Visualization and Management of Mappings in Ontology-based Data Access (Progress Report)

Domenico Lembo, Riccardo Rosati, Marco Ruzzi,
Domenico Fabio Savo, Emanuele Tocci

Dipartimento di Ingegneria Informatica Automatica e Gestionale
Sapienza Università di Roma
lastname@dis.uniroma1.it

Abstract. In this paper we present an in progress prototype for the graphical visualization and management of mappings for Ontology-based data access (OBDA). The tool supports the specification of expressive mappings linking a *DL-Lite* ontology to a relational source database, and allows the designer to graphically navigate them, according to various possible views, modify their textual representation, and validate their syntactic correctness. Furthermore, it gives preliminary support to the semantic analysis of the mappings, which aims to identify anomalies in the representation, like the specification of mapping queries that cannot return answers without contradicting the ontology. Such functionalities are currently being enhanced in our ongoing development of the tool.

1 Introduction

Ontology-based data access (OBDA) is a data integration paradigm that can be synthesized as follows: given a system composed by an ontology, a source database schema, and a mapping between them, compute the answer to a query posed over the ontology, on the basis of the data stored at the sources [20,17].

Initial research on OBDA has identified "maximal" languages for ontologies and queries that allow for *first-order rewritable* query answering, i.e., query answering that can be reduced to the evaluation of a first-order query (i.e., an SQL query) over the underlying data, organized in the form of an ontology ABox [6,7,14,4,23,13,19]. This is indeed a crucial requirement for OBDA to scale in the size of the data. In this setting, mappings do not complicate the process, since data have a direct connection with the ontology TBox (cf. also the W3C direct mapping recommendation[1]). However, in real-life information integration applications, data are commonly stored in autonomous data sources, and thus "full-fledged" OBDA requires mechanisms to link data to ontologies that go far beyond direct mappings. Such a setting is considered, e.g., in [20,21,10,12]. It has been shown (see, e.g., [5]) that, to preserve first-order rewritability of query answering also in the presence of complex mappings, the linkage between databases and ontologies has to be essentially specified in terms of the so-called *GAV* mapping [11,16], in which predicates of the ontology are seen as views, i.e., queries, over the source database. Interestingly, no restrictions are posed to the language used to specify such queries (e.g., relational data sources can be queried through full SQL).

[1] http://www.w3.org/TR/rdb-direct-mapping/

Whereas OBDA has been quite well-understood from the computational perspective and various reasoning tools for it have been realized in the past years, so far very little has been done to support the design and specification of OBDA systems. Indeed, even though, on the one hand, off-the-shelf tools for the design and management of the ontology level are available [2,1], on the other hand, the support for the specification and management of the mappings is very limited in existing solutions. For example, the Ontop platform [21] extends Protege with a textual mapping editor that provides only basic functionalities, e.g., some syntax highlighting and SQL query execution over the remote DBMS reached through a JDBC connection. In MASTRO STUDIO [8], an environment for OBDA system inspection, documentation, and deployment based on the MASTRO reasoner [20,9], mappings can be only visualized in a textual format, and some functionalities are provided to easily access mappings associated to an ontology predicate, but no support for mapping editing is given. Recently, some editors for specifying R2RML mappings between a relational database and an ontology have been developed (see, e.g., [22,18]). R2RML is the W3C recommendation for mapping relational databases to RDF datasets[2]. The aim of the mentioned tools is to somehow mitigate the effort of encoding mappings in this language, which presents some syntactic technicalities, and to offer a SPARQL endpoint that exploits the mapping for data retrieval. The tool described in [22] provides access to the metadata of the relational database, a plain editor for SQL queries, preview of tuples returned by SQL queries and of triples generated by the mapping, and some support for defining triples starting from the target list of an SQL query. The tool described in [18] offers some basic mechanisms to navigate both the ontology and the relational database and enables for the specification of correspondence assertions between them in a guided manner. Such assertions are automatically translated into R2RML mappings. However, the SQL part of each such mapping can be automatically generated only for simple cases, involving only selections and limited forms of joins. For more complex mappings, the SQL needs to be specified manually. We finally point out that even though the above mentioned tools can be used for specifying mappings for OBDA purposes, they are not specifically designed to this aim.

To meet the needs arising in OBDA, we developed a new tool to support mapping design and specification. The tool, which we present in this paper, has two distinguishing characteristics: (i) it allows designers to *graphically visualize* the mappings, and (ii) it offers functionalities for *semantic verification* of the mappings. The graphical visualization we propose turned out to be extremely useful in some real-life projects we conducted (e.g., [3]), for its ability of naturally showing how data flow from the sources towards the predicates of the ontology. This is particularly helpful when the mapping is constituted by a large number of assertions, as in the mentioned projects, where hundreds of such assertions have been specified.

