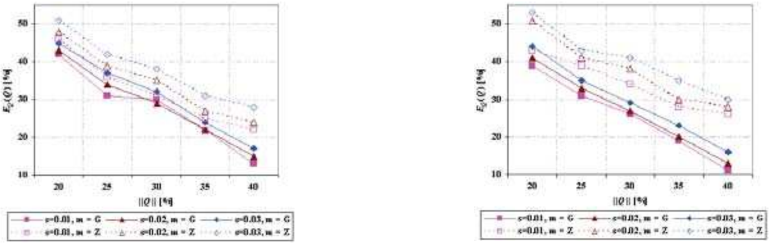
In turn, given a query-workload *QWL*, we decide about its *satisfiability* with respect to the accuracy/privacy constraint by inspecting the satisfiability of queries that compose *QWL*. Therefore, we say that *QWL* satisfies the accuracy/privacy constraint iif the following inequalities simultaneously hold: $\begin{cases} \overline{E}_Q(QWL) \leq \Phi_Q \\ \overline{E}_I(QWL) \geq \Phi_I \end{cases}$.

Given the target query-workload *QWL*, the criterion of our greedy approach used during the refinement process is the *minimization* of the average relative query error, $\overline{E}_Q(QWL)$, and the *maximization* of the average relative inference error, $\overline{E}_I(QWL)$, within the *minimum* number of movements that allows us to accomplish both the goals simultaneously (i.e., minimizing $\overline{E}_Q(QWL)$, and maximizing $\overline{E}_I(QWL)$). Furthermore, the refinement process is bounded by a *maximum occupancy of samples moved across queries of QWL*, which we name as *total buffer size* and denote as $\mathcal{L}_{A',QWL}$. $\mathcal{L}_{A',QWL}$ depends on several parameters such as the size of the buffer, the number of sample pages moved at each iteration, the overall available swap-memory etc.

3.4 The Refinement Phase. In the refinement process, the third phase of our technique, given the current representation of *A'* that does *not* satisfy the accuracy/privacy constraint with respect to the target query-workload *QWL*, we try to obtain an alternative representation of *A'* that satisfies the constraint, according to a best-effort approach. To this end, the refinement process encompasses the following steps: (*i*) sort queries in *QWL* according to their "distance" from the satisfiability condition, thus obtaining the ordered query set $QWL^{\mathcal{P}}$; (*ii*) select from $QWL^{\mathcal{P}}$ a pair of queries $Q^{\mathcal{T}}$ and $Q^{\mathcal{F}}$ such that (*ii.j*) $Q^{\mathcal{T}}$ is the query of $QWL^{\mathcal{P}}$ having the *greater positive distance* from the satisfiability condition, i.e. $Q^{\mathcal{T}}$ is the query of $QWL^{\mathcal{P}}$ that has the greater *surplus* of samples that can be moved towards queries in $QWL^{\mathcal{P}}$ that do not satisfy the satisfiability condition, and (*ii.jj*) $Q^{\mathcal{F}}$ is the query of $QWL^{\mathcal{P}}$ having the *greater negative distance* from the satisfiability condition, i.e. $Q^{\mathcal{F}}$ is the query of $QWL^{\mathcal{P}}$ that is in most need for new samples; (*iii*) move enough samples from $Q^{\mathcal{T}}$ to $Q^{\mathcal{F}}$ in such a way as to satisfy the accuracy/privacy constraint on $Q^{\mathcal{F}}$ while, at the same time, ensuring that $Q^{\mathcal{T}}$ does not violate the constraint; (*iv*) repeat steps (*i*), (*ii*), and (*iii*) until the current representation of *A'* satisfies, as much as possible, the accuracy/privacy constraint with respect to *QWL*, within the maximum number of iterations bounded by $\mathcal{L}_{A',QWL}$. For what regards step (*iii*), moving ρ samples from $Q^{\mathcal{T}}$ to $Q^{\mathcal{F}}$ means: (*i*) removing ρ samples from $R(Q^{\mathcal{T}})$, thus obtaining an *additional* space, said $B(\rho)$; (*ii*) allocating $B(\rho)$ to $R(Q^{\mathcal{F}})$, (*iii*) re-sampling $R(Q^{\mathcal{F}})$ by considering the additional number of samples that have became available – in practice, this means extracting from $R(Q^{\mathcal{F}})$ further ρ samples.

4 Experimental Evaluation

In order to test the effectiveness of our framework throughout studying the performance of our technique, we conducted an experimental evaluation where we tested how the relative query error (similarly, the accuracy of answers) and the relative inference error (similarly, the privacy of answers) due to the evaluation of populations of randomly-generated queries, which model query-workloads of our framework, over the synopsis data cube range with respect to the volume of queries. The latter is a relevant parameter costing computational requirements of any query processing algorithm (also referred as *selectivity* – e.g., see [4]). According to motivations given in Section 1, we considered the *Zero-Sum* method [9] as the comparison technique, being [9] the state-of-the-art perturbation-based privacy preserving OLAP technique available in literature. In our experimental assessment, we engineered two classes of two-dimensional synthetic data cubes: the *Uniform data cube*, which has been obtained by means of a Uniform distribution, and the *skewed data cube*, which has been obtained by means of a Zipf distribution.

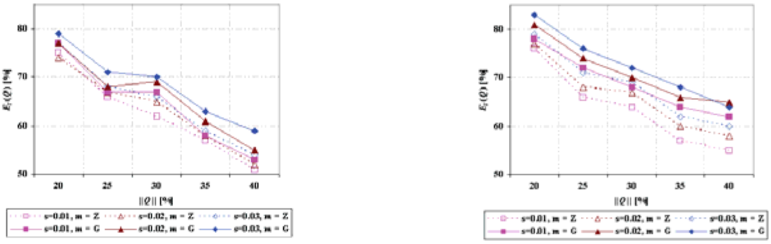


(*a*) (*b*)

Fig. 1. Relative query errors of synopsis data cubes built from Uniform (*a*) and skewed (*b*) data cubes.

To simplify, we set the accuracy and privacy thresholds in such a way as not to trigger the refinement process. This also because [9] does not support any "dynamic" computational feature (e.g., tuning of the quality of the random data distortion technique), so that it would have been particularly difficult to compare the two techniques under completely-different experimental settings. On the other hand, this aspect puts in evidence the innovative characteristics of our privacy preserving OLAP technique with respect to [9], which is indeed a state-of-the-art proposal in perturbation-based privacy preserving OLAP techniques.

Figure 1 shows experimental results concerning relative query errors of synopsis data cubes built from Uniform and skewed data, respectively, and for several values of the *sparseness coefficient s* [4]. Figure 2 shows instead the results concerning relative inference errors on the same data cubes. In both Figures, our approach is labeled as *G*, whereas [9] is labeled as *Z*. Obtained experimental results confirm the effectiveness of our algorithm, also in comparison with [9].



(*a*) (*b*)

Fig. 2. Relative inference errors of synopsis data cubes built from Uniform (*a*) and skewed (*b*) data cubes.

5 Conclusions and Future Work

A complete framework for efficiently supporting privacy preserving OLAP aggregations on data cubes has been presented and experimentally assessed in this paper. We rigorously presented theoretical foundations, as well as intelligent techniques for processing data cubes and queries, and algorithms for computing the final synopsis data cube whose aggregations balance, according to a best-effort approach, accuracy and privacy of retrieved answers. An experimental evaluation conducted on synthetic data cubes has clearly demonstrated the benefits deriving from the privacy preserving OLAP technique we propose, also in comparison with a state-of-the-art proposal. Future work is mainly oriented towards extending the actual capabilities of our framework in order to encompass intelligent update management techniques (e.g., what happens when query-workload's characteristics change dynamically over time?), perhaps inspired by well-known principles of *self-tuning databases*.

References

[1] R. Agrawal et al., "Privacy-Preserving OLAP", *ACM SIGMOD*, 251—262, 2005.

[2] A. Cuzzocrea, "Overcoming Limitations of Approximate Query Answering in OLAP", *IEEE IDEAS*, 200—209, 2005.

[3] A. Cuzzocrea, "Improving Range-Sum Query Evaluation on Data Cubes via Polynomial Approximation", *Data & Knowledge Engineering*, 56(2), 85—121, 2006.

[4] A. Cuzzocrea, "Accuracy Control in Compressed Multidimensional Data Cubes for Quality of Answer-based OLAP Tools", *IEEE SSDBM*, 301—310, 2006.

[5] J. Gray et al., "Data Cube: A Relational Aggregation Operator Generalizing Group-By, Cross-Tab, and Sub-Totals", *Data Mining and Knowledge Discovery*, 1(1), 29—54, 1997.

[6] M. Hua et al., "FMC: An Approach for Privacy Preserving OLAP", *DaWaK*, 408—417, 2005.

[7] A. Machanavajjhala et al., "*L*-diversity: Privacy beyond *k*-Anonymity", *ACM Trans. on Knowledge Discovery from Data*, 1(1), art. no. 3, 2007.

[8] A. Stuart et al., *Kendall's Advanced Theory of Statistics, Vol. 1: Distribution Theory*, 6th ed., Oxford University Press, New York City, NY, USA, 1998.

[9] S.Y. Sung et al., "Privacy Preservation for Data Cubes", *Knowledge and Information Systems*, 9(1), 38—61, 2006.

[10] L. Sweeney, "*k*-Anonymity: A Model for Protecting Privacy", *International Journal on Uncertainty Fuzziness and Knowledge-based Systems*, 10(5), 557—570, 2002.

Hand-draw sketching for image retrieval through fuzzy clustering techniques

Ugo Erra[1] and Sabrina Senatore[2]

[1] Università della Basilicata, Dipartimento di Matematica e Informatica,
Viale Dell'Ateneo 10, Macchia Romana - 85100, Potenza, Italy
ugo.erra@unibas.it

[2] Università di Salerno, Dipartimento di Informatica
Via Ponte Don Melillo - 84084 Fisciano(SA), Italy
ssenatore@unisa.it

Abstract. Nowadays, the growing of digital media such as images represents an important issue for multimedia mining applications. Since the traditional information retrieval techniques developed for textual documents do not support adequately these media, new approaches for indexing and retrieval of images are needed. In this paper, we propose an approach for retrieving image by hand-drawn object sketch. For this purpose, we address the classification of images based on shape recognition. The classification is based on the combined use of geometrical and moments features extracted by a given collection of images and achieves shape-based classification through fuzzy clustering techniques. Then, the retrieval is obtained using a hand-draw shape that becomes a query to submit to the system and get ranked similar images.

1 Introduction

The idiom 'a picture is worth a thousand words' refers to the concept that a single image can be used to quickly describe an idea but it also suggests us that we can not describe succinctly an image based on few words. Humans tend to describe an image using short sentences with keywords and pointing out different parts of the image, according to their cultural and professional background. On the other hand, finding an image that is close to a hand-draw sketch is a natural approach that fits well in the age of digital information where the growing amount of large-scale image repositories in many application domains emphasize the need for effective and practical means for retrieving digital images.

