Towards efficient and practical solutions for ontology-based data management

Valerio Santarelli
DIAG, Sapienza Università di Roma
santarelli@dis.uniroma1.it
Supervised by Domenico Lembo

ABSTRACT
This paper presents the research summary of my Ph.D. plan, which is currently in the early stages of its development. The objectives around which my work is focusing and that I will continue to pursue throughout the remainder of my doctoral activities are the development of tools and the definition of a methodology for the execution of those tasks that are typically performed when working with ontologies in the context of Ontology-Based Access (OBDA). The objective of this methodology is to guide the ontology engineer through ontology design, representation and approximation, and the issues I will address in order to provide the necessary tools to achieve this goal are the optimization of ontology classification in DL-Lite, the definition of a graphical language tailored towards ontologies, and the study of approximation of ontologies in expressive languages into DL-Lite logics. Such methodology proves to be necessary when facing the numerous challenges that often arise in real-world scenarios, in which the amount of data that is managed by information systems, and the processes that manage and act on this data grow continuously.

1. INTRODUCTION
The research project I am pursuing, which will be the focus of my doctorate thesis, is centered around Ontology-Based Information Systems. The purpose of such systems is to explicit the information content, independently from the underlying data structures which store this information. This goal is achieved through the use of Ontology-Based Access (OBDA) [4], a framework in which an ontology is used as a conceptual view over the underlying data sources, allowing users to access data without needing to possess specific knowledge on how the data is actually organized and where it is stored. In other words, in OBDA the aim is to use an ontology to mediate data access. The added value of OBDA with respect to direct access to data sources is two fold: on one hand, the ontology provides a semantic account of the information that is stored in the sources; on the other, query answering can be enriched by exploiting the constraints that can be expressed by the ontology.

To specify the semantic correspondence between the unified view of the domain and the data stored at the sources, the OBDA approach, analogously to data integration systems, uses an intermediate mapping layer between the global schema and the data sources. However, the unique feature of this approach is the fact that the global unified view is provided in terms of a conceptualization of the domain which is constructed independently from the data source schema. This conceptualization of the domain is, of course, formalized through an ontology.

The significance of this approach emerges clearly when dealing with large organizations, where, on one hand, information regarding the data is widespread and often difficult to access, and, on the other hand, there is a proliferation of data sources and services that are relevant for the organization. In such settings, the ontology and the corresponding mappings to the data sources provide a common ground for the documentation and the access to the information in the organization, with clear benefits for the management of the information system. Therefore, my work in this context aims at providing solutions that can be applied effectively to the numerous tasks that are typically undertaken when working with Ontology-Based Information Systems in such practical settings.

Ontology-Based Information Systems, and more specifically Ontology-Based Data Access, has been the focus of many different studies [4] [28] [22]. However, these studies typically focus on the core aspects of OBDA, such as query answering. Less attention, in this context, has been dedicated to other aspects such as ontology visualization, intentional reasoning (e.g. ontology classification), and content handling. These problems, while representing independent research fields, all play a significant part in the construction of an OBDA architecture in real-life settings, and therefore must be addressed. The results that are found in literature regarding these issues are unsatisfactory when analyzing them in the context of ontology-based data access, because they are not tailored towards the languages and the applications that are adopted under the OBDA framework.

The focus of my work therefore is the study of these issues while maintaining OBDA as the goal. This work is being developed by following two different objectives. On one hand, the improvement of reasoning services such as ontology classification and logical implication, which is achieved by implementing new algorithms and by optimizing solutions that are currently used. On the other hand, from a more practical point of view, the project’s aim is the realization of new tools, or the contribution to the improvement of existing ones, that are a primary necessity for the success of the ontology-based data access approach, as I have learned throughout my experience with working on the design and the integration of OBDA systems.

The final goal of my Ph.D. work is hence the development of a methodology for OBDA which starts from ontology design, through the use of a graphical language that is tailored to onto-
ogy representation, proceeds through the translation into logical axioms, takes advantage of tools for design quality control (intentional reasoning, i.e. ontology classification), and concludes with the integration into a single system that is equipped to offer a wide variety of services, through new or already existing contributions.

2. STATE OF THE ART OF THE RESEARCH FIELD

Ontologies provide a formal and explicit conceptualization of a domain of interest (the most effective definition of ontologies in computer science is probably in [16]). Through its role as a standardized reference model, the purpose of an ontology is to allow knowledge sharing and communication, and this goal is achieved by supporting human communication and comprehension of the domain of interest, and by facilitating access, communication, and integration among different systems. The importance of ontologies in modern information systems, and their use as a conceptual view over data repositories is constantly growing. In many fields such as Data Integration [24] [18] and the Semantic Web [3] [32], the use of an ontology to represent the domain of interest allows users to rely on a shared conceptualization of this domain when accessing the services provided by the underlying information system; furthermore, ontologies constitute the foundation of the ontology-based data access (OBDA) [4] approach, which allows users to access data that is spread out over different sources without having to possess specific knowledge on how this data is organized, but instead interacting with and querying on the ontology that is linked to the real data through appropriate mappings.