Semantic verification, instead, aims at identifying anomalous situations in which the mapping may cause unwanted behavior: for instance, it maps the same data into disjoint predicates, whatever the data at the sources are.

Our mapping editor is a stand-alone tool that has been developed within the MASTRO solution for OBDA. It is thus compliant with the framework adopted by MASTRO, i.e., it considers the source database to be relational, the ontology to be expressed in a

[2] http://www.w3.org/TR/r2rml/

language of the *DL-Lite* family [6] or in its OWL counterpart, the profile OWL 2 QL[3], and the mapping to be expressed in a variant of the GAV model. More precisely, the mapping is given in terms of a set of assertions of the form $\Phi \rightsquigarrow \Psi$, where Φ is an SQL query over the source database and Ψ is a conjunctive query without existential variables issued over the ontology. Besides the mentioned features, our tool provides other basic functionalities to support mapping specification. More in detail, its main capabilities are:

- graphical visualization of mappings according to three different views: mapping-centered, ontology-centered, and source-centered;
- mapping editing and update in textual modality;
- simple syntactic check of mappings, i.e., the tool verifies whether a mapping assertion refers only to existing ontology predicates and existing database relational tables, and that it does not contain syntactic errors, either in the SQL query or in the overall assertion;
- advanced semantic check, i.e., the tool verifies whether an assertion, despite its syntactic correctness, presents some anomalies from the logical point of view. In particular, the tool is able to identify situations in which the right-hand side of the mapping has an empty evaluation in every interpretation of the OBDA system, that is, it will always return no answers, independent from the actual data.

We point out that our tool is compliant with the R2RML standard, in the sense that it can export mappings in that format and is able to take an R2RML specification as input, provided that it encodes GAV mappings towards a *DL-Lite* ontology of the form described above.

The paper is organized as follows. In Section 2 we provide some preliminaries. In Section 3 we describe our mapping visualization model. In Section 4 we describe the support that the tool provides for both syntactic and semantic checks. Finally, in Section 5 we conclude the paper.

2 Preliminaries

An OBDA specification is a triple $\langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$, where $\mathcal{T}$ is an ontology, $\mathcal{S}$ is a source database schema, and $\mathcal{M}$ is the mapping between $\mathcal{T}$ and $\mathcal{S}$. More precisely, in our mapping editing tool, $\mathcal{T}$ is a TBox specified in *DL-Lite*$_{A,id,den}$ [15], the most expressive member of the *DL-Lite* family [6], whereas $\mathcal{S}$ is a relational schema. $\mathcal{M}$ is a set of mapping assertions specifying how instances of ontology predicates can be obtained from data stored at the sources. In particular, mapping assertions keep data values separate from objects, and construct object identifiers as (logic) terms over data values. More precisely, object identifiers are *object terms* of the form $f(d_1, \ldots, d_n)$, where f is a function symbol of arity $n > 0$, and $d_1, \ldots, d_n$ are data values. In detail, a mapping assertion is an expression of the form $\Phi(\boldsymbol{v}) \rightsquigarrow \Psi(\boldsymbol{w})$ where

- $\Phi(\boldsymbol{v})$, called the *body* of the mapping, is a first-order logic query over $\mathcal{S}$ with distinguished variables $\boldsymbol{v}$ (we will write such query in the SQL syntax), and

[3] http://www.w3.org/TR/owl2-profiles/

– $\Psi(\boldsymbol{w})$, called the *head*, is a conjunctive query whose atoms are of the form $A(f(\boldsymbol{v}'))$, or $P(f(\boldsymbol{v}'), f'(\boldsymbol{v}''))$, or $U(f(\boldsymbol{v}'), v'')$, where A, P, and U denote respectively an atomic concept, an atomic role and an attribute, f, f' are function symbols, $\boldsymbol{v}', \boldsymbol{v}''$ are sequences of variables occurring in $\boldsymbol{v}$, and v'' is a variable occurring in $\boldsymbol{v}$.