In general, two different approaches have been applied for image retrieval. The first one consists of attaching textual metadata to each image, then a keyword-based query is submitted to the database in order to retrieve relevant images [9]. This approach requires an initial time-consuming activity of images annotation activity; moreover, it is a laborious, human driven process that can affect the performance of the keyword-based image search, according to the naming and terminology used for annotate the images. The second approach, named Content-Based Image Retrieval (CBIR) exploits the features which characterize *objective* image properties such as color, texture, and shape. These techniques improve the effectiveness of image retrieval through multi-features combination [5] and then, by measuring similarity to a required query image [11]. But, the user does not always have a such image. An alternative to an image as a query is using a line-based hand-drawing, i.e., a sketch as a natural way to make a query [10]. Comparing a rough sketch to an image is a natural yet difficult task. Few approaches deal with sketch-based image retrieval (SBIR).

In [3] query-by-sketch exploits the spatial relationships between shapes in an image. The approach represents shapes, their spatial arrangement, color and texture attributes in the user sketch; then the image content is extracted starting from basic features and combining them in a higher-level description of spatial layout of components. Also interesting is the work in [7], which exploits wavelet-based indexing and query by sketch for color images retrieval. In some approaches, the use of fuzzy techniques can support the image description. In general, fuzzy retrieval models offer more flexibility in the representation of the terms' index, preferences among terms in a query which involve concepts and linguistic expression, through fuzzy values rather than crisp features values [8].

In this paper, we specifically discuss our approach to image retrieval based on the hand-draw sketch that represents a rough approximation of query image contour. The approach exploits fuzzy clustering techniques for image classification. The purpose of using the fuzzy clustering technique is twofold. First, the fuzzy clustering technique can reflect better the imprecise nature of images yielding an adequate classification of images collection that enable the search on a reduced search space. Second, the retrieval based on a query-by-sketch provides accurate results, evidencing the efficacy of such approach.

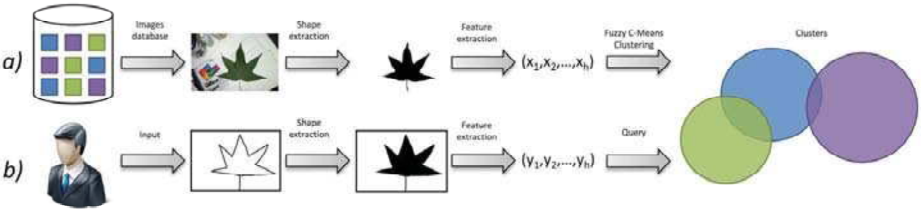


Fig. 1. Overview of the proposed approach.

2 Overview

Our approach accomplishes two stages. The first one performs an off-line images classification, through a fuzzy clustering technique (Fig. 1a), the second one instead, performs the query by hand-draw image sketch (Fig. 1b). Preliminary activity is shape and features extraction. The shape extraction separates the object (or region) of interest from other non-important image structures. There are several approaches for the extraction of the shape from a given image based on clustering methods, histogram methods, edge detection, level set methods, graph partitioning methods and so on. There is no a general methodological approach to reach this target: many factors and parameters (i.e., the background color, the contour, etc.) can affect the result. In our implementation, we adopt the k-means clustering algorithm for image segmentation which is suitable when the foreground and background colors contrast sufficiently with each other. Afterwards, a features extraction step gets the set of features from the region of interest that characterizes the image. Efficient shape features must have some essential properties such as identifiability, invariance, noise resistance, statistically independence and so on. We use several geometric description and image moments which are invariant to translation, rotation and scaling (Sect. 3).

The clustering algorithm achieves a partitioning of given images into clusters. In general a partition holds two properties: homogeneity within the clusters (data in a cluster must be similar) and homogeneity between clusters (isolation of a cluster from one another: data of different clusters have to be as different as possible). The images are opportunely translated into a matrix, where each row is a characteristic vector which represents an image described by the extracted features. In this study, we exploit the well-known *fuzzy C-Means* (FCM) algorithm [2], which takes as an input a collection of patterns (in our case, the collection of images) and produces fuzzy partitions of the given patterns (i.e. images) into (prefixed) c clusters (Sect 4). The computed clusters are analyzed in order to discover the nature of the data groups (similar images) and associate to each cluster the expected class name. A this point, the second stage can be applied. Query by image is a technique that generally provides the CBIR system with an example image that will base its search upon. The underlying search approach may vary depending on the application, but resulting images should be similar to the given sample query, sharing its common descriptors.

3 Shape descriptors

Three types of shape descriptions are adopted: geometric description, invariant moments and affine moments. The geometric features discriminate shapes with large difference. They are useful to eliminate false hits and usually are not suitable as single description, in fact they are combined with other shape descriptors to better discriminate shapes. The moment instead, represents a mathematical concept coming from the concept of moment in physics. It is used in computer vision for both contour and region of a shape. The invariant moments [6] are one of the most popular and widely used contour-based shape descriptors. Affine moments invariants are instead, features computed from moments that do not change their value in affine transformation. In the case of geometric features, let P and A denote the shape perimeter and area, respectively. Note that perimeter and area are invariants respect to translation and rotation but when combined, they are not invariant with respect to scale. The features we adopt are:

- Eccentricity E is the measure of aspect ratio. It is defined as the ratio $E = W_{bb}/H_{bb}$ where W_{bb} and H_{bb} are, respectively, the width and height of minimal bounding rectangle of the shape.

- Rectangularity R represents how rectangular a shape is, i.e. how much it fills its minimum bounding box. It is defines as $R = A/A_{bb}$ where A_{bb} is the area of the minimum bounding rectangle.
- Compactness C is a measure that combines area with perimeter. It is defined as $C = L^2/4\pi A$.
- The value π_{gen} is a measure of the compactness of a shape respect to a circle. It is defined as $\pi_{gen} = P/W_{bb}$.

Invariant moments m_{pq} are the simplest and is given as:

$$m_{pq} = \sum_x \sum_y x^p y^q f(x,y) \qquad p,q = 0,1,2,\ldots$$

where $f(x,y)$ is the intensity function at position (x,y) in a 2D gray level image. To obtain translation invariance, the central moments μ_{pq} should be applied:

$$\mu_{pq} = \sum_x \sum_y (x-\overline{x})^p (y-\overline{y})^q f(x,y) \qquad p,q = 0,1,2,\ldots$$

where $\overline{x} = m_{10}/m_{00}$ and $\overline{y} = m_{01}/m_{00}$. Given central moments we are able to compute a set of 7 invariant moments [6], given by:

$$\begin{aligned}
I_1 &= \eta_{20} + \eta_{02} \\
I_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
I_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
I_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
I_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} - \eta_{12})^2 - 3(\eta_{21} - \eta_{03})^2] + \\
&\quad (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[(3\eta_{30} + \eta_{12})^2 - (\eta_{21} - \eta_{03})^2] \\
I_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} - \eta_{03})^2] + \\
&\quad 4\eta_{11}^2(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
I_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} - \eta_{12})^2 - 3(\eta_{21} - \eta_{03})^2] + \\
&\quad (3\eta_{12} - \eta_{03})(\eta_{21} + \eta_{03})[(3\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]
\end{aligned}$$

where $\eta_{pq} = \mu_{pq}^{\gamma}$ and $\gamma = 1 + (p+q)/2$ for $p + q = 2, 3, \ldots$. These moments are simple to calculate and they are invariant to translation, rotation and scaling. From central moments with a little computational effort we are able to obtain also an affine transform invariance which includes the similarity transform and in addition to that stretching and second rotation. We adopt affine moments as defined in [4] and given as:

$$\begin{aligned}
AMI_1 &= (\mu_{20}\mu_{02} - \mu_{11}^2)/\mu_{00}^4 \\
AMI_2 &= (\mu_{30}^2\mu_{03}^2 - 6\mu_{30}\mu_{21}\mu_{12}\mu_{03} + 4\mu_{30} + \mu_{12}^3 + \\
&\quad 4\mu_{03}\mu_{21}^3 - 3\mu_{21}^3\mu_{12}^3)/\mu_{00}^{10} \\
AMI_3 &= (\mu_{20}(\mu_{21}\mu_{03} - \mu_{12}^2) - \mu_{11}(\mu_{30}\mu_{03} - \mu_{21}\mu_{12}) + \\
&\quad \mu_{02}(\mu_{30}\mu_{12} - \mu_{21}^2))/\mu_{00}^7
\end{aligned}$$

All these features are sufficient to characterize the shape of an image. The rationale behind the choice of these moments is that we are interesting in translation, rotation, scale, and projective transform invariance in order that the location, orientation, and scaling of the shape do not affect the extracted features.

4 Fuzzy clustering

FCM represents the most common fuzzy clustering, particularly useful for flexible data organization. It takes as input a collection of patterns of a universe U in form of matrix and produces fuzzy partitions of the given patterns into (prefixed) c clusters. In fact, after the fuzzy clustering execution, each pattern has associated a c-dimensional vector, where each cell represents the membership (in the range $[0, 1]$) of that pattern to each cluster. Compared to the crisp version, the fuzzy clustering generates a flexible partitioning, more intuitive to interpret: a pattern can have some characteristics that are natively representative of more than one cluster. In the fuzzy approach, the membership values better reveal the nature of data set and allow a clearer data analysis. Anyway, it is conceivable to assign a pattern to the cluster, whose membership is the highest. More formally, each row of the matrix is a vector that represents an image $I \longleftrightarrow \underline{x} = (x_1, x_2, \ldots, x_h)$, where each component of vector is a value computed for a feature. The FCM algorithm aims at minimizing the objective function constituted by the weighted sum of the distances $dist_{i,k}$ between data points $\underline{x}_k = (x_{k,1}, x_{k,2}, \ldots, x_{k,h})$ and the centers (or prototypes) $\underline{v}_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,h})$, according to this formula:

$$Q(U, c) = \sum_{i=1}^{c} \sum_{k=1}^{n} u_{i,k}^{m} (dist(\underline{x}_k,\ \underline{v}_i))^2$$

where $c \geq 2$ is the number of clusters, $u_{i,k} \in$ [0,1] is the membership degree of $\underline{x}_k$ (k=1, ..., n) in the i-th cluster A_i (i=1, ..., c), and $m > 1$ is the fuzzifier, which controls the quantity of fuzziness in the classification process (common choice of fuzzifier is $m = 2$) and finally $dist(\underline{x}_k,\ \underline{v}_i)$ represents the euclidean distance between the data $\underline{x}_k$ and the center $\underline{v}_i$ of the i-th cluster.