To model the domain and to allow for reasoning over it, an ontology must be represented by a well-defined language. Description Logics (DLs), which structure knowledge in terms of classes of objects and binary relations between them, are designed to achieve these goals, making them the ideal solution for ontology representation. Traditionally, expressive DLs have been used for these tasks, with a strong focus on intensional level reasoning, i.e. reasoning only on the schema level of the ontology but disregarding data, or simple forms of query answering, such as instance checking. However, in an effort to allow for more efficient extensional level reasoning over ontologies, tractable Description Logics [7] such as the QL, EL, and RL profiles of OWL 2 QL\(^1\), which are designed to limit the computational complexity of reasoning services with respect to the data, have recently been developed and adopted. Currently, various reasoners such as FaCT++ [35], Hermit [33], and Pellet [34] offer numerous services for ontology reasoning, but these systems are mainly tailored towards expressive DLs and are not equipped with OBDA features (in particular they do not allow for the specification and use of mappings towards external data sources).

Two examples of OBDA systems are Mastro [4] and Quest [28]. Mastro offers a wide array of reasoning services over DL-Lite [5], a data-oriented DL which is the logical underpinning of the OWL 2 QL profile. In particular, Mastro offers conjunctive query answering, mapping management, constraint management, consistency checking, and answering of expressive queries (beyond conjunctive queries) under suitable semantic approximations [6]. Quest instead provides SPARQL query answering under the OWL 2 QL or RDFS entailment regimes while focusing on large volumes of data and ontologies.

OBDA systems such as Mastro and Quest can be considered tools for an advanced form of semantic data integration [9] [31] [25] [14] [12]. However, in these systems, the integrated view provided to clients is not simply a data structure accommodating the various data at the sources, but a semi-structurally rich description of the relevant concepts of the domain, as well as of the relationships between them. As already stated, in such systems aspects such as ontology visualization, evolution, and intentional reasoning have been so far overlooked, while the focus has mainly been placed on query answering.

As stated in the introduction, my work will focus on a specific subset of these problems, to develop solutions and tools to be integrated in the Mastro system. In the following sections, I will begin the discussion of these problems with a brief state of the art overview for each one.

3. RESEARCH OBJECTIVES AND METHODOLOGY

My experience in working with the DASI-lab at the Department of Computer, Control, and Management Engineering “Antonio Ruberti” at Sapienza Università di Roma has provided valuable examples of scenarios in which my doctorate thesis work can provide numerous advantages when working in real-life scenarios. I have had the opportunity to work on projects commissioned by the Italian Ministry of Economy and Finance, the Monte dei Paschi di Siena banking institute, and I am currently heavily involved in a project in collaboration with the Telecom Italia telecommunications company. These projects, which center around the design and integration of OBDA systems, corroborate the validity of the ideas behind the OBDA method, proving that the declarative approach to data access and integration, and the logical and physical independence of the information system aid in allowing data access to users who are not necessarily fully aware of the details of the organization of the underlying data layer. However, they also prove that the construction of the OBDA architecture is still not supported by a formal schema, and that the development of new tools and of a methodology for the execution of the various tasks which are necessary in building this architecture is a crucial step towards the success of the ontology-based data access approach.

Instead of focusing on the study and improvement of a specific aspect of the OBDA approach, the objective of my doctorate thesis, as stated in the introduction, is to develop a methodology for OBDA, and to provide, when lacking, the necessary tools for its development. The goal of this methodology is to define a workflow that guides the ontology engineer through the process of ontology design, visualization, and formalization, allowing him to produce the documentation that, along with the specification of mappings and data sources, forms the requisites for the execution of the core services of OBDA, i.e. query rewriting, consistency checking, and ontology query answering.

This workflow calls for: (i) definition of the ontology through a graphical language, which can be enriched with auxiliary documentation regarding the design choices that were made, (ii) translation of this graphical formalization of the ontology into a set of processable logical axioms, through an automated tool, (iii) refinement of these axioms for OBDA aims (i.e., adoption of OBDA languages possibly equipped with suitable constraints), (iv) execution of intentional reasoning tasks such as ontology classification, to verify the quality and correctness of the choices made during ontology design.

The issues that I will tackle during my doctorate work are the following:

- optimization of intentional reasoning tasks over ontologies, with a specific focus on ontology classification;
- definition of a graphical language tailored towards ontologies which allows for a direct translation into OWL;

\(^1\)www.w3.org/TR/owl2-profiles/
approximation of ontologies formulated in expressive languages (i.e. OWL) into logics of the DL-Lite family.