Example 1. Consider the following mapping assertions:

M_1 : `SELECT S_CODE, S_NAME FROM STUDENTS WHERE DOB<='1990/01/01'` $\rightsquigarrow$ *Student*(**st**(S_CODE)), *name*(**st**(S_CODE), S_NAME)

M_2 : `SELECT S_CODE,C_CODE FROM COURSES` $\rightsquigarrow$ *takesCourse*(**st**(S_CODE), **co**(C_CODE))

Such assertions relate a database schema containing the relational tables STUDENTS and COURSES to an ontology containing the concept *Student*, the attribute *name*, and the role *takesCourse*. M_1 is a mapping assertion that in its body selects, from the table STUDENTS, the code (variable S_CODE) and the name (variable S_NAME) of students whose date of birth (variable DOB) is after December 31, 1989. For each tuple returned by its body, M_1 builds an object term of the form **st**(S_CODE), representing an instance of the concept *Student*, and an object/value pair representing an instance of the attribute *name*, which is composed by such object term and the value denoted by the variable S_NAME. Similarly, M_2 is a mapping assertion relating the relation COURSES to the role *takesCourse*. More precisely, from each tuple constituted by the code of a student (variable S_CODE) and the code of a course she takes (variable C_CODE), M_2 builds a pair of object terms (**st**(S_CODE),**co**(C_CODE)), where **co** is a function symbol used to build object terms representing courses taken by students. Such a pair represents an instance of the role *takesCourse*.

The mapping editor we present in this paper is able to process mapping assertions written in the R2RML syntax. R2RML is the W3C standard to specify how RDF triples can be built starting from data stored in relational databases by means of so-called *triple-maps*. Actually, our mapping language is a subset of the R2RML standard, as shown by the following example.

Example 2. The mapping assertions M_1 and M_2 given in Example 1 can be represented in R2RML by means of the triple maps <#M1> and <#M2> described below.

```
<#M1>
  rr:logicalTable <#Table1>
  rr:subjectMap [
    rr:template "http://uniroma1.it/st/{S_CODE}"
    rr:class ex:Student
  ];
  rr:predicateObjectMap [
    rr:predicate ex:name
    rr:objectMap [rr:column "S_NAME"]
  ].

<#M2>
  rr:logicalTable [ rr:tableName "COURSES" ]
  rr:subjectMap [
    rr:template "http://uniroma1.it/st/{S_CODE}"
```

```
  ];
  rr:predicateObjectMap [
    rr:predicate ex:takesCourse
    rr:objectMap [rr:template "http://uniroma1.it/co/{C_CODE}"]
  ].
```

<#Table1> used in mapping <#M1> is defined as follows.

```
<#Table1> rr:sqlQuery """
  SELECT S_CODE, S_NAME FROM STUDENTS
  WHERE DOB <= '1990/01/01'
""".
```

We note that function symbols used in our language to build object instances are replaced in R2RML by template URI definitions.

3 Visualization features

This section introduces the visualization functionalities provided by our tool. The visualization is designed according to the following goals:

1. graphically represent a single mapping assertion;
2. graphically represent all the mapping assertions involving a given ontology predicate;
3. graphically represent all the mapping assertions involving a given relational table of the source database;
4. provide the user with the capability of editing the textual representation of a mapping assertion.

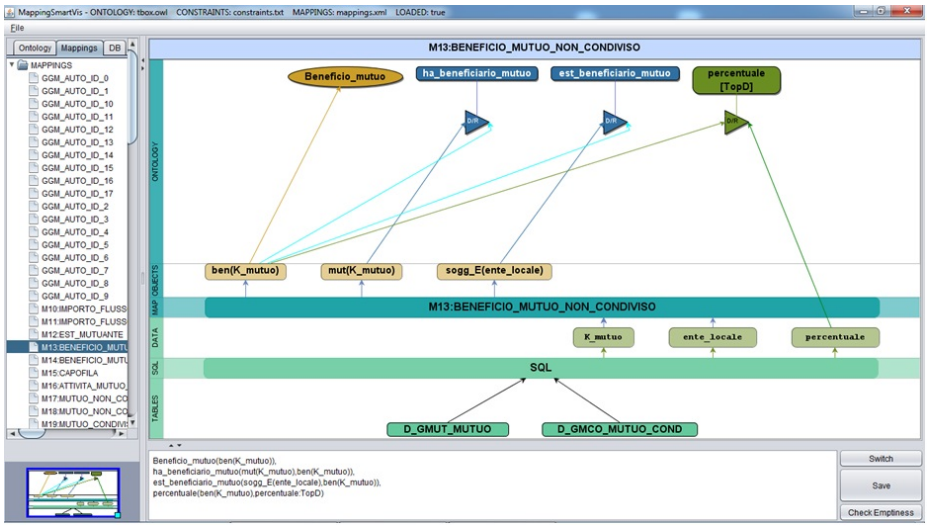

Fig. 1: Mapping-centered view.