5 Query by hand-draw image

After the FCM execution, the generated fuzzy partitioning emphasizes that each pattern, i.e., each image is assigned to a clustering according to its highest value of membership. Then, output of FCM is the prototypes that are the centers of clusters and represent the more representative elements for each cluster. A query is considered a new image that must be placed in the clusters space, in order to find which cluster it belongs to and which images it is more similar to. Thus, it is processed to extract its features and then translated into a vector-based representation to be homogeneously compared with images in the clusters space. A simple way to get this evaluation is to measure the euclidean distance between the query and the prototypes. The prototype with minimal distance from the query represents the cluster which contains the most similar images. This approach has a drawback due to nature of the clustering, that is fuzzy and thus, the memberships of the images may be affected by some characteristics (features) that are natively representative of more than one cluster. Then, in order to enable user to yield a more accurate results from the hand-draw query, we focus only on patterns that lie inside a given space around the prototype. Formally, given a query s, a number c of centers v_i, and a fuzziness threshold ϵ, we use the following criteria to obtain the cluster membership. The potential cluster j is found by taking into account the minimal euclidean distance between the center and the query, that is

$$i = min_{1 \leq j \leq c} \{j | dist(s, v_j)\}$$

Because we do not have a way to establish the membership of s inside the cluster chosen, we select a reference image x_i belonging to the cluster i whose membership is immediately above the fuzziness threshold. It is obtained as follows.

$$x_i = min_{1 \leq k \leq n} \{x_k | u_{k,i} \geq \epsilon\}$$

where, n is the size of cluster i. Then, if $dist(v_i, s) \leq dist(v_i, x_i)$, the query is in the space computed as a sphere, with radius equals to $dist(v_i, x_i)$. Then, we return all the images contained in the cluster i sorted based on euclidean distance and whose membership is above the fuzziness threshold. This ranked list represents the answer to the given query.

6 Experimental results

The experimentation consists of two testing phases. The first one is related to the classification of the images. The second one evaluates the retrieval through query by sketched image. For the classification experiment, we consider a collection of images downloaded exploiting Google [1]. This testbed consists of a sample of 965 images, composed of six classes of about 160 images.All the features presented above has been exploited in the experiment. A 965 × 14 data matrix as been given as an input of FCM. Let note that the main peculiarity of the fuzzy clustering is that the membership of an image may result distributed on more than one clusters. Typically, an image (in general, a pattern) is associated to the cluster in which its membership is the highest. In order to emphasize the natural flexibility of the fuzzy clustering methods in the distribution of data, we have processed the dataset on crisp k-means too.

Table 1 synthesizes the results, associated to the experiment for the two clustering algorithms. Each cell indeed, contains resulting values computed for k-means and FCM clustering algorithms, respectively. Each row of the matrix represents a cluster obtained by the algorithm execution. Once identified which images are mainly representative of each cluster, the name of the category/class (bottle, guitar, etc.) is associated to the cluster. The first cell of each row represents the cluster/category. Thus, in Table 1, column headings contains the name

Table 1. Cluster-based evaluation, for k-means and FCM clustering, respectively

Classes	*# Misclassified.*	*# Undecided.*	*Recall. %*	*Precision. %*
bottle	21-6	0-0	86-96	93-100
guitar	7-3	0-0	95-98	95-96
leaf	92-58	0-7	44-64	41-78
apple	94-16	0-2	41-90	79-92
motorcycle	17-22	0-5	89-86	57-70
gun	13-13	0-3	91-91	92-93

of the class, the *misclassified* images, i.e. those images that have the highest membership in a class, which is not the expected one, the *undecided* images, viz. all the images which membership is almost equally distributed among two or more clusters. Note that, due to the crisp nature of the k-means algorithm, there are no resulting *undecided* images. Then, *recall* and *precision* is evaluated inside each cluster. Particularly, we define *local recall* and the *local precision*, as follows:

$$Recall = \frac{relevant\ retrieved\ images}{relevant\ images} \quad Precision = \frac{relevant\ retrieved\ images}{retrieved\ images}$$

where the *relevant images* are the images which are expected in a certain class, the *retrieval images* are all the (correct and incorrect) images which are returned in that cluster, while the *relevant retrieved images* are just the images that really belong to the right cluster, associated to the correct class. By analyzing the results, the set of leaf is the worst classified. Let us note that, comparing to each other the classes of images, the class of leaf is composed of very heterogeneous image samples: leaves with one or more tips, maple leaves are put together, without considering the obvious differences in the forms. In fact, most of misclassified data appear in cluster of apples; this is due to the different shapes of leaves: after the image processing, some leaves present rounded shapes (such as walnut and cherry leaves) that can be easily confused with apples. In fact, geometrical features such as π_{gen} and compactness assume values similar to those ones of apples. Similar considerations arise by comparing some leaves with motorcycles shapes. That means, the class of leaves is not adequately representative, because its individual are confused with other classes. A possible consequence is the amount of false positive (i.e., retrieved elements that are wrongly considered to belong to a class) increases, affecting negatively the precision. However, the global performance of FCM clearly overcomes the crisp clustering, that reveals its intrinsic weakness in clustering images. The second stage of this experiment considers the process retrieval, given a query-by-sketch. We have hand-drawn a sketch and according to Section 5, we retrieve the more similar images. In this experiment, shape extraction algorithm plays a key role in retrieving similar images results. We exploit the shape to generate the feature vector associated to an image and then, to return the similar images. Thus, the quality of result depends on the validity of extracted shapes. Figure 2 shows the results of three queries. Note that for the first and second sketches, the matches provide a very reasonable answer to the user query in the data base, while for the third sketch, the matches provide also wrong answers, due to misclassification of leaves dataset.

7 Conclusion

This paper proposes an approach for retrieving image by hand-drawn object sketch. The use of fuzzy clustering allows a better representation of the image domain: the recall and precision measures reveal a discrete performance by combining geometrical and moments features. Then, the shape query submitted as the free hand drawing evidences the effectiveness in the retrieving of returns

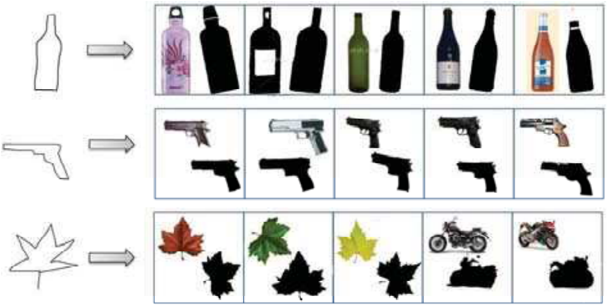


Fig. 2. Queries by sketch: ranked list of similar images based on the submitted queries. For each query, similar image and extracted shape are returned.

relevant similar images. Future extensions of this work foresee a development of a GUI-based application which supports the features extraction, the clustering technique; moreover a tool for hand-draw images to submit to the system.

References

1. The dataset is available at:. `http://www.dmi.unisa.it/people/senatore/www/dati/dataset.rar`.
2. J. C. Bezdek. *Pattern Recognition with Fuzzy Objective Function Algorithms.* Kluwer Academic Publishers, Norwell, MA, USA, 1981.
3. E. Di Sciascio, F. M. Donini, and M. Mongiello. Spatial layout representation for query-by-sketch content-based image retrieval. *Pattern Recogn. Lett.*, 23:1599–1612, November 2002.
4. J. Flusser and T. Suk. Pattern recognition by affine moment invariants. *Pattern Recognition*, 26(1):167–174, 1993.
5. J.-Y. Ha, G.-Y. Kim, and H.-I. Choi. The content-based image retrieval method using multiple features. In *Proc. of the 2008 Fourth International Conference on Networked Computing and Advanced Information Management*, pages 652–657, Washington, DC, USA, 2008. IEEE Computer Society.
6. M.-K. Hu. Visual pattern recognition by moment invariants. *Information Theory, IRE Transactions on*, 8(2):179 –187, february 1962.
7. C. E. Jacobs, A. Finkelstein, and D. H. Salesin. Fast multiresolution image querying. In *Proc. of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 277–286, New York, NY, USA, 1995. ACM.
8. R. Krishnapuram, S. Medasani, S.-H. Jung, Y.-S. Choi, and R. Balasubramaniam. Content-based image retrieval based on a fuzzy approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(10):1185–1199, 2004.
9. H. Lieberman, E. Rozenweig, and P. Singh. Aria: An agent for annotating and retrieving images. *Computer*, 34:57–62, 2001.
10. J. M. Saavedra and B. Bustos. An improved histogram of edge local orientations for sketch-based image retrieval. In *Proc. of the 32nd DAGM conference on Pattern recognition*, pages 432–441, Berlin, Heidelberg, 2010. Springer-Verlag.
11. D. Zhang and G. Lu. Evaluation of mpeg-7 shape descriptors against other shape descriptors. *Multimedia Syst.*, 9(1):15–30, 2003.

From service identification to service selection: an interleaved perspective (extended abstract)

Devis Bianchini*, Francesco Pagliarecci+ and Luca Spalazzi+

* Dipartimento di Ingegneria dell'Informazione
Universita' degli Studi di Brescia, Via Branze, 38, 25123 Brescia
{bianchin}@ing.unibs.it

+ Dipartimento di Ingegneria Informatica, Gestionale e dell'Automazione
Universita' Politecnica delle Marche, Via Brecce Bianche, 60131 Ancona
{pagliarecci|spalazzi}@univpm.it

Abstract. Business process implementation can be fastened by identifying component services that can be used to implement one or more process tasks and by selecting them from a repository of already implemented services. In this paper, we propose an on-going work for the design of an iterative procedure to address this issue, by combining the two macro-phases of service identification and service selection. Starting from a workflow-based specification of the business process, service identification is firstly executed. The result of this phase is a decomposition tree, where basic process tasks are progressively organized into sub-processes (the candidate services) by applying an agglomerative clustering algorithm, based on cohesion and coupling metrics. Within the decomposition tree, a set of candidate services that minimize the coupling/cohesion ratio for the overall process is chosen. The service selection phase works on this decomposition and looks for available services. If the service selection phase fails for some candidate services, a revised set of candidate services is selected by leveraging on the decomposition tree.

1 Introduction

Business process implementation can be fastened by identifying component services that can be used to implement one or more process tasks and by selecting them from a repository of already implemented services. Service identification is a debated topic in the recent literature. It is defined as a procedure which starts from the business process specification and decomposes it into candidate component services, that can be used to implement one or more process tasks [1, 2]. Candidate services can be either retrieved among existing ones or implemented from scratch. According to this perspective, service identification comes before service selection or service implementation and precedes service invocation and deployment. Service identification and service selection are mainly considered as distinct phases, sequentially executed.