After introducing some necessary preliminaries in Section 4, in the continuation of this paper I will illustrate the methodologies and the techniques that I have developed and that I am studying in order to attempt to solve these problems. I will also discuss their evaluation and validation through application in real-life scenarios in the context of OBDA project-related activities.

4. PRELIMINARIES

In this section, I present some basic notions on description logic ontologies and the DL-Lite family of logics.

Description Logic ontologies. Description Logics (DLs) [2] are decidable logics that can be used to represent the knowledge of a domain of interest in terms of objects, representing individuals, concepts, which are abstractions for sets of objects, and roles, which denote binary relations between objects. In addition, some DLs distinguish concepts from values, and roles from attributes, which denote binary relations between objects and values. DLs are the formal underpinning of the OWL 2 language, where concepts are called classes, roles are called object-properties, and attributes are called data-properties.

We consider a signature $\Sigma$ containing symbols for (atomic) concepts and (atomic) roles. Given a DL language $\mathcal{L}$, an $\mathcal{L}$-TBox (or simply a TBox, when $\mathcal{L}$ is clear) over $\Sigma$ contains universally quantified first-order (FOL) assertions, i.e., axioms specifying general properties of concepts, roles, and attributes. An $\mathcal{L}$-ABox (or simply an ABox, when $\mathcal{L}$ is clear) is a set of assertions on individual constants, which specify extensional knowledge. In OBDA, and in this paper, the only relevant component of the ontology is the TBox. Indeed, the information about the instances of concepts, roles, and attributes is not provided by the ABox, but by the combination of the database and the mappings.

The DL-Lite family. One of the most critical aspects that are faced when working with Description Logics involves the trade-off between the expressive power of the language and the computational complexity of sound and complete reasoning. Many DLs with efficient reasoning algorithms lack the modeling power that is required to capture conceptual models and basic ontology languages, while other DLs that possess this modeling power suffer from inherently worst-case exponential time behavior in reasoning. When dealing with relatively small ontologies, the requirement for polynomial tractable reasoning is less stringent [7]. However, when working with Ontology-Based Information Systems, it is common, as we have seen, to deal with huge quantities of data, and in these cases the need for efficient reasoning is paramount.

The DL-Lite family of DLs is specifically tailored for this purpose: it is rich enough to capture significant ontology languages, and is able to maintain all reasoning tasks tractable, particularly with polynomial time complexity w.r.t. the size of the knowledge base. By reasoning in this context one refers not only to computing of subsumptions and checking the satisfiability of the knowledge base, but also to answering complex queries, i.e., unions of conjunctive queries, over the set of instances maintained in the sources.

The significance of the DL-Lite family is testified by the fact that it constitutes the logical underpinning of OWL 2 QL, one of the three profiles of OWL 2. According to the official W3C profiles, the purpose of OWL 2 QL is to be language of choice for those applications that use very large amounts of data, and where query answering is among the most important reasoning tasks.

DL-Lite $\rho$. I now present the syntax of DL-Lite $\rho$, a logic of the DL-Lite family, which extends the core language of the DL-Lite family with the ability of specifying inclusion assertions between roles. In the following sections of this paper, I will adopt the syntax and semantics of the DL-Lite $\rho$ presented here, which is an extension of the traditional DL-Lite $\rho$ syntax with qualified existential cardinality restrictions over roles. Concepts and roles in DL-Lite $\rho$ are formed according to the following syntax:

$$B \rightarrow A | \exists Q \quad Q \rightarrow P | P^-$$
$$C \rightarrow B | \neg B | \exists Q.A \quad R \rightarrow Q | \neg Q$$

Here $A$ and $P$ denote respectively an atomic concept and an atomic role, $P^-$ denotes the inverse of $P$; $\neg$ denotes negation, $\exists Q$, also called unqualified existential role, denotes the set of objects related to some object by the role $Q$; the concept $\exists Q.A$, or qualified existential role, denotes the qualified domain of $Q$ with respect to $A$, i.e., the set of objects that $Q$ relates to some instance of $A$; $B$ and $Q$ denote respectively a basic concept and a basic role; and $C$ and $R$ denote respectively a general concept and a general role.

A DL-Lite $\rho$ TBox $T$ is a finite set of axioms of the form:

$$B \subseteq C \quad Q \subseteq R$$

The above axioms, from left to right, denote subsumptions between concepts and between roles, respectively. We call positive inclusions axioms of the form $B_1 \subseteq B_2$, $B_1 \subseteq \exists Q.A$, and $Q_1 \subseteq Q_2$, and negative inclusions axioms of the form $B_1 \subseteq \neg B_2$ and $Q_1 \subseteq \neg Q_2$. Notice however that such DL allows for specifying inclusion assertions between concepts, inclusion assertions between roles, typings of roles, disjointness between concepts or roles, and mandatory participation to roles. The formal semantics of DL-Lite $\rho$ ontologies and TBoxes is given in the standard way [1].