Figure 1 presents the user interface of our tool. The window displayed by the figure contains: (i) a menu bar in the upper side, with only the item File, through which it is possible to access the functionality that initializes the MASTRO system, save modifications on mapping assertions, and export mappings in the R2RML format; (ii) a tabbed pane in the left side, containing three tabs, namely, Ontology (which shows the list of all the predicates in the ontology alphabet, grouped into CONCEPTS, ROLES, and ATTRIBUTES) Mappings (which lists the IDs of all the mapping assertions) and DB (which lists all the tables in the source database); (iii) a text area in the bottom side, which is used to edit the head and the body of a mapping assertion, and which can be shifted to the right side of the window; (iv) a frame in the center, which provides the actual mapping visualization according to one of the modalities described in the following; and (v) an outline of the central frame in the left bottom corner.

As shown in the figure, colors used in the graphical representation of ontology predicates are as in Protégé. In particular, concepts are denoted by orange ellipses, roles by blue rounded rectangles, and attributes by green rounded rectangles. The overall visualization is built by means of the Java open source graphic library JGraphX.

According to the objectives mentioned at the beginning of this section, our tool allows users to visualize mapping assertions by selecting one of three different views, called mapping-centered, ontology-centered, and source-centered, respectively. Each such view is conceived as a directed, layered graph, which actually traces the layers of an OBDA system. More in detail, each such graph is organized according to the following six layers (listed starting from the upper one):

1. the ONTOLOGY layer, in which a node represents a predicate in the alphabet of the ontology.
2. the OBJECTS layer, in which a node represents an object built from data in the DATA layer,
3. the MAP layer, in which a node represents the head of a mapping assertion,
4. the DATA layer, in which a node represents a column of the result set retrieved by an SQL query in the body of a mapping assertion,
5. the SQL layer, in which a node represents an SQL query in the body of a mapping assertion,
6. the TABLES layer, in which a node represents a table in the source schema.

All the graph edges are oriented from lower to upper layers.
In the following, we describe in detail the graph provided by each view.

Mapping-centered view. This view aims to reach the first of the above visualization goals, i.e., representing a single mapping assertion once the user selects its ID in the MAPPINGS tab (see Figure 1).

The nodes and connections allowed in this view are listed below:

– Each node in the ONTOLOGY layer can be either a predicate node or an property-argument node. A predicate node represents a concept, role, or attribute occurring in the conjunctive query in the head of the mapping, and is depicted as described above. Every predicate node representing a role or an attribute S is connected to a property-argument node for each occurrence of an atom with predicate S in the head of the mapping assertion. These nodes have the shape of a triangle, have the

same color of the predicate node to which they are connected, and are labeled with the string 'D/R', which stands for predicate domain and range. Intuitively, they are used to distinguish the first and the second argument of an atom with predicate S. Indeed, each such node has two incoming edges by lower layers: the one ending in its left-hand side connects the node with the term occurring in its first argument, whereas the one ending in its right-hand side connects the node with the term occurring in its second argument.

- Each node in the OBJECTS layer represents a term of the form $f(\boldsymbol{v})$ occurring in the right-hand side of the mapping assertion. The label of each such node has therefore the form `f(alias1,...,aliasN)`, where `alias1,...,aliasN` are the aliases in the target list of the SQL query in the left-hand side of the assertion. The nodes in this layer are linked either to concept nodes or to property-argument nodes in the ONTOLOGY layer. Intuitively, in the first case OBJECTS nodes instantiate a concept, whereas in the second case they instantiate the role or the attribute linked to the property-argument node. In this respect, we notice that the connection with property-argument nodes relative to attributes can be only towards their left-hand side.
- The node in the MAP layer represents the head of a mapping assertion and is labeled with the ID of the mapping assertion. The node in this layer is linked to each node in the OBJECTS layer.
- Each node in the DATA layer represents a field of the target list of the SQL query in the body of the mapping assertion and is labeled with the SQL alias of such field. Every node in this layer is linked either to the node in the MAP layer or directly to property-argument nodes in the OBJECTS layer. In this latter case, the property-argument nodes are in turn linked to a node representing an attribute.
- The node in the SQL layer is labeled with 'SQL' and represents the body of the mapping assertion. The node in this layer is linked to each node in the DATA layer.
- Each node in the TABLES layer represents a table in the data source referenced by the body of the mapping assertion and is labeled with the name of the table it represents. All the nodes in this layer are linked to the node in the SQL layer.

To access the textual specification of the mapping, the user can click on the nodes in the SQL and MAP layers. In this way she visualizes in the text area in the bottom side of the window the body and the head of the mapping assertion, respectively. The text area can then be edited to modify the mapping.

When a node in the ONTOLOGY layer is double-clicked, the tool switches to the ontology-centered view relative to the selected node; similarly, by double-clicking a node in the TABLES layer, the corresponding source-centered view is presented.