In this paper, we investigate a different perspective, where the service identification and selection phases are more interleaved. Starting from the methodology presented in [2], service identification is firstly executed. The result of this phase is a decomposition tree, where basic process tasks are progressively organized into sub-processes (the candidate services) by applying an agglomerative clustering algorithm, based on cohesion and coupling metrics already applied in software engineering field. Within the decomposition tree, a set of candidate services that minimize the coupling/cohesion ratio for the overall process is chosen. The service selection phase starts from this decomposition and looks for available services, by applying techniques already presented in [9]. If the service selection phase fails for some candidate services, a feedback is propagated back to the service decomposition phase, that proposes a revised set of candidate services by relying on the decomposition tree. The feedback strategy constitutes a step forwards with respect to work in [2] and [9]. In fact, the methodology in [2] does not take into account the effective availability of existing component services. On the other hand, the selection procedure described in [9] can take advantage by relying on a preliminary identification of services to be searched for. The result is a business process decomposition that takes into account both given guidelines for service design, such as high cohesion and low coupling (*to-be* perspective) and concrete services actually available and already implemented, among which candidate services must be selected (*as-is* perspective).

The paper is organized as follows: in the next section some related work are discussed; in Section 3 we present an application scenario and our vision; Section 4 and Section 5 summarize the service identification and selection phases; Section 6 describes their iterative combination; finally, Section 7 closes the paper.

2 Related Work

Service indentification. Ghosh et al. [5] listed three kinds of service identification procedures: *top-down*, *bottom-up* and *meet-in-the-middle*. Top-down procedures focus on analysis of business domains and business process modeling to identify services, components and flows. In [11] authors guide the service designer by suggesting the order through which the different techniques should be used and providing some tips for the evaluation of the results; in [6] authors define a set of heuristics to support service identification, but do not propose any quantitative model to efficiently support the analysis. In [10] a bottom-up approach is presented; authors introduce process templates, which are reusable process skeletons devised to reach particular goals and made up of states and transitions; a state corresponds to the execution of a service (called component service) that is selected from a Web service community. In [5] a meet-in-the-middle strategy is followed. Identified services are grouped according to some kind of logical affinity. Subsequently, the SLT (rationalization) is applied as a set of criteria to resolve whether a candidate service should be exposed.

Service selection. The increasing availability of Web services that offer similar functionalities with different features increases the need for more sophisticated selection processes to match user requests. Most of the existing techniques rely on syntactic descriptions of service interfaces to find Web services with disregard to non- functional service parameters. Previous research demonstrates how this situation generates major problems [7, 8]. To solve some of them, Web service descriptions are enhanced with annotations through ontological concepts and semantic matching of functional and non-functional properties (NFPs) [3]. The service selection problem is also investigated in [13] by using a combinatorial model and a graph model. In [12], a formal service model is defined and a dynamic programming based approach is proposed to select the best service providers.

3 Problem statement

Let consider a collaborative environment, where distinct partners contribute together to a business goal. Collaboration can be expressed as a business process using a workflow-based notation (e.g., BPMN), independent from implementation technologies and platforms. A business process $\mathcal{B}P$ can be represented as a set of simple tasks, combined through control flow structures (e.g., sequence, choice, cycle or parallel) and specified through the performed operations and I/O data flow between them. Collaborative business processes are designed as processes spanning over partners, that interact each other as responsible of one or more sub-processes. Beyond the platform-independent implementation of the collaborative business process $\mathcal{B}P$, we consider a repository of implemented services $\overline{\mathcal{S}}$, represented at the process level (based on BPEL) with a minimal semantic annotations and a language that can express requirements on the behavior of the service that has to be selected. The business process $\mathcal{B}P$ can be decomposed into a set of subprocesses $\mathcal{S}$, that can be totally or partially implemented by services in the repository. Each partner provides one or more services in the repository. Let's denote with $\mathcal{S}'$ the subprocesses which can be implemented through one of the services in the repository. The problem we address in this paper is the identification of the set $\mathcal{S}$ such that: $\mathcal{S}' \subseteq \mathcal{S}$, and it does not exist another $\mathcal{S}'' \sqsubseteq \mathcal{S}$ such that $\mathcal{S}' \sqsubseteq \mathcal{S}''$ (that is, $\mathcal{S}'$ is the decomposition that better exploits the set of available services). In this way, given a new platform-independent business process specification, the goal is to identify those partners that contribute to the new business goals by implementing one or more subprocesses through their services.

As a case study, we consider an application from the computer supplying domain. A computer retailer receives computer orders and, after their approval, generates the bill of materials (BOM) and sends orders for components to a subset of his/her suppliers. The procedure to order components is different depending on the type of the suppliers. In particular, for external ones, an invoice is prepared and sent to the retailer. After receiving all the components, the retailer assembles the computer, prepares the final invoice and ships the product to the client. The workflow-based, platform-independent representation of the process is shown in the upper part of Figure 1. We distinguish between the *process level*, where the design of the collaborative business process is represented, and the *semantic service level*, where semantic-enriched descriptions of component services that implement one or more sub-processes are identified.

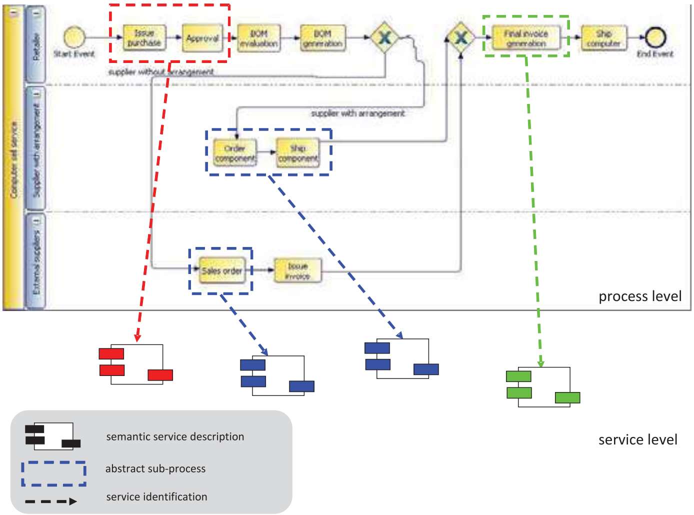


Fig. 1. Running example.

4 Service identification

The service identification methodology has been extensively described in [2]. The methodology is mainly based on the notion of component service as a particular kind of sub-process, where the following properties hold: (i) services are self-contained and interact each other using decoupled message exchanges (high cohesion, low coupling); (ii) each service is the minimal set of tasks that performed together create an output that is a tangible value for one of the actors involved in the overall process execution. In [2] a value has been defined as an intermediary process output that is not used as input of another task of the same partner in the collaborative process (for example, the invoice issued by the external supplier is a value for the retailer).

The best set of candidate component services is identified, according to the features listed above, through the following steps:

value-based identification of candidate services - value exchanges are identified throughout the process flow and the process is split into a preliminary list of candidate services;

refinement of the process decomposition - the overall coupling/cohesion ratio on the set of candidate services is evaluated and is minimized by aggregating/splitting the preliminary set of candidate services (see [2] for details);

reconciliation of similar services - component services must be clustered on the basis of the similarity of their tasks and I/O data, in order to identify similar services represented at different granularity (that is, number of simple tasks which compose the services) and enable the design of reusable component services (for example, the {`OrderComponent,ShipComponent`} and the `SalesOrder` subprocesses in the case study).

5 Service selection

The service selection phase detailed in [9] looks for semantic-enriched Web services that satisfy the workflow-based representation of the required component by means of a Semantic Model Checking. The service selection phase is therefore based on the following key ideas:

requirement specification - the abstract workflow-based representation of the required service to be searched for is expressed by means of a BPMN diagram where tasks are semantically annotated; the ontological (semantics) part of the specification is expressed by means of semantic annotations written in SWSAL [4]; this language allows a user to specify which tasks a service must implement and in which order they must be executed; this representation is easily (automatically) translated into a temporal logic specification, using the Computation Tree Logic (CTL) enriched with (concept and role) assertions of a Description Logic (assertions are used instead of atomic propositions as in the traditional CTL); this logic is called *Annotated* CTL (A*n*CTL);

service specification - behavior of available services is represented by means of the BPEL language where activities are semantically annotated; this representation is easily (automatically) modeled as a state transition system with semantic annotation; this model is called *Annotated State Transition Systems* (ASTS);

Semantic Model Checking - the problem of verifying whether an implemented service (modeled as an ASTS) satisfies an abstract specification (represented by an A*n*CTLformula) can be solved by means of the algorithm of Semantic Model Checking that has been defined in [9]; this algorithm has been proved to be sound and complete; its complexity depends on the expressiveness of the description logic that has been used and ranges from PTIME to CONP.

6 Interleaving service identification and selection phases

The service identification and selection phases have been combined as follows. The first step is the construction of a decomposition tree by means of an agglomerative clustering of tasks based on their coupling. The leaves of the three are single tasks and intermediate nodes are subprocesses collecting tasks which present high coupling. Tasks with higher coupling are grouped first. The root of the tree represents the overall process. The decomposition tree for the running example is shown in Figure 2. The nodes of the decomposition tree are all potential services that could be identified. The service identification phase is applied to identify the best set of candidate component services which present the minimum ratio between their mutual coupling and their average internal cohesion (see [2] for details). A weight is associated to each node to denote the variation of the ratio. A negative weight means that the split of the node into its children decreases the ratio. For example, if the mode $\mathcal{S}_{4567}$ is split into its children $\mathcal{S}_{45}$ and $\mathcal{S}_{67}$ the ratio is decreased by 0.25. A positive weigth means that the split of the node into its children increases the ratio. In the running example, the candidate component services identified are $\Sigma = \{\mathcal{S}_{01}, \mathcal{S}_2, \mathcal{S}_3, \mathcal{S}_{45}, \mathcal{S}_{67}, \mathcal{S}_{89}\}$, as shown in the tree in Figure 2 by means of the dashed line.

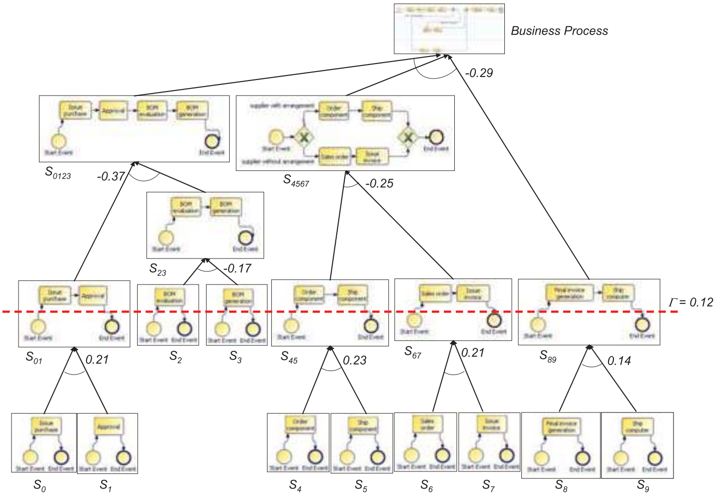


Fig. 2. Decomposition tree for the running example.

We use the BPMN workflow and the semantic annotations to generate semantically annotated BPMN specifications of candidate component services, that we use to check if there is an implemented service in the repository which satisfies each specification (by applying the selection procedure presented in Section 5). If the Semantic Model Checking fails for some specifications, the algorithm proposed in Figure 3 is applied.