5. INTENTIONAL REASONING FOR OBDA APPLICATIONS

Much attention has been placed in recent studies on the production of efficient implementation algorithms for various intentional and extensional reasoning services in different DLs, such as conjunctive query answering for DLs of the DL-Lite family, and deductive closure of the intensional level of an ontology. In fact, intentional and extensional reasoning tasks such as instance-checking on OWL-based ontologies are currently offered by optimized automated reasoning services such as Hermit, Pellet, and FaCT++.

However, these results are still partial, as evidenced by the fact that none of these systems fully supports conjunctive query answering, and the fluctuating performances guaranteed by these tools when carrying out services such as the classification of TBoxes with varying levels of expressiveness indicate that these processes are still far from optimal, and that there is still room for improvement.

Therefore, one of the objectives that I am pursuing is to explore new solutions for efficient implementation of ontological reasoning services in the context of OBDA applications, with a focus on intensional level reasoning, and specifically on the classification of a TBox, and to attempt to adopt these solutions in other reasoning tasks such as logical implication.

Ontology classification. Ontology classification is the problem of computing all subsumption relationships inferred in an ontology between concept and property (i.e., role and attribute) names in the ontology signature. It is considered a core service for ontology reasoning, which can be exploited for various tasks, at both design-time and run-time, ranging from ontology navigation and visualization to query answering. Besides being interesting per se, efficient ontology classification can also be crucial for query answering, which can exploit such classification, as for example hap-
pens in the Presto algorithm for query answering in DL-Lite [30], currently implemented in the DL-Lite reasoner QUONTO \(^2\) at the core of the Mastro system.

Designing efficient methods and implementations for ontology classification is a particularly challenging task, since classification is in general a costly operation. Most popular reasoners for OWL ontologies, such as Pellet [34], FaCT++ [35], Hermit [33], and Racer [17] offer highly optimized classification services for expressive DLs, and have reached very good performances through the years. However, these reasoners are still not able to efficiently classify very large ontologies such as GALLEN [29] or FMA [15]. Other reasoners such as ELK [21], JCEL [26], and the CB reasoner [20] are tailored to intensional reasoning over specific DL fragments, i.e., logics of the EL family, the logical underpinning of OWL 2 EL, for ELK and JCEL, and the Horn-SHIQDL for the CB reasoner. These tools show excellent performance in classification of ontologies specified in such languages, but their implementation is rather specific for such ontologies. Instead, to the best of my knowledge, no techniques have been developed that are specifically tailored to intensional reasoning in OWL 2 QL, the “data oriented” profile of OWL 2, nor for any logic of the DL-Lite family [8], which constitutes the logical underpinning of OWL 2 QL.

My studies on this topic have thus provided a new technique for ontology classification in DL-Lite. Note that while in the following I refer to DL-Lite, the algorithms and implementations can be easily adapted to all logics of the DL-Lite family and to OWL 2 QL. The basic idea behind this technique is to encode the ontology terminology (TBox) into a directed graph, or digraph, and to compute the transitive closure of the graph through graph reachability in order to then obtain the classification of the ontology.

The analogy between inference rules in DLs and graph reachability is quite easy to grasp. For example, consider an ontology containing subsumptions \(A_1 \subseteq A_2\) and \(A_2 \subseteq A_3\), where \(A_1\), \(A_2\), and \(A_3\) are concept names in the ontology signature. Associate to this ontology a graph having three nodes labeled \(A_1\), \(A_2\), and \(A_3\), respectively, an edge from \(A_1\) to \(A_2\), and an edge from \(A_2\) to \(A_3\). It is straightforward to see that \(A_3\) is reachable from \(A_1\), and therefore an edge from \(A_1\) to \(A_3\) is contained in the transitive closure of the graph. This corresponds to the inferred subsumption \(A_1 \subseteq A_3\). However, things become much more complicated when complex concepts, such as those in the left-hand side of a subsumption that feature conjunction, come into play. Nevertheless, in this respect it can be demonstrated that for a DL-Lite ontology it is possible to easily construct a graph whose closure constitutes the major sub-task in ontology classification, because it allows to obtain all subsumptions inferred by the “positive knowledge” of the TBox.

Following these ideas, the classification technique can be divided into two steps. It first concentrates solely on the computation of the set of inclusions that are inferred by the positive portion of the ontology, denoted by \(\Phi_T\), not considering those inclusions that feature negation on the right-hand side. This first step is sufficient to obtain all the “non-trivial inclusions” inferred by the ontology, i.e., those subsumptions that do not have an unsatisfiable concept or role in their left-hand side. Secondly, the technique is refined to compute the set of missing subsumptions implied by unsatisfiable predicates of the ontology, denoted by \(\Omega_T\), in order to achieve soundness and completeness with respect to the problem of ontolgy classification. In the following we describe these two steps in further detail.

