Artificial Intelligence 238 (2016) 11–62

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Knowledge base exchange: The case of OWL 2 QL

Marcelo Arenas [a], Elena Botoeva [b,*], Diego Calvanese [b], Vladislav Ryzhikov [b]

[a] Pontificia Universidad Católica de Chile, Chile
[b] Free University of Bozen–Bolzano, Italy

ARTICLE INFO

Article history:
Received 31 January 2015
Received in revised form 21 December 2015
Accepted 10 May 2016
Available online 16 May 2016

Keywords:
Description logic
Knowledge exchange
DL-Lite
Data exchange
Query inseparability

ABSTRACT

In this article, we define and study the problem of exchanging knowledge between a source and a target knowledge base (KB), connected through mappings. Differently from the traditional database exchange setting, which considers only the exchange of data, we are interested in exchanging implicit knowledge. As representation formalism we use Description Logics (DLs), thus assuming that the source and target KBs are given as a DL TBox+ABox, while the mappings have the form of DL TBox assertions. We define a general framework of KB exchange, and study the problem of translating the knowledge in the source KB according to the mappings expressed in OWL 2 QL, the profile of the standard Web Ontology Language OWL 2 based on the description logic *DL-Lite*$_{\mathcal{R}}$. We develop novel game- and automata-theoretic techniques, and we provide complexity results that range from NLogSpace to ExpTime.

1. Introduction

Ontologies are at the heart of various Computer Science disciplines, among which the most prominent ones are Semantic Web, Biomedical informatics, and of course, Artificial Intelligence and Knowledge Representation. Here, for simplicity, by *ontology* we mean a formal representation of the knowledge about a domain in terms of concepts (unary predicates) and roles (binary predicates). In the biomedical domain, e.g., *Pneumonia* and *Lung* could be concepts, and *finding_site* could be a role, and the knowledge about the domain could be asserted in an axiom of the form *"The finding site of pneumonia is lungs"* [1,2]. The advantages of using ontologies are that, on the one hand, they provide a framework for organizing and structuring information, and on the other hand, they are equipped with capabilities to reason about concepts and roles.

When representing the knowledge about a domain of interest in terms of an ontology, on the one hand the designer is free to choose the formalism in which to express the ontology, among a variety of different alternatives (e.g., a relational database possibly with constraints, Datalog, or Description Logics). On the other hand, she can select the specific terminology she considers more appropriate to convey the domain semantics. For instance, when creating a biomedical ontology about deseases, the lungs can be modeled as *Pair_of_lungs* or *Both_lungs*. This leads to having complex forms of information, maintained in different formats and organized according to different structures. Often, this information needs to be shared between agents: to reuse the existing ontologies, to integrate knowledge from different agents, and so on. Therefore in recent years, both in the data management and in the knowledge representation communities, several settings have been investigated that address this problem from various perspectives: *(i)* in *information integration*, uniform access is provided to a collection of data sources by means of an ontology (or global schema) to which the sources are mapped [3]; *(ii)* in

* Corresponding author.
E-mail addresses: marenas@ing.puc.cl (M. Arenas), botoeva@inf.unibz.it (E. Botoeva), calvanese@inf.unibz.it (D. Calvanese), ryzhikov@inf.unibz.it (V. Ryzhikov).

http://dx.doi.org/10.1016/j.artint.2016.05.002
0004-3702/

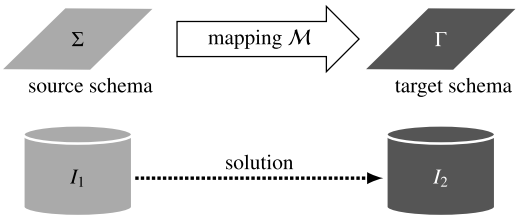

Fig. 1. Data exchange framework.

peer-to-peer systems, a set of peers declaratively linked to each other collectively provide access to the information assets they maintain [4–6]; *(iii)* in *ontology matching*, the aim is to understand and derive the correspondences between elements in two ontologies [7–9]; *(iv)* in *ontology modularity*, the aim is to extract independent, possibly small, subsets of an ontology, so-called modules [10–12]; *(v)* in *knowledge translation*, axioms are being translated from one representation (i.e., logical language and vocabulary) into another [13–15]; and, finally, *(vi)* in *data exchange*, the information stored according to a source schema needs to be restructured and translated so as to conform to a target schema [16,17]. The work we present in this article is inspired by this latter setting investigated in databases.