The algorithm implements two different strategies: *decomposition* and *aggregation*. Given a candidate component service $\mathcal{S}$ that has no implementations in the repository, according to the decomposition strategy, the selection procedure is applied to $\mathcal{S}$ children (Children($\mathcal{S}$)) in order to find them. On the contrary, according to the aggregation strategy, the selection procedure is applied

Algorithm

```
input  : $\Delta = \Sigma - \Phi$ /* Set of not found services */
output: $\Phi'$ /* Set of found services */

(1) { $\Phi' := \Phi$;
(2)  while $\Delta \neq \emptyset$ do
(3)  { $\Upsilon := \emptyset$;    /* Set of services to be found */
(4)     for each $S \in \Delta$
(5)     { $\Psi(S) := \emptyset$;/* Set of siblings of service $S$ */
(6)        $\Omega(S) := \emptyset$;  /* Set of not found siblings of service $S$ */
(7)        $\Psi(S) :=$ SIBLING($S$);
(8)        if $(\Psi(S) \neq \emptyset) \wedge (\gamma_p > \gamma_c)$ do
(9)        { for each $\psi \in \Psi(S)$
(10)          if $\psi \in \Delta$ do $\Omega(S) := \Omega(S) \bigcup \{\psi\}$;
(11)          if ( $\Omega(S) \equiv \Psi(S)$ ) $\wedge$ ( $\Omega(S) \subseteq \Delta$ ) do
(12)          { $\Upsilon := \Upsilon \bigcup$ PARENT($S$);
(13)             $\Delta := \Delta - \Omega(S)$ }
(14)       } else if $\neg$ LEAF($S$) $\wedge$ $(\gamma_p < \gamma_c)$ do
(15)       { $\Upsilon := \Upsilon \bigcup$ CHILDREN($S$); $\Delta := \Delta - \{S\}$ }
(16)    } SELECTION($\Upsilon$); { $\Phi' := \Phi' \bigcup \Phi$; $\Delta := \Upsilon - \Phi$ }
(17) } return $\Phi'$;
(18) }
```

Fig. 3. The algorithm.

to the parent node of $\mathcal{S}$ (Parent($\mathcal{S}$)). The choice among decomposition and aggregation strategies is made taking into account the weights on the Weighted Decomposition Tree. Let be $\Psi(S)$ the set of siblings of $\mathcal{S}$, among which $\Omega(S)$ those siblings that have not been found (rows 5-7). The decomposition strategy is applied if the variation of ratio associated to the split of $\mathcal{S}$ (γ_c) is smaller than the variation of the ratio obtained by aggregating $\mathcal{S}$ and its siblings (γ_p) and $\mathcal{S}$ is not a leaf node (rows 8-13). In all the other cases, if γ_p is smaller than γ_c or $\mathcal{S}$ is a leaf node, we apply the aggregation strategy (rows 14-16). In our scenario the service selection phase performed with Σ has found four services $\Phi = \{\overline{\mathcal{S}_{01}}, \overline{\mathcal{S}_2}, \overline{\mathcal{S}_3}, \overline{\mathcal{S}_{89}}\}$ and has not found any service that accurately meet with processes $\mathcal{S}_{45}$ and $\mathcal{S}_{67}$. We apply the algorithm and, according to aggregation strategy, the output is $\Phi' = \{\overline{\mathcal{S}_{01}}, \overline{\mathcal{S}_2}, \overline{\mathcal{S}_3}, \overline{\mathcal{S}_{4567}}, \overline{\mathcal{S}_{89}}\}$. The algorithm stops if all the candidate component services have been found or the whole Weighted Decomposition Tree has been inspected.

7 Conclusion

The methodology presented in this paper costitutes an on-going work on the interleaved application of service identification and service selection phases. The result is a business process decomposition that takes into account both given guidelines for service design, such as high cohesion and low coupling and concrete services actually available and already implemented, among which candidate services must be selected. Current experimentation on a repository of Web services and business process specifications taken from real case scenarios is under development to demonstrate that the proposed approach ensures a coverage of the business process through available services higher than the one obtained by applying the service identification phase only. On the other hand, the decomposition tree and the algorithm shown in Fig.3 enable less applications of the service selection phase, thus ensuring better performances with respect to the ones discussed in [9].

References

1. J. Amsden. Modeling SOA: Part 1. Service identification. Technical report, IBM, `http://www.ibm.com/developerworks/rational/library/07/-1002amsden/`, 2007.
2. D. Bianchini, C. Cappiello, V. De Antonellis, and B. Pernici. P2S: a methodology to enable inter-organizational process design through Web Services. In *Proc. of the 29th International Conference on Advanced Information Systems*, pages 334–348, Amsterdam, The Netherlands, 2009.
3. Soon Ae Chun, Vijayalakshmi Atluri, Nabil, and R. Adam. Using semantics for policy-based web service composition. *Distributed and Parallel Databases*, 18:37–64, 2005.
4. I. Di Pietro, F. Pagliarecci, and L. Spalazzi. SWSAL: Semantic Web Service Annotation Language. no. 2008004453, SIAE Sezione Opere Inedite, Roma, 2008.
5. S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, K. Holley, and A. Arsanjani. SOMA: A method for developing service-oriented solutions. *IBM Systems Journal*, 47:377–396, 2008.
6. R.S. Kaabi, C. Souveyet, and C. Rolland. Eliciting service composition in a goal driven manner. In *Proc. of the 2nd Int. Conference on Service Oriented Computing (ICSOC'04)*, pages 308–315, New York, NY, USA, 2004.
7. Kyriakos Kritikos and Dimitris Plexousakis. Semantic qos metric matching. In *ECOWS'06*, pages 265–274, 2006.
8. E. Michael Maximilien and Munindar P. Singh. Toward autonomic web services trust and selection. pages 212–221. ACM Press, 2004.
9. Ivan Di Pietro, Francesco Pagliarecci, and Luca Spalazzi. Model checking semantically annotated services. *IEEE Transactions on Software Engineering*, 2011.
10. Q.Z. Sheng, B. Benatallah, Z. Maamar, M. Dumas, and A.H.H.Ngu. Enabling Personalized Composition and Adaptive Provisioning of Web Services. In *(CAiSE 2004)*, pages 322–337, 2004.
11. H.M. Shirazi, N. Fareghzadeh, and A. Seyyedi. A Combinational Approach to Service Identification in SOA. *Journal of Applied Sciences*, 5(10):1390–1397, 2009.
12. Qi Yu and Athman Bouguettaya. Framework for web service query algebra and optimization. *ACM Trans. Web*, 2:6:1–6:35, March 2008.
13. Tao Yu and Kwei-Jay Lin. Service selection algorithms for composing complex services with multiple qos constraints. In *ICSOC*, pages 130–143, 2005.

Preserving Unlikability against Covert Channels in Multi-Party Security Protocols (Extended Abstract)

Mikaël Ates[1], Francesco Buccafurri[2], Jacques Fayolle[3], and Gianluca Lax[2]

[1] Entr'ouvert, 169 rue du Château, 75014 Paris, France mates@entrouvert.com
[2] DIMET, University of Reggio Calabria, via Graziella, Feo di Vito, 89122 Reggio Calabria, Italy bucca@unirc.it, lax@unirc.it
[3] DIOM, Université de Lyon, 23 rue Michelon, 42023 Saint-Etienne, France jacques.fayolle@univ-st-etienne.fr

Abstract. Unlinkability is a privacy feature supported by those multi-party security protocols allowing anonymous users' credential exchanges among different organizations. Proper signature schemes, based on discrete logarithms, must be used in order to guarantee the above requirements as well as selective disclosure of information. In this paper, we highlight that whenever a concrete architecture based on the above protocols is implemented, some aspects concerning how to manage the association between bases of discrete logarithms and attributes used in attribute certificates should be carefully considered, in order to guarantee that unlinkability really holds. We show that the problem is concrete by testing that the state-of-the-art implementation suffers from the above problem.

1 Introduction

A credential is a powerful means to establish an identity, a role, or an attribute, in order to control accesses to digital services over the Web. There exist many application contexts in which users can be required to show a credential to access a service. For example, a car renting company could deploy an on-line registration process where users are asked to prove the possession of the driving license. However, there does not exist a standard credential exchange architecture, though the main stakes and issues are now well-known [4]. One of the known issues is that credentials exchanges can represent a threat on users' privacy. Indeed, as stated by [15], any information that distinguishes one person from another can be used for re-identifying data. For instance, considering the use of the driving license document, this one would induce the user to reveal unnecessary personal information (name, date of birth, and so on). Thus, in order to protect user's privacy in digital information exchange processes, the credential exchange architecture must permit users to perform a *selective disclosure* of their attributes in credentials, like the proof that an attribute value lies in a given interval (for example, a user could prove to be of age without revealing the date of birth).

Unfortunately, selective disclosure is not enough. Another important threat on privacy arises from the fact that two organizations could exchange the credentials shown by users in order to infer more information about the users. This is related to the issue of *linkability* of the user's transaction records hosted by multiple organizations [18]. Due to the grow of attention toward privacy concerns, nowadays *multi-party security* protocols often deal with the above issue. It is worth noting that the unlinkability property should be a feature aimed to defend the individual from the privacy threats coming from every party, including those whose position, role and dimension give them a strong control power on users, and, at the same time, a seeming trustworthiness. As a consequence, we cannot exclude in general that even authoritative entities (in principle, also a government organization, for example) could be malicious as far as the privacy issue is concerned.

In the literature, there are many results contributing to address the issue of linkability of user transactions. This topic was introduced by Chaum in [12, 10]. He also presented a signature scheme based on a blind signature allowing unlinkable certificate issuance and presentation [9, 11]. The certified data revealed as a factor of linkability is also widely studied. Revealing information preserving anonymity is usually unified under the wider topic of k-anonymity [13]. Two very relevant signature schemes proposed in the literature to guarantee that the signature of attribute certificates is not a factor of linkability are the *CL-Signature signature schema* [8] and the *Secret Key Certificate signature schema* [7]. Both are based on discrete logarithms for representing values of attributes in order to support also selective disclosure. The attribute certificates generated with the above schemes are called *anonymous credentials* and *private credentials*, respectively. Microsoft has recently implemented a multi-party security protocol called *U-Prove* [20, 5], based on the scheme [7]. In fact, it represents the state-of-the art implementation of this class of protocols. The design of privacy-preserving cryptographic protocols is a hard task [3]. Especially when dealing with anonymity since even cryptographic material can be the place for steganography [1]. Indeed, covert channels can be set-up with X509 certificates [19] and also within anonymity systems [14]. A credential exchange architecture is thus particularly exposed.