When the node in the MAP layer is double-clicked, all the nodes in the OBJECTS layer are shown together with the edges linking them to the predicates that they instantiate. The user can visualize the result of some simple reasoning over the mapping assertion. For example, when a node X in the OBJECTS layer is clicked, the tool highlights all the edges in the paths starting from the node(s) in the DATA layer representing the data used to build the object associated to X and ending in the node(s) representing the predicates instantiated by the object associated to X. Something similar can be done with the nodes in the DATA layer.

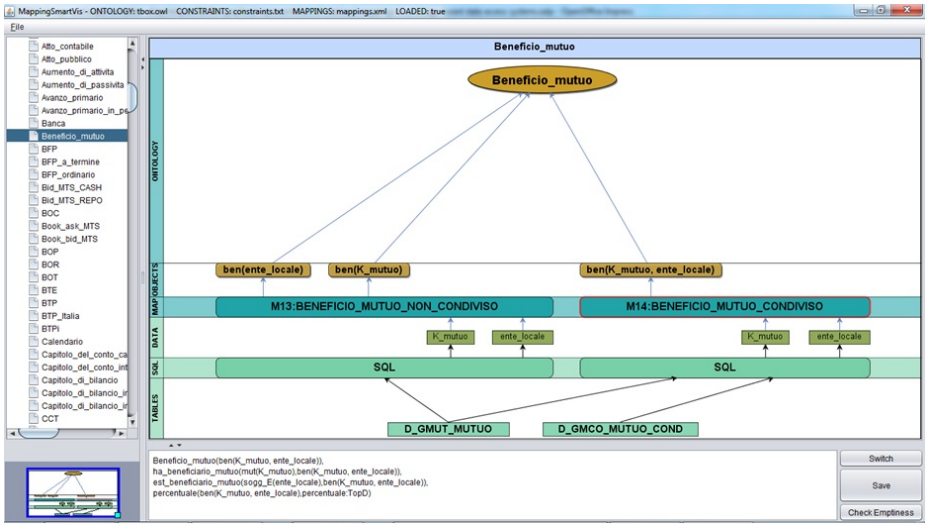

Fig. 2: Ontology-centered view.

Ontology-centered view. This view aims to reach our second visualization goal, i.e., representing all the mapping assertions involving a given predicate, once the user selects it in the ONTOLOGY tab (see Figure 2).

The nodes and connections allowed in this view are listed below:

- The node in the ONTOLOGY layer represents the predicate on which the view is centered. If the node represents a role or an attribute it is also linked to a triangle-shaped child node (of the same color) with label 'D/R'.
- Each node in the OBJECTS layer represents an object built by mean of a function symbol, analogously to nodes in the OBJECTS layer of the mapping centered view. Also connections towards nodes in the ONTOLOGY layer are done in the same way, and obviously depend on the kind of predicate the view is centered on.
- The MAP, DATA, SQL, and TABLES layers are as in the mapping-centered view, but in this case they are representing more than one mapping. Therefore, both the MAP layer and the SQL layer may contain more than one node, i.e., a node for each mapping assertion that involves the predicate on which the view is centered. Also, each node in the TABLES layer can be connected to more than one node in the SQL layer, if the table it represents occurs in more than one SQL mapping query of interest for the predicate on which the view is centered.

As in the mapping-centered view, the user can click on a nods in the SQL or MAP layer to visualize, and also edit in the text area, the body and the head of a mapping assertion. Again, the switch to other views is possible by double-clicking nodes in the MAP and TABLES layers, and some path-based visualization functionalities are allowed in a way similar to the ontology-centered view.

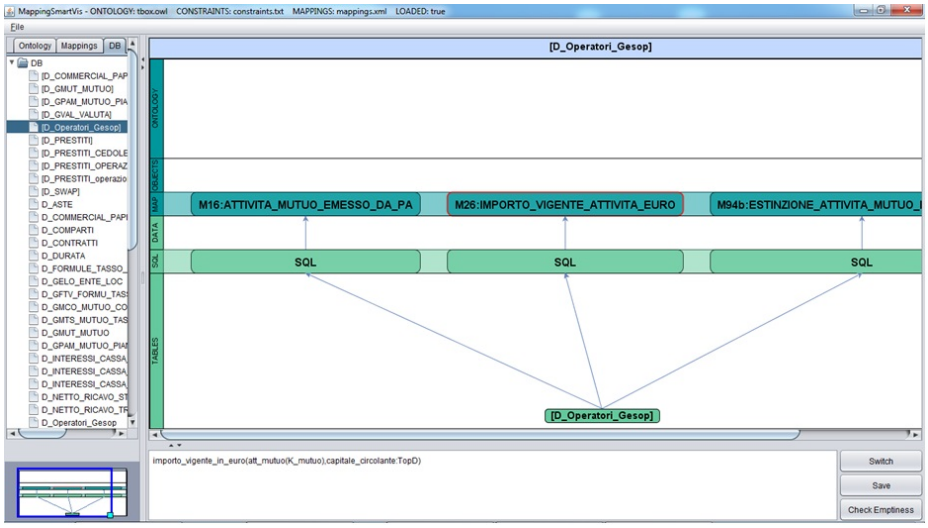

Fig. 3: Source-centered view.