**Computation of \(\Phi_T\).** Given a DL-Lite\(_R\) TBox \(T\), in order to compute \(\Phi_T\), the set of positive inclusions in \(T\) is encoded into a digraph \(G_T\) and the transitive closure of \(G_T\) is computed in such a way that each subsumption \(S_1 \sqsubseteq S_2\) in \(\Phi_T\), where \(S_1\) and \(S_2\) are two basic concepts or two basic roles, corresponds to an arc \((S_1, S_2)\) in such transitive closure, and vice versa. The following constructive definition describes the appropriate manner to obtain the digraph TBox representation for this aim.

**Definition 1.** Let \(T\) be a DL-Lite\(_R\) TBox over a signature \(\Sigma\). We call the digraph representation of \(T\) the digraph \(G_T = (\Sigma', E')\) built as follows:

1. for each atomic concept \(A\) in \(\Sigma\), \(\mathcal{N}\) contains the node \(A\);
2. for each atomic role \(P\) in \(\Sigma\), \(\mathcal{N}\) contains the nodes \(P\), \(P^{-}\), \(\exists P\), \(\exists P^{-}\);
3. for each concept inclusion \(B_1 \sqsubseteq B_2\) in \(T\), \(E\) contains the arc \((B_1, B_2)\);
4. for each role inclusion \(Q_1 \sqsubseteq Q_2\) in \(T\), \(E\) contains the arches \((Q_1, Q_2), (Q_1, Q_2^\top), (\exists Q_1, \exists Q_2), (\exists Q_1^\top, \exists Q_2^\top)\);
5. for each concept inclusion \(B_1 \sqsubseteq \exists Q.A\) in \(T\), \(E\) contains the arc \((B_1, \exists Q)\).

The general idea is that each node in the digraph \(G_T\) represents a basic concept or a basic role, and each arc models a positive inclusion, i.e., a subsumption, contained in \(T\). The other node of the arc represents the left-hand side of the subsumption and the target node the right-hand side. We then denote with \(G_T = (\Sigma', E')\) the transitive closure of \(G_T\).

**Theorem 1.** Let \(T\) be a DL-Lite\(_R\) TBox and let \(G_T = (\Sigma', E')\) be its digraph representation. Let \(S_1\) and \(S_2\) be two atomic concepts, two atomic roles, or two atomic attributes. An inclusion assertion \(S_1 \sqsubseteq S_2\) belongs to \(\Phi_T\) if and only if there exists in \(\alpha(E')\) an arc \((S_1, S_2)\).

By exploiting Theorem 1, given a DL-Lite\(_R\) TBox \(T\), the algorithm for computing \(\Phi_T\) consists of first building the digraph \(G_T = (\Sigma', E')\) according to Definition 1, then computing its transitive closure, and finally building \(\Phi_T\) by adding an inclusion assertion \(S_1 \sqsubseteq S_2\) for each arc \((S_1, S_2)\) in \(\alpha(E')\).

**Computation of \(\Omega_T\).** While the presence of unsatisfiable predicates in an ontology is mainly due to errors in the design, it is however not rare to find such predicates, especially in very large ontologies, or in ontologies that are still “under construction”. Therefore it is necessary to consider the subsumptions that are inferred by such predicates, and to this aim I have developed the computeUnsat algorithm, which exploits the transitive closure of the graph and computes all unsatisfiable predicates for ontologies in DL-Lite. The basic idea behind the algorithm is the following.

Let \(S\) be either a concept expression or a role expression. Given a digraph \(G_T = (\Sigma', E')\) and a node \(n\) in \(\Sigma\), the set of predecessors \(\mathbf{predecessors}(n, G_T)\) contains all those nodes \(n'\) in \(\Sigma\) such that \(G_T\) contains the arc \((n', n)\), which means that there exists a path from \(n'\) to \(n\) in \(G\). We have that for each \(S'\) in \(\mathbf{predecessors}(S, G_T)\), the TBox \(T\) entails \(S' \sqsubseteq S\). Hence, given a negative inclusion assertion \(S_1 \sqsubseteq \neg S_2\), for each \(S_1'\) in \(\mathbf{predecessors}(S_1, G_T)\) and for each \(S_2'\) in \(\mathbf{predecessors}(S_2, G_T)\), \(T \models S_1' \sqsubseteq \neg S_2'\). Therefore, for each negative inclusion \(S_1 \sqsubseteq \neg S_2\) in \(T\) the algorithm computes the set of predecessors \((S_1, G_T)\) and \((S_2, G_T)\) and is able to

---

\(^2\)http://www.dis.uniroma1.it/quonto/
recognize as unsatisfiable all those concepts and roles whose node occurs in both the sets. For a DL-Lite TBox, it can be proven that an atomic concept or atomic role is unsatisfiable if and only if it is contained in the set of unsatisfiable predicates returned by computeUnsat.