In this paper, by extending a result originally presented in [2], we highlight that the implementation of multi-party security protocols based on discrete logarithms for representing values of attributes should take care of some aspects concerning the association between bases of discrete logarithms and attributes used in attribute certificates. The above association is included into *certificate metadata*. We show that, if the above issue is not correctly handled, then the resulting system would allow adversarial organizations to break the unlinkability of user transactions by establishing a covert channel based on certificate metadata. In particular, it is possible to implement an attack which exploits the possibility of choosing the associations between bases of logarithms and the attributes they represent. The issuer could assign to a user, or to a set of users (for example, female users), specific base/attribute associations, in such a way that a colluding verifier could infer more information from the transactions. The relevance of the problem highlighted in the paper is confirmed by the fact that the U-Prove integration into the Identity Metasystem [16] suffers from the above problem. Concerning this issue, we highlight that the problem does not regard the cryptographic protocol itself [6]. Before submitting the paper, we contacted the Microsoft's U-Prove Team in order to inform them about our results. We received a feedback both interesting and encouraging. Indeed, they agree that the problem we have identified in the paper really exists in the current Community Technology Preview release of U-Prove. They are experimenting different ways to prevent this issue in the release version. Another contribution of the paper is to present a practical solution to the above issue based on public certificate metadata retrieved anonymously by the users. Importantly, this solution can be applied to the system U-Prove basically preserving its architecture.

The rest of the paper is organized as follows.

In Section 2, we show how the certificate metadata can be used in order to implement a covert channel breaking unlinkability. Then, we apply this attack on the U-Prove Technology integrated into the Identity Metasystem V1.0. In Section 3, we present a solution to handle certificate metadata for unlinkable certificates. Finally, in Section 4, we draw our conclusions.

2 The Attack: Certificate Metadata as Covert Channel

In this section, we show that if no proper strategy is adopted in the management of certificate metadata, an attack is possible allowing the issuer and the colluding verifier to infer more information about the prover than that actually disclosed by the prover. This attack is based on the possibility of creating a covert channel between the issuer and the verifier exploiting the certificate metadata exchanged between the issuer and the prover and between the prover and the verifier. A covert channel is a communication channel allowing a process to transfer information in a manner that violates the system security policy [21]. We can adapt this definition to our case saying that certificate metadata of an unlinkable attribute certificate are a covert channel if they allow the issuer to transfer to the verifier information about the prover.

In order to be more concrete, consider the following example of covert channel. The malicious issuer can give the provers two different types of certificate metadata, namely t_1 and t_2. The issuer and the colluding verifier agree that the type t_1 will be given only to a particular user u (or to a class of users, for example, male users). For the other users, the type t_2 will be exploited. It is clear that whenever the verifier will receive an attribute certificate whose metadata are of the type t_1, it can deduce that the prover is u (or a male, if the association t_1/male has been adopted) without the prover can detect such an information leak. Conversely, if the metadata type shown by the prover is t_2, the verifier deduces that the prover is not u (or, that the prover is a female). As a consequence, the certificate metadata are a covert channel since they reveal information about the prover, without the prover knowing. The question we have to address now is: Is it possible to use certificate metadata as a covert channel?

Observe that it is not possible to have metadata that differ for the names (e.g., exploiting the case sensitiveness we could use ”Surname” and ”surname” for the same attribute) or to use the numeric values of the bases for that purpose. Indeed, the name space has to be case sensitive and common, and the values of the bases are included in the public key of the issuer so that they are fixed and used pervasively. The possibility we have found to create different types of certificate metadata is exploiting the association between the attributes and the bases used to represent them. This obviously has to be allowed by the implementation of the protocol. Clearly, even though the protocol permits this, an honest issuer should not provide provers with different certificate metadata. Conversely, the malicious issuer and the colluding verifier can set up a covert channel against users which is based on the fact that the issuer gives specific metadata for each user. Said a the number of attributes, the issuer can create $a!$ different associations between the attributes and the bases, so that it can identify a subset of users with cardinality $a! - 1$, leaving one association to deal with the rest of the population.

2.1 Proof of Concept

In order to demonstrate that the warning detailed in the previous section is concretely relevant, we show that also the U-Prove integration into the Identity Metasystem, specified in [16], is not immune from the attack. The specification has been implemented under the U-Prove Community Technology Preview (CTP) name. These implementations are presented in [17]. The implementation of U-Prove is based on the extensions of Active Directory Federation Services 2.0, Windows Identity Foundation and CardSpace 2.0. The test bench is composed of:

- Two certificate issuers, called *Token Issuers*, one based on Active Directory Federation Services 2.0 CTP and one based on Windows Identity Foundation CTP, hosted by a Windows Server Enterprise 2008 SP2 station.
- A verifier, called *Relying Party*, based on Windows Identity Foundation CTP, hosted by a Windows Server Enterprise 2008 SP2 station.
- Two user environments, one with a Web browser Internet Explorer 7.0 hosted by a Windows Vista SP1 station and one with a Web browser Internet Explorer 8.0 hosted by a Windows Server Enterprise 2008 SP2 station, and both provided with an application for token management, called *Identity Selector*, which is CardSpace 2.0 CTP.

According to the specifications of [16], certificate metadata are called here *issuer parameters*. The issuer parameters contain the association between bases and attributes. The specifications indicate that during the first round of a token issuance, if the identity selector does not give an identifier of up-to-date issuer parameters, then the issuer provides a response containing the issuer parameters. This response is preceded by a user request containing the user credentials to authenticate to the issuer. The issuer parameters are thus here given when the user is authenticated. Moreover, there is no other means specified in [16] to provide the identity selector with the issuer parameters.

In our test, the user retrieves from the issuer an *Information Card.* Such a document indicates to the identity selector the authentication mechanisms required, applicative endpoints for issuance, and the attributes (called *Claims*) the user can obtain. This document does not contain the issuer parameters. The user adds this document to the identity selector. By a user action on the relying party, the latter triggers a token request. The user Web browser forwards this request to the identity selector. The identity selector asks the user which issuer to request and the authentication credentials. We have observed with both the issuers deployed that whenever the identity selector performs the first token request, without indicating an identifier of up-to-date issuer parameters, the issuer replies with a message response containing the tokens and the issuer parameters.

Then, the tests confirm that the configuration options offered by CardSpace 2.0 CTP do not permit to add issuer parameters. As a consequence, before the first token request, there is no way for the user to obtain the issuer parameters. The user is then provided with the issuer parameters at the token issuance and these issuer parameters are the ones used by the identity selector to verify the token validity and to lead the proofs.

Finally, we have tested the system with multiple issuer parameters and verified that correct orders for associations between discrete logarithm bases and attribute types are required for successful user proofs.

We conclude that this method is the unique way to provide users with issuer parameters in the U-Prove implementation. Since the issuer parameters are given to the user when the user is authenticated and the user has no means to check the uniqueness of the issuer parameters in all the realm, the association of bases and attributes can be user-specific and the attack can be operated.

3 The Solution

Recall that the covert channel underlying the attack is implemented by changing the association between bases and attributes in the attribute certificates. As a consequence, the first intuitive solution that one could identify is to have a third trusted party (TTP) publishing the *legal* association which is included in the trust chain, ensuring that the association is unique for all the users of a given issuer. For instance, the user, before the certificate proving, might download the certificate metadata for the considered issuer from TTP in order to check whether the issuer is adopting the legal association. This solution clearly works, but it is in practice little feasible, since it results in a strictly hierarchical architecture strongly limiting the pervasiveness of the system (for example, think of the management of join and leave of issuers). This is in fact coherent with the choice done in the U-Prove architecture, which does not adopt any rigid hierarchy on top of the issuers. Moreover, our attack is based on the assumption that even authoritative entities (in principle, also a government organization, for example) could be malicious as far as the privacy issue is concerned. In fact the unlinkability property should be a feature aimed to defend the individual from the privacy threats coming from every party, including those whose position and dimension give them a strong control power on users, and, at the same time, a seeming trustworthiness. Under this assumption, it is difficult to identify in the real case which entity could play the role of TTP.

Due to the above considerations, we have excluded the first intuitive solution. On the contrary, we propose a solution preserving the architecture of U-Prove and relying only on the autonomous ability of the user to check the trustworthiness of the issuer.

The solution, which we call *two-phase-issuance (2PI)*, consists in dividing the issuance step into two distinct phases:

1. The user retrieves from the issuer the certificate metadata anonymously.
2. The user retrieves from the issuer the signature values on a set of attribute values, without revealing any information about certificate metadata previously obtained.

The solution expects that time-correlation attacks are not applicable, but this is typical in the context of unlinkability. Indeed, if this is not the case, the time correlation between issuance and proving steps would reveal the user identity.

Observe that, the above protocol can be obtained by using the features of U-Prove, as we describe in Section **??**. We next show that the above solution works, in the sense that the probability that unlinkability is broken by means of a malicious behavior of issuers and verifiers is the same as it happens whenever the two parties guess (with no a-priori knowledge) this linking information. We note that the unlinkability is broken if a pair issuer-verifier is able to distinguish a subset of users.

Consider a pair issuer-verifier, say I and V. Let denote by U the set of all users. Let u be a subset of users characterized by some values (or ranges) of the attributes. Obviously, if V tries to distinguish the subset u just by guessing, the success probability is $\frac{|u|}{|U|}$.

Consider now the case that I and V agree on a particular association bases-attributes in order to identify u. For example, they want to distinguish male and female users. The issuer generates two associations, say A_M and A_F, and the expected goal is to assign A_M to male users and A_F to female users. Thanks to 2PI, there is no way to deterministically know if the user retrieving anonymously certificate metadata (i.e., issuer parameter in U-Prove terminology) is male or female. As a consequence, the only possibility is to guess this feature. Clearly, if the guessing succeeds, then the user will not be able to detect the malicious behavior since the certificate metadata obtained in the second phase of the protocol 2PI coincides to those obtained before. In the general case, the only possibility for I to implement a cover channel allowing V to link the subset

u is to guess that the user requiring certificate metadata in phase 1 of 2PI is belonging to u. The attack succeeds only if this happens, thus with probability $\frac{|u|}{|U|}$. The solution thus fully preserves unlinkability, since there is no higher probability for V to infer linking information thanks to the colluding issuer I w.r.t. the case that V cannot rely on the cooperation of I.

4 Conclusion

In this paper, we have highlighted a possible risk of vulnerability arising from the implementation of multi-party security protocols based on discrete logarithms for representing attributes. In particular, we have shown that if the issuer is free to manage maliciously the association between bases of discrete logarithms and attributes used in attribute certificates, then a covert-channel-based attack is possible allowing colluding issuers and verifiers to break unlinkability enforced by the protocol. We have identified the problem by defining how the covert channel can be implemented and checked that this problem is not only an abstract hypothesis, but a concrete issue. We have reached this conclusion by checking that the most important existing system aimed to provide unlinkable multi-party credential exchange, which is U-Prove, allows malicious organizations to implement the above covert channel, thus potentially breaking unlinkability. The paper addresses also the issue of the prevention of the above risk, by proposing a solution easily applicable also to the concrete architecture of U-Prove. Even though the paper includes some implementation issues which we have applied to the case of U-Prove in order to incorporate in it our solution, it could be interesting to implement a complete system prototype extending U-Prove in the direction we have identified. This is a matter of our future work.