Data exchange is a field of database theory, motivated by several applications from industry [18,19], that deals with transferring data between differently structured databases. In the seminal article [16], the data exchange problem was defined as the problem of transforming data structured under a source schema into data structured under a target schema, given a mapping specifying how to translate data from the source to the target schema. This problem is depicted in Fig. 1, where the obtained target data instance is referred to as a *solution*. The data exchange problem has been studied for different combinations of languages used to specify the source schema, the target schema and the mapping [17,20,21]. Most of the results in the literature consider source-to-target tuple generating dependencies (tgds) as the language to specify mappings. The dependencies in this class allow one to express containment of conjunctive queries: if a conjunction of several predicates holds, then a conjunction of some other predicates must hold as well. For example, the tgd

$$\forall a, b \,.\, \mathit{AuthorOf}(a, b) \rightarrow \exists y, g \,.\, \mathit{BookInfo}(b, a, y) \wedge \mathit{BookGenre}(b, g) \tag{1}$$

says that if a is the author of a book b, then there exist y and g such that b is a book with author a that was published in year y, and b has genre g. Many database integrity constraints can be expressed by tgds, so these dependencies have been widely used in databases. Source-to-target tgds (st-tgds) are tgds of a special shape: the conjunction on the left-hand side uses only symbols from a source schema, while the conjunction on the right-hand side uses only symbols from a target schema.

A fundamental assumption in the (traditional) data exchange framework is that the source is a complete database: every fact is either true or false. On the other hand, a target instance can be incomplete and a source instance can have many different solutions, as incomplete information can be introduced by the mapping layer (see also [22]).

Example 1.1. If we consider the mapping consisting of the constraint (1), and a source instance consisting of one entry $\mathit{AuthorOf}(\mathsf{tolkien}, \mathsf{lotr})$, encoding that Tolkien is the author of 'The Lord of the Rings', then the following two target instances, I_2 and I_2', are solutions:

$$I_2 = \{\mathit{BookInfo}(\mathsf{lotr}, \mathsf{tolkien}, 1937),\ \mathit{BookGenre}(\mathsf{lotr}, \mathsf{fantasy})\},$$
$$I_2' = \{\mathit{BookInfo}(\mathsf{lotr}, \mathsf{tolkien}, \mathrm{NULL}_1),\ \mathit{BookGenre}(\mathsf{lotr}, \mathrm{NULL}_2)\}.$$

Note that here incompleteness is caused by the existential restriction $\exists y, g \ldots$, which can be satisfied by introducing new objects: either named individuals (or constants), like fantasy, or anonymous objects, like NULL_1. Note also that NULL_1 and NULL_2 are *labeled nulls*, which are widely used in databases to represent anonymous objects. □

To characterize *good* transformations, several criteria have been considered [23]. We emphasize two types of good translations, *universal solutions* and *query solutions*. Universal solutions are the most general solutions: any other solution is more specific (I_2' in Example 1.1 is a universal solution), while query solutions are good solutions from the point of view of answering target queries, i.e., queries formulated over the target schema.

Data exchange with incomplete information. As mentioned before, in the (traditional) data exchange framework, source instances are assumed to contain complete information. However, there are natural scenarios where source instances may contain incomplete information [24,25,21]. In particular, the problem of data exchange with incomplete source data was studied in [25], where an incomplete specification is understood as an object with (possibly infinitely) many interpretations. A simple example of such an object is a database with nulls: assume that we have a table storing information about book genres, and that 'The Lord of The Rings' is a book whose genre is unknown. In this case, the table would consist of an entry of the form $\mathit{BookGenre}(\mathsf{lotr}, \mathrm{NULL})$, which represents all different instances containing a concrete value for the genre of 'The Lord of The Rings': $\mathit{BookGenre}(\mathsf{lotr}, \mathsf{fantasy})$, $\mathit{BookGenre}(\mathsf{lotr}, \mathsf{history})$, $\mathit{BookGenre}(\mathsf{lotr}, \mathsf{scifi})$, etc.