Implementation and evaluation. An OWL 2 QL-tailored version of this new technique for classification has been implemented as a new module of the QuOnto reasoner, and an extensive experimentation phase as been carried out, with a specific focus on very large ontologies. The ontologies that have been considered are well-known, typical benchmarks for ontology classification, and have been suitably approximated to OWL 2 QL. These results are summarized in Figure 1. Timeout was set at one hour and all times are in milliseconds.

By virtue of its ability to exploit peculiar characteristics of OWL 2 QL, QuOnto showed better performance on these ontologies, in many cases corresponding to enormous gains, with respect to other tableau-based reasoners such as Pellet, FaCT++, and HermiT. The CB reasoner, which in general is the only reasoner which displays comparable performances to QuOnto for ontologies without negative inclusions, is faster for the EL-Galen and the Galen ontologies, but does not always perform complete classification. For instance, it does not compute property hierarchy.

This version of the algorithm for ontology classification and the results that have been achieved after extensive evaluation have been submitted [23] and are available from the authors.

I am currently working to extend this technique to compute all inclusions that are inferred by the TBox, i.e. the deductive closure, which, in DL-Lite is finite. In this regard, note that through \( \Gamma \models \) it is already possible to obtain the classification of all basic concepts, basic roles, and attributes, and that, with slight modifications of computeUnsat, it is possible to obtain the set of all negative inclusions inferred by the TBox. The remaining challenge is then to devise an efficient mechanism to obtain all inferred positive inclusions involving qualified existential roles and attribute domains.

Logical implication. Supported by these encouraging results, a further objective is to extend this graph-based technique to other intensional reasoning services, and in particular to logical implication, which allows to determine whether a given assertion \( \alpha \), e.g. a positive or negative inclusion, is a logical consequence of the TBox, i.e. if \( T \models \alpha \).

Preliminary studies on this issue are being carried out by exploring two different directions: on one hand, the development of DL-Lite-specific techniques for logical implication which do not require deductive closure of the ontology, and on the other hand, thanks to the performance of the graph-based classification algorithm for even the largest tested ontologies, the study of solutions for logical implication that take advantage of the graph-based representation of the TBox and the transitive closure of the di-graph.

### 6. A GRAPHICAL LANGUAGE FOR ONTOLOGIES

A crucial aspect when adopting the OBDA approach, and specifically when dealing with large-scale domains in real-life scenarios, is the ability of both domain experts and ontology designers to understand the domain of interest and to be able to communicate about it in an efficient way. As a standardized reference model for the domain, an ontology is an important first step towards this goal. However, there are still obstacles that must be overcome in order to reach maximum efficiency. These obstacles are of various nature: on one hand, generally domain experts do not possess the necessary skills to interpret the syntactic forms through which ontologies are typically expressed, whether it be an RDF/XML syntax, a functional syntax, or a set inclusion assertions; on the other hand, the sheer size of the domain makes it difficult for both domain experts and ontology designers to handle the ontology. Given these difficulties, it seems natural to look for an intuitive and compact solution for the representation of an ontology.

While much attention has been dedicated to the issue of ontology visualization, such a solution, when dealing with ontologies of real-life domains, is more difficult to find. In fact, generally researchers tend to use Protégé [13] for the creation and the visualization of ontologies. Natively Protégé offers a tree view for the visualization of ontology hierarchies. However, there are several different tools, such as OntoGraf [11] and OWLViz [19], available for Protégé, which can be used for visualizing and navigating both hierarchies and relationships. There have also been efforts towards the definition of a diagrammatic version of DLs through different diagrammatic reasoning systems such as spider and constraint diagrams and existential and conceptual graphs [10].

None of these alternatives however represent an ideal solution when dealing with ontologies that can feature hundreds of classes and properties. In fact, while the above mentioned tools for Protégé can be effective for the visualization of smaller ontologies or of specific hierarchies inside an ontology, the presentation of entire ontologies, or even of substantial portions of large ontologies, is extremely cluttered. This is also a critical issue for the diagrammatic versions of DLs, whose aim is not the provide the user with a global view of an entire ontology. Furthermore, the roles that relate the concepts in the ontology are not clearly presented to the user, and the use of directed edges for their representation lacks the necessary expressive power.