Source-centered view. This view aims to reach the third of our visualization goals, i.e., representing all the mapping assertions getting data from a table selected by the user in the DB tab (see Figure 3).

The nodes and connections allowed in this view are listed below:

- There are no nodes in the ONTOLOGY, OBJECTS and DATA layers.
- Each node in the MAP layer represents the head of a mapping assertion and is labeled with the ID of the mapping assertion.
- Each node in the SQL layer is labeled with "SQL" and represents the body of a mapping assertion. The nodes in this layer are linked to the related node in the MAP layer.
- The node in the TABLES layer represents the table on which the view is centered. This node is linked to all the nodes in the SQL layer.

Visual functionalities similar to those described for the other views are available also in the mapping-centered view.

4 Editing and analysis features

In this section, we present the functionalities currently provided by our tool for mapping editing support. In particular, we describe some features that help the mapping designer to fulfill a correct specification by identifying possible flaws in the mappings.

We consider two categories of problems: syntax errors and semantic anomalies. Issues of the first kind obviously refers to an incorrect use of the mapping syntax, whereas problems of the second kind concern situations in which, despite its syntactic correctness, the mapping specification may be considered somehow logically incoherent. We

point out that we can safely assume here that the other components of the OBDA system we are managing do not present similar problems, since they are checked through the use of other tools that are part of our OBDA solution. In particular, we can assume that the ontology is a legal *DL-Lite*$_{A,id,den}$ TBox, i.e., it is syntactically well-formed, and also that it does not present semantic anomalies, i.e., it does not contain unsatisfiable concepts, roles, or attributes[4].

Among syntax errors, we observe that those that can be more frequently found in mappings assertion specified without any support are: (i) the presence, in the head query, of predicate atoms that do not belong to the ontology alphabet, often simply due to misspellings; (ii) the use, in the head query, of variables that do not occur in the target list of the SQL body query; (iii) the presence of syntactic errors in the SQL query. In our tool, we are able to identify most of the syntax errors through a dedicated parser, which, for example is able to find out and report the user with all errors of kind (i) and (ii) above. As for SQL syntax verification, we rely on the external parser provided by the DBMS managing the source data.

For what concerns semantic problems for the mapping, given an OBDA specification $\langle \mathcal{T}, \mathcal{M}, \mathcal{S} \rangle$ we say that a mapping assertion $m \in \mathcal{M}$ is semantically anomalous if the answer to either the head query of m or the body query of m is "intensionally" empty. More formally, let m be the assertion $\Phi \rightsquigarrow \Psi$, intensional emptiness of the head query means that the evaluation of the conjunctive query Ψ over every model of $\mathcal{T}$ is empty, whereas intensional emptiness of the body query means that the evaluation of the SQL query Φ over every database instance for $\mathcal{S}$ is empty. Having one such query (or both) empty in the sense described above may be either useless or dangerous in our specification. Indeed, if Φ is empty, the assertion m is useless, since it will never contribute to instantiating the ontology predicates. On the other hand, if Ψ is empty and Φ is not, the assertion may lead to a contradiction, since the semantics of the mapping is imposing that the tuples returned by Φ instantiate the ontology in such a way that Ψ is not empty, while the semantics of the TBox $\mathcal{T}$ is imposing the intensional emptiness of Ψ (cf. the examples given below).

We observe that the identification of semantic anomalies related to the emptiness of SQL queries requires to solve an undecidable problem. Even though approximate solutions to this problem could be adopted, in the current implementation of our tool we have restricted this capability only to the pinpointing of semantic anomalies concerning the emptiness of the head queries of mappings, i.e., conjunctive queries issued over a *DL-Lite*$_{A,id,den}$ TBox. If this is the case, we say the the mapping m is *semantically anomalous* with respect to $\mathcal{T}$.