Acknowledgments

We are very grateful to Microsoft Corporation for the interest shown in our work. This work was partially funded by the French Ministry of Economy and Industry and by the Italian Ministry of Research through the PRIN Project EASE (Entity Aware Search Engines).

References

1. L. V. Ahn. Public-key steganography. In *In: Advances in Cryptology Proceedings of Eurocrypt 04*, pages 323–341. Springer-Verlag, 2004.
2. M. Ates. *Digital Identities : User Centric and Privacy-Respectful Cross-Organizational Identity Management*. PhD thesis, Université de Lyon - SATIN Team DIOM Laboratory Telecom Saint-Etienne University of Saint-Etienne, 2009.
3. T. Balopoulos, S. Gritzalis, and S. Katsikas. Specifying and implementing privacy-preserving cryptographic protocols. *International Journal of Information Security*, 7:395–420, 2008. 10.1007/s10207-008-0057-y.

4. A. Bhargav-Spantzel, J. Camenisch, T. Gross, and D. Sommer. User centricity: a taxonomy and open issues. In *DIM '06: Proceedings of the second ACM workshop on Digital identity management*, pages 1–10, New York, NY, USA, 2006. ACM.
5. S. Brands. U-prove technology overview v1.0. Technical report, 2010.
6. S. Brands and C. Paquin. U-prove cryptographic specification v1.0. Technical report, 2010.
7. S. A. Brands. *Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy.* MIT Press, Cambridge, MA, USA, 2000.
8. J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *International Conference on Security in Communication Networks - Lecture Notes in Computer Science*, volume 2576, pages 268–289, 2002.
9. D. Chaum. Blind signatures for untraceable payments. In *International Cryptology Conference on Advances in Cryptology*, pages 199–203, 1983.
10. D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.
11. D. Chaum, A. Fiat, and M. Naor. Untraceable electronic cash. In *International Cryptology Conference on Advances in Cryptology*, pages 319–327, London, UK, 1990. Springer-Verlag.
12. D. L. Chaum. Untraceable electronic mail, return addresses, and digital pseudonyms. *Commun. ACM*, 24(2):84–90, 1981.
13. V. Ciriani, S. De Capitani di Vimercati, S. Foresti, and P. Samarati. Theory of privacy and anonymity. In M. Atallah and M. Blanton, editors, *Algorithms and Theory of Computation Handbook (2nd edition).* CRC Press, 2009.
14. S. J. Murdoch. Covert channel vulnerabilities in anonymity systems. Technical report, 2007.
15. A. Narayanan and V. Shmatikov. Myths and fallacies of personally identifiable information. *Commun. ACM*, 53(6):24–26, 2010.
16. C. Paquin. U-prove technology integration into the identity metasystem v1.0. Technical report, 2010.
17. C. Paquin and G. Thompson. U-prove ctp white paper. Technical report, 2010.
18. A. Pfitzmann and M. Kohntopp. Anonymity, unobservability, and pseudonymity - a proposal for terminology. In *Lecture Notes in Computer Science: Designing Privacy Enhancing Technologies*, volume 2009, pages 1–9. Springer Berlin / Heidelberg, 2001.
19. C. Scott. Network covert channels: Review of current state and analysis of viability of the use of x.509 certificates for covert communications. Technical report, 2008.
20. U-Prove. Microsoft Corporation Technology, 2010. `https://connect.microsoft.com/content/content.aspx?contentid=12505\&siteid=642` [Online; accessed 1-September-2010].
21. US-DoD. *Trusted Computer System Evaluation. U.S. Department of Defense. The Orange Book.* Publication DoD 5200.28-STD, 1984.

MediaPresenter, a web platform for multimedia content management[⋆]

Sonia Bergamaschi[1], Fabio Ferrari[2], Matteo Interlandi[1], and Maurizio Vincini[1]

[1] DII, University of Modena and Reggio Emilia
via Vignolese 905, 41015 Modena, Italy
firstname.lastname@unimore.it
Addiction Creation Media Lab, Via Del Quaresimo 4,
[2] 42100 Codemondo (RE), Italy,
firstname@addiction.it

Abstract. The composition of multimedia presentations is a time and resource consuming task if not afforded in a well defined manner. This is particularly true for medium/big companies, where people having different roles and following different high-level directives, collaborate in the authoring and assembling of a final product. In this paper we present MediaPresenter, the framework we developed to support companies in the creation of multimedia communication means, providing an instrument that users can exploit every time new communication channels have to be created.

Keywords: digital asset, multimedia, presentation, tag cloud

1 Introduction

Nowadays companies use different software products for authoring and presenting their multimedia presentations. We define a multimedia presentation as a composition of textual information and digital assets such as images, photos, video and audio. In an enterprise context, applications such as Microsoft Power Point or OpenOffice Impress have the advantage to be competitive from the point of view of costs, but limits arise when the presentation creation process becomes intensive and hence it requires agile dynamics and company-specific logics.

Usually medium/big companies count on communication means developed **(1)** in **outsourcing** (web sites, multimedia presentations, paper communications), or **(2) internally** developed and managed, with poor efficiency compared with the potentiality provided by the used software tools.

These modi operandi have emphasize a series of issues. In the first case **(1)** the company must be always followed by another external and specialized enterprise which will cover every communication demand. This scenario comport an

[⋆] This work was partially supported by Lisea Laboratory (Italy, Emilia-Romagna Region, PRRRIITT Misura 3.4) economic investment which, due to budget constraints, is often not feasible if new communication products have to be created constantly. The major issue comes when the workflow must be managed from inside to outside the company and vice-versa. The demand to communicate its own services, products and brand appeal, is not compatible with the outsourcing model, since, for every new activity, the company needs to invest time and resources in producing and approving shared materials and therefore efforts are subtracted to activities which create more added-value for the company.

In the second case **(2)** all the software packets do not contemplate the possibility to natively share among users the collective multimedia assets such as the enterprise logo, background images, internal photos, etc.. Moreover, since these collective assets are local and personal, they have to be updated continuously. This update process can be very resource consuming if the number of devices and people is high.

Based on the extensive analysis of issues an enterprise can face during the creation of multimedia presentation, we adopted an extension of the SATP approach [1]. This approach divides the process of creating a multimedia presentation into four sequential phases: Select, Assemble, Transform, Present).

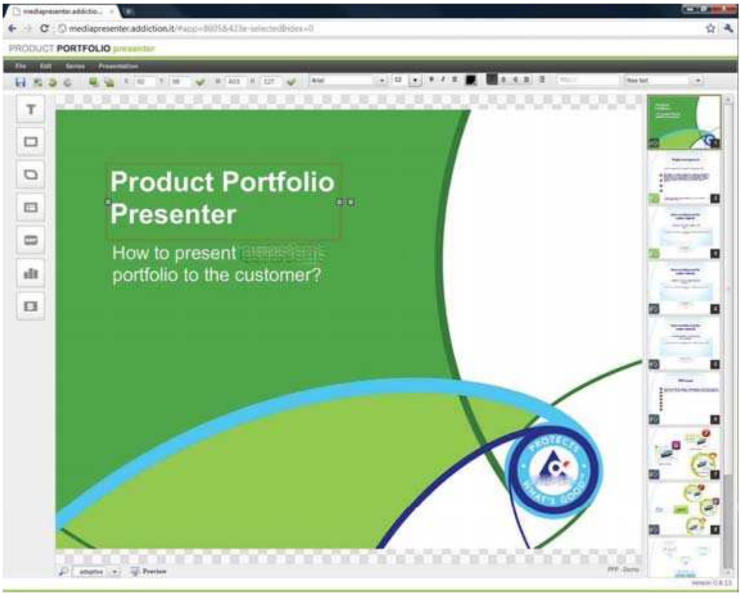
In this paper we introduce **MediaPresenter**, the framework we developed to support companies in the creation of multimedia communication means, providing an online and centralized instrument (MediaBank) that users can exploit every time new communication channels have to be created. In this way the company can have a communication mean homogeneous and always on line with the marketing directives.

The paper is structured as follow: Section 2 is an overview of MediaPresenter and the Assemble, Transform, Present approach we followed for its design. Section 3 outlines the architecture we developed, meanwhile Section 4 describes the Data Cloud and the algorithm we use for the ranking of the entities and related tags. The paper is closed by a short Conclusion.

2 MediaPresenter

MediaPresenter is developed by a join collaboration between the DBGroup at University of Modena and Reggio Emilia (www.dbgroup.unimo.it) and Addiction Creation Media Lab (www.addiction.it), an Italian SME Media Agency. Part of the activity is founded by Italian Emilia Romagna region, within the LISEA lab.

MediaPresenter is an on-line cross-platform application, perfect incarnation of the Software-as-a-Service (SaaS) paradigm for business enterprise. It offers a large number of services for sharing and managing digital archives. Such services include the concurrent access to data by multiple users with different roles, import of multiple type of digital assets (3D object, video, images, etc), export in various format (swf, pptx, png and pdf), and a keyword search engine for retrieving contents that uses a tag cloud based approach to propose to the user related digital content. Figure 1 shows a snapshot taken during the process of series creation.



Product Portfolio Presenter

How to present [illegible] portfolio to the customer?

Fig. 1. MediaPresenter slide creation

MediaPresenter has been designed as a collaborative framework, thus groups of users can be created and policies can be set both on groups and single users. We chose this approach because it allows to model in MediaPresenter the internal hierarchy of an enterprise, so different people with different duties may contribute to a presentation in different manners.

In MediaPresenter the authoring of new communication media follows the Select, Assemble, Transform and Present creation chain approach whose schema is depicted in Figure 2. The original schema has been maintained, but in our approach we developed a different semantics for each phase, specifically tailored for our purpose. In the followings we briefly describe each phase.

2.1 Select

In this phase, elements contained in the MediaBank are retrieved using the keyword search and the tag cloud. We define as element, a digital assets, slides, series of slides or already composed presentations. Starting from the assumption that the simple file name is just an ID and it does not describe the element content(s), each element is inserted in the repository and has associated a series of tag which specify contents or properties that characterize such element.

In this way, elements can be retrieved using a keyword-base search engine and the results visualized as tag cloud. The user is thus able to browse among the various tags and retrieve the best suitable element. Policies are associated

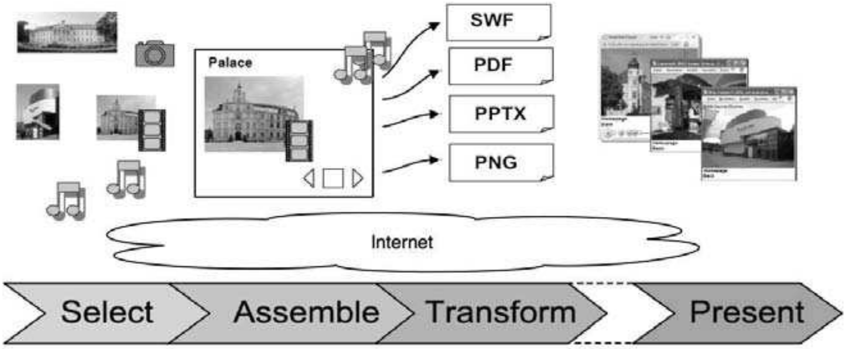


Fig. 2. SATP raffiguration

to each user, based on the group the user belongs to. Search results, hence, may depend on the policy settings for the user.