Therefore, in the earlier stages of our work in OBDA projects, the tool that was chosen for this task was the Entity-Relationship diagram. While such diagrams succeed in providing a compact and fairly intuitive representation of the domain, it became almost im-

<table>
<thead>
<tr>
<th>Ontology</th>
<th>QuOnto</th>
<th>FaCT++</th>
<th>HermiT</th>
<th>Pellet</th>
<th>CB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mouse</td>
<td>0.156</td>
<td>0.282</td>
<td>0.296</td>
<td>0.179</td>
<td>0.159</td>
</tr>
<tr>
<td>Transportation</td>
<td>0.015</td>
<td>0.045</td>
<td>0.163</td>
<td>0.151</td>
<td>0.195</td>
</tr>
<tr>
<td>DOLCE</td>
<td>1.327</td>
<td>0.245</td>
<td>25.619</td>
<td>1.696</td>
<td>1.358</td>
</tr>
<tr>
<td>AEO</td>
<td>0.650</td>
<td>0.745</td>
<td>0.920</td>
<td>0.647</td>
<td>0.605</td>
</tr>
<tr>
<td>Gene</td>
<td>1.255</td>
<td>1.400</td>
<td>3.810</td>
<td>2.803</td>
<td>1.918</td>
</tr>
<tr>
<td>EL-Galen</td>
<td>2.788</td>
<td>109.855</td>
<td>7.966</td>
<td>50.770</td>
<td>2.446</td>
</tr>
<tr>
<td>Galen</td>
<td>4.600</td>
<td>145.485</td>
<td>34.608</td>
<td>timeout</td>
<td>2.505</td>
</tr>
<tr>
<td>FMA 1.4</td>
<td>0.688</td>
<td>'timeout'</td>
<td>93.781</td>
<td>timeout</td>
<td>1.243</td>
</tr>
<tr>
<td>FMA 2.0</td>
<td>4.111</td>
<td>out of memory</td>
<td>out of memory</td>
<td>timeout</td>
<td>7.142</td>
</tr>
<tr>
<td>FMA 3.2.1</td>
<td>4.146</td>
<td>4.576</td>
<td>11.518</td>
<td>24.117</td>
<td>4.976</td>
</tr>
<tr>
<td>FMA-OBO</td>
<td>4.827</td>
<td>'timeout'</td>
<td>50.842</td>
<td>16.852</td>
<td>7.433</td>
</tr>
</tbody>
</table>

Figure 1: Classification times of OWL 2 QL ontologies.
mediately clear that they are unable to represent the full scope of expressions that can be modeled by the ontology language typically used in OBDA, i.e., DL-Lite. A clear example of these shortcomings, for instance, is the impossibility to distinguish the first from the second component of a role, i.e., distinguish $\exists P$ from $\exists P^\bot$ for a role $P$, as well as the impossibility to express qualified existential restrictions, i.e., those existential restrictions on roles where the restriction class is not the top concept. Because these existing solutions were deemed unsatisfactory, the decision was made to formalize a graphical language that is specifically tailored towards the representation of ontologies in DL-Lite.

The goal of this graphical language is to allow the ontology designer to create diagrams that have a graph-like structure, similar to that of an Entity-Relationship model, determined by a set of terminal symbols for concepts, roles, and attributes (for those DL-Lite dialects that allow for attribute specification), a set of non terminal symbols for conjunctions, disjunctions, etc., and a set of relations between elements of these two sets. This structure is represented by simple graphical elements such as circles, diamonds, squares, and edges, in order to require minimum prerequisite knowledge on the part of the domain experts.

The basic principle that guides this graphical formalization is the following: each graphical element in the diagram represents a specific term, expression, or assertion. In this context, three fundamental modeling choices control the construction of the diagram. Firstly, the terms that make up the alphabet of the ontology, i.e., those included in the signature $\Sigma$ are modeled by atomic graphical elements such as rectangles for atomic concepts, diamonds for atomic roles and circles for attributes. Secondly, an inclusion assertion in the ontology is modeled by a directed edge. Lastly, non-terminal symbols are used to create complex expressions such as property restrictions, conjunction (disjunction) of concepts, and concept hierarchies. For example, existential restrictions on a role $R$ and on its inverse are represented, respectively, by a white square and a black square, linked to the diamond representing $R$ and to the concept in the scope of the existential restriction by non-directed dotted edges. Figure 2 shows an example of qualified existential restriction in this graphical formalism. The diagram represents the following DL-Lite assertions:

1. $\text{County} \sqsubseteq \exists(is\text{PartOf}).\text{State}$
2. $\text{State} \sqsubseteq \exists(is\text{PartOf})^\bot.\text{County}$

Notice that in this example isPartOf is not typed on County and State, since we are assuming that it can be also used for relating different concepts.

Having defined the core elements of this language, which allow the ontology designer to capture the entire expressivity of DL-Lite, a natural evolution to this process is the expansion of the language to OWL, by utilizing the same graphical symbols, but extending their use to create more complex expressions, and by modeling different property restrictions such as cardinality and universality by using labels on the domain and range squares. This is currently under development.