We observe that semantic anomalies of this kind may occur only in one of the following cases:

- In the head of m, a concept, role, or attribute S occurs such that $\mathcal{T} \models S \sqsubseteq \neg S$.
- In the head of m, some join variables occur in positions relative to predicate arguments that are entailed to be disjoint by $\mathcal{T}$. For instance, consider an ontology that specifies that someone who is a Student cannot be also a Professor. Then the mapping assertion

$$\texttt{SELECT S\_CODE FROM T1} \rightsquigarrow \textit{Student}(\mathbf{st}(\texttt{S\_CODE})), Professor(\mathbf{st}(\texttt{S\_CODE}))$$

[4] We recall that a *DL-Lite*$_{A,id,den}$ TBox is always satisfiable [15]

is semantically anomalous with respect to $\mathcal{T}$.

– The query in the head of m is entailed to be empty by a denial assertion in $\mathcal{T}$. For instance, consider an ontology $\mathcal{O}$ in which the denial assertion $\forall x.(isParentOf(x, x)) \rightarrow \bot$ is used to specify the irreflexivity of the $isParentOf$ role. Then the mapping assertion

$$\texttt{SELECT ID FROM T2} \rightsquigarrow isParentOf(\mathbf{p}(\texttt{ID}), \mathbf{p}(\texttt{ID}))$$

is semantically anomalous with respect to $\mathcal{T}$.

– The query in the head of m is forced to be empty by functional assertions over roles or attributes, or by identification assertions specified in $\mathcal{T}$. For instance, consider an ontology $\mathcal{T}$ in which the functionality of the role $bornIn$ is asserted. Then, the mapping assertion

$$\texttt{SELECT ID}_1\texttt{, ID}_2\texttt{ FROM T3} \rightsquigarrow bornIn(\mathbf{p}(\texttt{ID}_1), \mathbf{w}(\texttt{ID}_2)), bornIn(\mathbf{p}(\texttt{ID}_1), \mathbf{k}(\texttt{ID}_2))$$

is semantically anomalous with respect to $\mathcal{T}$. Indeed, it is easy to verify that, due to the use of the two different function symbols **w** and **k** in head of the mapping assertion, for every pair of values (ID_1, ID_2), the functionality of $bornIn$ is violated.

Our tool identifies mapping assertions that are semantically anomalous with respect to the TBox through query containment checks. More specifically, the tool first properly converts disjunction, denial, functional, and identification assertions in conjunctive queries (possibly with inequalities), and then, for each of such queries, it verifies if there is a query in the head of some mapping assertion that is contained in it.

5 Conclusion

The mapping visualization model described in Section 3 is the result of the iterative refinement of various proposals that have been tested and developed during some industrial OBDA projects in which we have been recently involved (see, e.g., [3]). Even though such projects have been a useful testbed for our tool, we plan to further assess the effectiveness of our solution through some user evaluation tests, aimed at validating both the graphical model and the usability of the tool.

Our experiences also allowed us to identify main analysis functionalities for our tool. The relative implementation is still ongoing. In particular, we are working on the semantic analysis of SQL queries in the mappings, to discover properties and relations between them that are useful to prevent situations which may lead to contradictions (this process relies on the use of a first-order theorem prover).

Furthermore, we plan to also implement functionalities for the graphical editing of the mapping, which in the current release can be modified only in textual modality.

We finally remark that the graphical representation of mappings presented in this paper is independent of the ontology language used in the OBDA specification: that is, it can be used with ontologies that go beyond the expressiveness of DL-Lite. On the other hand, the semantic checks identified by our approach are actually tailored for the *DL-Lite*$_{A,id,den}$ language.

Acknowledgments. This research has been partially supported by the EU under FP7 project Optique (grant n. FP7-318338).