2.2 Assemble

In this phase, the user assembles the final or semifinal product (for example a single slide, a series of slide or a presentation) starting from the single elements already retrieved in the previous phase.

Users also assemble different types of elements depending on their role, for example one user can be a slide-maker and therefore he/she can assemble just slides, meanwhile a presentation-maker can assemble complete presentations.

2.3 Transform

Once the user has repeated iteratively the previous phases and therefore the complete presentation has been created, the transform phase permits to save the final product in different formats following the user needs.

Thanks to its ability to manage different formats without rely on a proprietary language, MediaPresenter is also able to integrate and make available to the user presentations not initially designed with this framework. If, for example, some legacy products, made with a third part software application, have to be integrated with other presentations designed in other formats, MediaPresenter seamlessly permits it.

2.4 Present

MediaPresenter client is a Flex[3] application running on browser, hence a presentation can be potentially be shown on each device having a browser and an

[3] http://www.adobe.com/products/flex/

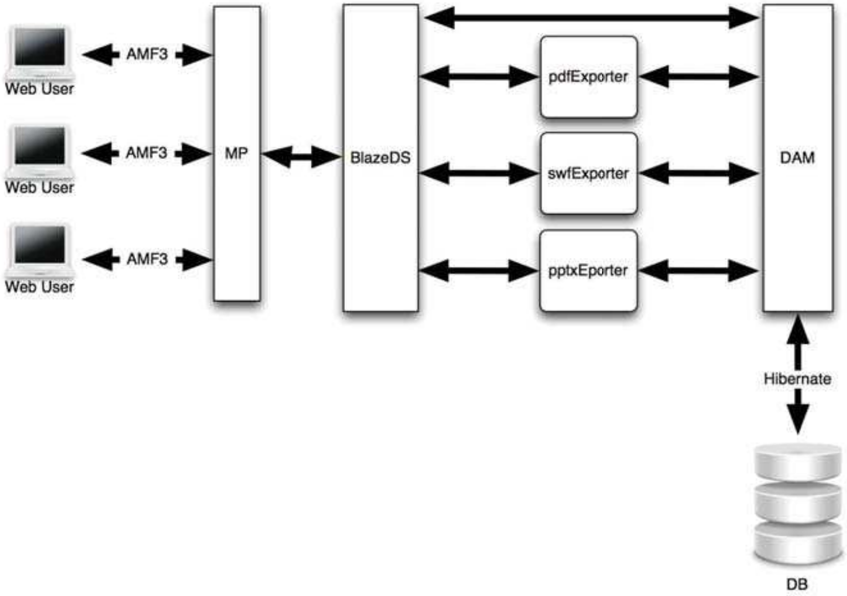


Fig. 3. Mediapresenter architecture

internet connection. But since a presentation can be transformed in different formats, an user may also employ third part software products.

3 Architecture

MediaPresenter is a 3-tier web system for contents managing and its architecture is shown in Figure 3.

The information that the storage layer (MediaBank) memorizes can be classified in two sets: **presentations** and **digital assets**. For digital assets are considered all the digital elements such as images, photos, videos, audios animations, etc.. which represent the company's multimedia knowledge. Since a medium/big enterprise can easily reach the number of 30 - 40.000 digital assets, users can associate significative keywords to each single asset. Digital assets can be easily accessed through a generic API and, since MediaPresenter has been designed to be completely independent to a specific repository, they could be stored directly in the MediaPresenter database or in specialized facilities called Digital Assets Management (DAM) such as Celum[4] and Xinet[5].

Presentations represent the ”final products” created by the user with MediaPresenter and they contain combinations of different digital assets together

[4] http://www.celum.com

[5] htpp://www.xinet.com

with textual contents. To facilitate the modularity and reuse, the user can create a pieces of presentation, called series, that can be reused to compose multiple presentations. Similarly to digital assets, presentations can be tagged to increase the search accuracy.

Each presentation is saved in an open XML-compliant format and can be exported in different formats (SWF, pdf, png and pptx so far).

On the client side the user develops his/her presentation using the MediaPresenter web application completely developed in Adobe Flex. Since all the slides, series, presentations as well as digital assets are remotely stored, every time the user needs to save or load data, a remote call to a java method is fired. The remote call is serialized using the AMF3 binary format protocol and sent over http to the application server where BlazeDs[6] is able to capture, deserialize and manage the request.

4 Data Cloud

The main goal of MediaPresenter is to produce multimedia communication channel for enterprise internal or external use. To reach this goal, the system provides to the user all the available information which can be briefly summarized in presentations, unpacked slides or series of slides and digital assets already used in the past or available as digital content of the enterprise.

During the creation process, the user has to retrieve an already available presentation or a specific digital asset stored into the MediaBank. The typical way to access these data are searching, e.g. based on name, dimension, type, date of creation and so forth. The results are often unsatisfactory due to the lack of knowledge and experience of the user.

To overcome these issues, our system permits to relate significative words to each digital assets and series: we define these words as tags that can be directly specified by the user either choosing among a predefined set, or created exnovo at run-time. The action of organize resources by adding metadata is called "tagging" and it's gaining popularity on the web in this years [2]. In this way, users can perform a free keyword search over the MediaBank and the system retrieves the entities related to the tags correspondent to the keywords.

In addition, using tags to label resources, allows the system to create a set of tags starting from the results returned by the usual search by keyword. This set of tags, called **Tags Cloud**, considers each terms as an hyperlink that can be used to refine the search results, dynamically guiding users in the hidden relationship among contents and eventually leading to serendipitous discoveries of interesting results.

The objective is then to summarize keyword search results using **Data Cloud**, which presents the most significant words (tags) associated with the search results. The advantage of this approach is to highlight the most significant concepts and hidden complex relationship in the modeling context.

[6] htpp://www.opensource.adobe.com/wiki/display/blazeds/BlazeDS

4.1 Model for Data Cloud: ranking search entities and tags query

We consider a set $\mathcal{C}$ of object, i.e. digital assets, and the set T of all tags. These tags are textual labels (words) assigned to objects, thus each object $c \in \mathcal{C}$ is associated to a set of tags, denoted with T_c.

We assume, without loss of generality, that a keyword query q is only expressed in terms of tags $t \in T$. Given a query q, the set of result are denoted as $\mathcal{C}_q \subseteq \mathcal{C}$, that contains the set of objects related at least to a tag contained in the query.

We denote T_q the set of tags related to at least one object of $\mathcal{C}_q$, and for each tag $t \in T_q$ we denote $A_q(t)$ as the set of objects associated to the tag t.

We define two score functions, one for objects retrieved from the query and one for the tags $t \in T_q$. Both are based on IR-standard ranking methods [3], i.e., $tf * idf$ for any tag of the query.

The term frequency of a tag t in the query related to an object c can be computed as:

$$tf_{t,c} = \frac{|A_q(t)|}{\sum\limits_{t_i \in \tau_q} |A_q(t_i)|} \tag{1}$$

While the inverse document frequency idf for t is:

$$idf_t = \ln\left(\frac{|\mathcal{C}|}{|A_q(t)|}\right) \tag{2}$$

Then, we consider the $tf * idf$ for each tag in the query and we define the score for an object c w.r.t. a query q as [4]:

$$score(c, q) = \sum_{t \in q} tf_{t,c} * idf_t \tag{3}$$

This function permits to MediaPresenter to show the retrieved digital assets in an ordered way. Analogously, we can define the score for a tag t w.r.t. a query q in the following way:

$$score(t, q) = \frac{|A_q(t)|}{|\mathcal{C}_q|} * \ln\left(\frac{|\mathcal{C}|}{|A_q(t)|}\right) \tag{4}$$

that permits to rank the tags in order to produce the suitable visual summary of a collection of texts by visually depicting the tag score by font size [5]

In [6] was proved that the IF-IDF based algorithm for score computation is the most suitable to maximize the metrics useful in Tag Cloud environment, i.e. Coverage, Overlap, Cohesiveness, Balance, Relevance and Popularity; therefore we are reasonably guarantee that our approach produce the best service for the MediaPresenter users.

In addition, since the user creates tags in a context of a group, and each group has a label identifying its generic topic, we can consider that given a certain search result, the sum of all the groups' labels is itself a Tag Cloud summarizing the whole topics of the results. In order to sum up the tag clouds resulting from the tags and the one resulting from the group labels, we use colors to identify for each tag the group it belongs to, and an index showing all the groups related to the search. In this way the user is able to perform a refinement over the results using two different levels of granularity: by generic topic, selecting the group of interest on the index, and one more detailed using the tags from the cloud.

5 Conclusion

In medium/big companies the creation of multimedia presentations can be a heigh resource consuming task, especially if it needs collaboration among people with different roles. In this paper we described the industrial requirements we tried to fulfill and the approach we followed in the development of a framework that enables users to collaborate in the creation of multimedia presentations. We put particular focus on the design of the keyword search engine and the tag cloud.

References

1. Ansgar Scherp, "Canonical processes for creating personalized semantically rich multimedia presentations," *Multimedia Systems*, vol. 14, pp. 415–425, 2008, 10.1007/s00530-008-0139-8.
2. Scott A. Golder and Bernardo A. Huberman, "The structure of collaborative tagging systems," *CoRR*, vol. abs/cs/0508082, 2005.
3. Vagelis Hristidis, Luis Gravano, and Yannis Papakonstantinou, "Efficient ir-style keyword search over relational databases," in *Proceedings of the 29th international conference on Very large data bases - Volume 29*. 2003, VLDB '2003, pp. 850–861, VLDB Endowment.
4. Georgia Koutrika, Zahra Mohammadi Zadeh, and Hector Garcia-Molina, "Data clouds: summarizing keyword search results over structured data," in *EDBT*, 2009, pp. 391–402.
5. Bongshin Lee, Nathalie Henry Riche, Amy K. Karlson, and M. Sheelagh T. Carpendale, "Sparkclouds: Visualizing trends in tag clouds," *IEEE Trans. Vis. Comput. Graph.*, vol. 16, no. 6, pp. 1182–1189, 2010.
6. Petros Venetis, Georgia Koutrika, and Hector Garcia-Molina, "On the selection of tags for tag clouds," in *Proceedings of the fourth ACM international conference on Web search and data mining*, New York, NY, USA, 2011, WSDM '11, pp. 835–844, ACM.

Author Index