Web Semantics: Science, Services and Agents on the World Wide Web 19 (2013) 42–58

Contents lists available at SciVerse ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Ontology evolution without tears

Haridimos Kondylakis *, Dimitris Plexousakis

Information Systems Laboratory, FORTH-ICS, N. Plastira 100, Vassilika Vouton, GR-700 13 Heraklion, Crete, Greece

ARTICLE INFO

Article history:
Received 29 September 2011
Received in revised form
7 November 2012
Accepted 13 January 2013
Available online 23 January 2013

Keywords:
Ontology evolution
Data integration
Query rewriting

ABSTRACT

The evolution of ontologies is an undisputed necessity in ontology-based data integration. Yet, few research efforts have focused on addressing the need to reflect the evolution of ontologies used as global schemata onto the underlying data integration systems. In most of these approaches, when ontologies change their relations with the data sources, i.e., the mappings, are recreated manually, a process which is known to be error-prone and time-consuming. In this paper, we provide a solution that allows query answering in data integration systems under evolving ontologies without mapping redefinition. This is achieved by rewriting queries among ontology versions and then forwarding them to the underlying data integration systems to be answered. To this purpose, initially, we automatically detect and describe the changes among ontology versions using a high level language of changes. Those changes are interpreted as sound global-as-view (GAV) mappings, and they are used in order to produce equivalent rewritings among ontology versions. Whenever equivalent rewritings cannot be produced we a) guide query redefinition or b) provide the best "over-approximations", i.e., the *minimally-containing* and *minimally-generalized* rewritings. We prove that our approach imposes only a small overhead over traditional query rewriting algorithms and it is modular and scalable. Finally, we show that it can greatly reduce human effort spent since continuous mapping redefinition is no longer necessary.

1. Introduction

The development of new scientific techniques and the emergence of new high throughput tools have led to a new information revolution. The nature and the amount of information now available open directions of research that were once in the realm of science fiction. During this information revolution the data gathering capabilities have greatly surpassed the data analysis techniques, making the task to fully analyze the data at the speed at which it is collected a challenge. The amount, diversity, and heterogeneity of that information have led to the adoption of data integration systems in order to manage it and further process it. However, the integration of these disparate data sources raises several semantic heterogeneity problems.

By accepting an ontology as a point of common reference, naming conflicts are eliminated and semantic conflicts are reduced. Ontologies are used to identify and resolve heterogeneity problems, usually at schema level, as a means for establishing an explicit formal vocabulary to share. During the past years, ontologies have been used as global schemata in database integration [1], obtaining promising results, for example in the fields of biomedicine and bioinformatics [2,3]. When using ontologies to integrate data, one is required to produce mappings, to link similar concepts or relationships from the ontology/ies to the sources by way of an equivalence. This is the *mapping definition process* [4] and the output of this task is the *mapping*, i.e., a collection of mappings rules. In practice, this process is done manually with the help of graphical user interfaces and it is a *time-consuming*, *labor-intensive* and *error-prone* activity [5].

Despite the great amount of work done in ontology-based data integration, an important problem that most of the systems tend to ignore is that ontologies are living artifacts and subject to change [4]. Due to the rapid development of research, ontologies are frequently changed to depict the new knowledge that is acquired. The problem that occurs is the following: when ontologies change, the mappings may become invalid and should somehow be updated or adapted.

In this paper, we address the problem of data integration for evolving RDF/S ontologies that are used as global schemata. We address the problem for a core subset of SPARQL queries that correspond to a union of conjunctive queries. We argue that ontology change should be considered when designing ontology-based data integration systems. A typical solution would be to regenerate the mappings and then regenerate the dependent artifacts each time the ontology evolves. However, as this evolution might happen too often, the overhead of redefining the mappings each time is significant. The approach, to recreate mappings from scratch each time the ontology evolves, is widely recognized to be problematic [5–7], and instead, previously captured information

* Corresponding author. Tel.: +30 2810 391499; fax: +30 2810 391428.
E-mail addresses: kondylak@ics.forth.gr, kondylak@gmail.com (H. Kondylakis).

1570-8268/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
doi:10.1016/j.websem.2013.01.001

should be reused. However, all current approaches that try to do that suffer from several drawbacks and are inefficient [8,9] in handling ontology evolution in a state of the art ontology-based data integration system. The lack of an ideal approach leads us to propose a new mechanism that builds on the latest theoretical advances on the areas of ontology change [10] and query rewriting [11,12] and incorporates and handles ontology evolution efficiently and effectively. More specifically:

- We present the architecture of a data integration system, named *Evolving Data Integration* system, that allows the evolution of the ontology used as global schema. *Query answering* in our system proceeds in two phases: (a) *query rewriting* from the latest to the earlier ontology versions and (b) query rewriting from one ontology version to the local schemata. Since query rewriting to the local schemata has been extensively studied [11–13], we focus on a layer above and deal only with the query rewriting between ontology versions.
- The query processing in the first step consists of: (i) *query expansion* that considers constraints coming from the ontology, and (ii) *valid query rewriting* that uses the changes between two ontology versions to produce rewritings among them.
- In order to identify the changes between the ontology versions we adopt a high-level language of changes. We show that the proposed language possesses salient properties such as *uniqueness*, *inversibility* and *composability*. *Uniqueness* is a pre-requisite for the solution described in this paper, where the other two properties are nice to have, but they are not necessary for our solution. The sequence of changes between the latest and the other ontology versions is produced automatically at setup time and then those changes are translated into logical GAV mappings. This translation enables query rewriting by unfolding. Moreover, the *inversibility* is exploited to rewrite queries from past ontology versions to the current, and vice versa, and *composability* to avoid the reconstruction of all sequences of changes among the latest and all previous ontology versions.
- Despite the fact that query rewriting always terminates, the rewritten queries issued, using past ontology versions, might fail. We show that this problem is not inhibiting in our algorithms but a consequence of information unavailability among ontology versions. To tackle this problem, we propose two solutions: (a) either to provide best "over-approximations" by means of *minimally-containing* and *minimally-generalized* queries, or (b) to provide insights for the failure by means of *affecting change operations*, thus driving query redefinition.
- We show that our method is sound and complete and does not impose a significant overhead. Finally, we present our experimental analysis using two real-world ontologies. Experiments performed show the feasibility of our approach and the considerable advantages gained.

Such a mechanism, that provides rewritings among data integration systems that use different ontology versions as global schemata, is *flexible*, *modular* and *scalable*. It can be used on top of any data integration system—independently of the family of the mappings that each specific data integration system uses to define mappings between one ontology version and the local schemata (GAV, LAV, GLAV [13]). New mappings or ontology versions can be easily and independently introduced without affecting other mappings or other ontology versions. Our engine takes the responsibility of assembling a coherent view of the world out of each specific setting.

This paper is an extended and revised version of a previously published conference paper [14] whereas the implemented system was demonstrated in [15]. However, only the basic ideas were described in [1], without a detailed analysis of the theoretical foundation of the approach. This manuscript adds to the previously published results, the related work, the formal properties of the language of changes used to capture ontology evolution and the specific semantics of the implemented architecture. In addition, the new algorithms that were created are presented, their correctness is proved and their complexity is analyzed. Finally, an evaluation of the system is presented for the first time using real and synthetic set of queries, and a discussion is added to the conclusion of this paper.

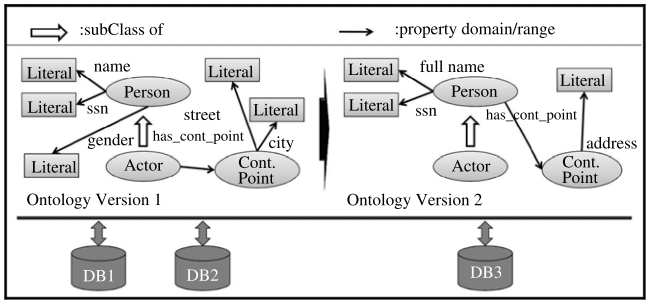

Fig. 1. The motivating example of an evolving ontology.

The rest of the paper is organized as follows: Section 2 introduces the problem by an example and presents related work. Section 3 presents the architecture of our system and describes its components. Section 4 describes the semantics of such a system and Section 5 elaborates on the aforementioned query rewriting among ontology versions. Finally, Section 6 presents our experimental analysis and Section 7 provides a summary and an outlook for further research.

2. Motivating example and related work

Consider the example RDF/S ontology shown on the left of Fig. 1. This ontology is used as a point of common reference, describing persons and their contact points ("*Cont.Point*"). We also have two relational databases DB1 and DB2 mapped to that version of the ontology. Assume now that the ontology designer decides to move the domain of the "*has_cont_point*" property from the class "*Actor*" to the class "*Person*", and to delete the property "*gender*". Moreover, the "street" and the "*city*" properties are merged to the "*address*" property. Merging is a concatenation with some special character like comma between the words. Furthermore, the "*name*" property is renamed to "*fullname*" as shown on the right of Fig. 1. Then, one new database DB3 is mapped to the new version of the ontology leading to two data integration systems that work independently. In such a setting we would like to issue queries formulated using any ontology version available. Moreover, we would like to retrieve answers from all underlying databases.

Several approaches have been proposed so far to tackle similar problems. For example, for XML databases there have been several approaches that try to preserve mapping information under changes [16] or propose guidelines for XML schema evolution in order to maintain the mapping information [17]. Moreover, augmented schemata were introduced in [18] to enable query answering over multiple schemata in a data warehouse, whereas other approaches change the underlying database systems to store versioning and temporal information such as [19–24]. However, our system differs from all the above in terms of both goals and techniques.

Other works focus on the problem of updating RDF/S [25–27] or OWL–DL [28] knowledge bases. These works mostly try to determine the effects and side-effects of elementary or complex change operations and to characterize the different class of updates with a well-defined semantics. In our work, however, we do not deal with the effects of the change operations on the ontology but