References

1. The Neon Toolkit, http://neon-toolkit.org/
2. Protégé, http://protege.stanford.edu/
3. Antonioli, N., Castanò, F., Civili, C., Coletta, S., Grossi, S., Lembo, D., Lenzerini, M., Poggi, A., Savo, D.F., Virardi, E.: Ontology-based data access: the experience at the Italian Department of Treasury. In: Proc. of the Industrial Track of the 25th Int. Conf. on Advanced Information Systems Engineering (CAiSE 2013). CEUR Electronic Workshop Proceedings, http://ceur-ws.org/, vol. 1017, pp. 9–16 (2013)
4. Calì, A., Gottlob, G., Lukasiewicz, T.: A general datalog-based framework for tractable query answering over ontologies. Semantic Web J. 14, 57–83 (2012)
5. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rosati, R., Ruzzi, M.: Data integration through *DL-Lite*$_\mathcal{A}$ ontologies. In: Schewe, K.D., Thalheim, B. (eds.) Revised Selected Papers of the 3rd Int. Workshop on Semantics in Data and Knowledge Bases (SDKB 2008). Lecture Notes in Computer Science, vol. 4925, pp. 26–47. Springer (2008)
6. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. J. of Automated Reasoning 39(3), 385–429 (2007)
7. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Data complexity of query answering in description logics. Artificial Intelligence 195, 335–360 (2013)
8. Civili, C., Console, M., De Giacomo, G., Lembo, D., Lenzerini, M., Lepore, L., Mancini, R., Poggi, A., Rosati, R., Ruzzi, M., Santarelli, V., Savo, D.F.: MASTRO STUDIO: Managing ontology-based data access applications. Proc. of the VLDB Endowment 6, 1314–1317 (2013)
9. De Giacomo, G., Lembo, D., Lenzerini, M., Poggi, A., Rosati, R., Ruzzi, M., Savo, D.F.: Mastro: a reasoner for effective ontology-based data access. In: Proc. of the OWL Reasoner Evaluation Workshop (ORE 2012). CEUR Electronic Workshop Proceedings, http://ceur-ws.org/, vol. 858 (2012)
10. Di Pinto, F., Lembo, D., Lenzerini, M., Mancini, R., Poggi, A., Rosati, R., Ruzzi, M., Savo, D.F.: Optimizing query rewriting in ontology-based data access. In: Proc. of the 16th Int. Conf. on Extending Database Technology (EDBT 2013). pp. 561–572 (2013)
11. Doan, A., Halevy, A.Y., Ives, Z.G.: Principles of Data Integration. Morgan Kaufmann (2012)
12. Haase, P., Horrocks, I., Hovland, D., Hubauer, T., Jiménez-Ruiz, E., Kharlamov, E., Klüwer, J.W., Pinkel, C., Rosati, R., Santarelli, V., Soylu, A., Zheleznyakov, D.: Optique system: towards ontology and mapping management in OBDA solutions. In: Proc. of the 2nd Int. Workshop on Debugging Ontologies and Ontology Mappings. pp. 21–32 (2013)
13. König, M., Leclère, M., Mugnier, M.L., Thomazo, M.: On the exploration of the query rewriting space with existential rules. In: Proc. of the 7th Int. Conf. on Web Reasoning and Rule Systems (RR 2013). pp. 123–137 (2013)
14. Kontchakov, R., Lutz, C., Toman, D., Wolter, F., Zakharyaschev, M.: The combined approach to ontology-based data access. In: Proc. of the 22nd Int. Joint Conf. on Artificial Intelligence (IJCAI 2011). pp. 2656–2661 (2011)
15. Lembo, D., Lenzerini, M., Rosati, R., Ruzzi, M., Savo, D.F.: Inconsistency-tolerant first-order rewritability of DL-Lite with identification and denial assertions. In: Proc. of the 25th Int. Workshop on Description Logic (DL 2012). CEUR Electronic Workshop Proceedings, http://ceur-ws.org/, vol. 846 (2012)
16. Lenzerini, M.: Data integration: A theoretical perspective. In: Proc. of the 21st ACM SIGACT SIGMOD SIGART Symp. on Principles of Database Systems (PODS 2002). pp. 233–246 (2002)

17. Lenzerini, M.: Ontology-based data management. In: Proc. of the 20th Int. Conf. on Information and Knowledge Management (CIKM 2011). pp. 5–6 (2011)
18. Neto, L.E.T., Vidal, V.M.P., Casanova, M.A., Monteiro, J.M.: R2RML by assertion: A semi-automatic tool for generating customised R2RML mappings. In: The Semantic Web: ESWC 2013 Satellite Events. pp. 248–252 (2013)
19. Ortiz, M.: Ontology based query answering: The story so far. In: Proc. of the 6th Alberto Mendelzon Int. Workshop on Foundations of Data Management (AMW 2013) (2013)
20. Poggi, A., Lembo, D., Calvanese, D., De Giacomo, G., Lenzerini, M., Rosati, R.: Linking data to ontologies. J. on Data Semantics X, 133–173 (2008)
21. Rodriguez-Muro, M., Kontchakov, R., Zakharyaschev, M.: Ontology-based data access: Ontop of databases. In: Proc. of the 12th Int. Semantic Web Conf. (ISWC 2013). pp. 558–573 (2013)
22. Sengupta, K., Haase, P., Schmidt, M., Hitzler, P.: Editing R2RML mappings made easy. In: Proc. of the 12th Int. Semantic Web Conf., Posters & Demos Track (ISWC 2013). pp. 101–104 (2013)
23. Venetis, T., Stoilos, G., Stamou, G.B.: Query extensions and incremental query rewriting for OWL 2 QL ontologies. J. on Data Semantics 3(1), 1–23 (2014)