In order to verify the validity of this graphical language, particularly when compared to other interactive visual tools for ontology navigation such as OntoGraf [11], an extensive evaluation will be performed, in which usability, functionality and acceptability of the system will be tested. These tests will involve user evaluation, through experimental, observational, and query methods.

**Scalability and modularization.** An important issue when dealing with the representation of ontologies, and specifically of large ones, is scalability. This is an aspect that must be considered by ontology designers in the perspective of user comprehension and of ontology manipulation by the designers. It is indeed unfeasible when dealing with very large ontologies that feature hundreds or thousands of concepts to include them all in a single graphical representation.

In the context of the projects in which I am involved, the solutions we are working towards pertain to the modularization of the ontologies. These solutions actually feature a two-dimensional modularization, both horizontal, by dividing the ontology into separate domains, and vertical, by singling out particularly complex areas of a domain and proposing various representations, each of growing detail. The end goal is to provide a visual representation of the ontology through various diagrams, each an instance of this graphical formalism. This allows to limit the dimensions of the more “general” representation of the domain, which highlights the aspects of the ontology that are common throughout the domain, in their most abstract form, and to produce other, more detailed diagrams of each specific and independent sub-domain.

The benefits of this solution go beyond simply the visual advantage of dealing with smaller diagrams (both for ontology engineers and final users): there is also the issue of user comprehension that is addressed. In fact, it is quite common, particularly when working with large companies, to come across users whose expertise and knowledge is very specific, pertaining perhaps even to a single and well-defined area or sub-domain, of the entire domain of interest. In such occurrences, the modularization of the ontology and the definition of several different and specific views on the ontology itself allows for smoother communication between the designer and the domain expert, and for a lesser burden on the system user, who is called upon to deal only with the ontological formalization of a sub-domain (or domains) with which he is familiar. Furthermore, the mode abstract and “general” view of the ontology allows on the other hand to illustrate the main modeling choices that are made by the designer, and to immediately pinpoint the perimeter of the domain of interest that is represented by the ontology.

**Visualization.** To further aid user comprehension of ontologies expressed through this graphical language, I am also contributing to preliminary studies that are being conducted on efficient visualization techniques for large-scale ontologies that are formalized through this representation. The direction in which these studies are proceeding is the investigation of techniques for the identification of the “relevant context” of a concept or of a portion of the domain in the ontology, and the design of a dynamic model for the
visualization of the ontology. While a static, two-dimensional representation may represent a valid solution for small ontologies, it loses effectiveness when dealing with ontologies that model large domains, because the user may have to navigate through large portions of the diagram in order to identify the information he is seeking. Therefore, the goal of this model is to effectively identify, group together, and highlight all the relevant concepts and roles in a specific portion of the ontology, while moving the remaining information into the background. This layout solution can provide a more intelligent use of the available display space, and allow a smoother navigation of the ontology by the user.

7. ONTOLOGY APPROXIMATION

The extension of this graphical language to OWL is beneficial for the ontology designer because it allows the formalization in the diagram of assertions that go beyond the expressive abilities of data-oriented DLs. However, a crucial requirement of OBDA is that query answering for unions of conjunctive queries (UCQs) can be reduced to the evaluation of a first-order query (directly translatable into SQL) over a database. For this reason, the languages through which ontologies in OBDA are typically expressed are the members of the DL-Lite family of description logics which allow for this property. Therefore, in order to fulfill the OBDA requirement of efficiently accessing large data bases, it is necessary to approximate such expressive ontologies by means of less expressive ones.

The goal of an approximation is to capture as much as possible of the semantics of the original ontology. Common syntactic approximations only consider the syntactic form of the axioms that are to be approximated, disregarding those axioms which are not compliant with the syntax of the ontology language by means of which the original ontology is to be approximated [36]. While this approximation can generally be executed by fast and simple algorithms, it does not, in general, guarantee soundness, i.e. to not imply additional unwanted inferences, or completeness, which guarantees that all entailments of the original ontology that are also expressible in the target language are preserved.

Therefore, the semantic approach to ontology approximation poses a more interesting but more complex challenge [27]. The main issue with this approach is that, while providing better guarantees w.r.t. soundness and completeness of the result, it often necessitates to perform a complete classification, and therefore tends to be significantly slower. The objective that I will pursue regarding this issue will then be to develop an algorithm for semantic approximation that guarantees a reasonable trade-off between efficiency and expressiveness of the approximated ontology. In this respect, since the target language is DL-Lite, the basic idea of the approach is to treat each OWL axiom \( \alpha \) of the original ontology in isolation, and compute, through the use of an OWL reasoner, all DL-Lite axioms constructible over the signature of \( \alpha \) that are inferred by \( \alpha \). A challenging question is then how to deal with the loss of knowledge, i.e., OWL knowledge that cannot be captured. A possible choice is to adopt a further approximation that consists in interpreting some OWL axioms through a different semantics, which is weaker than the standard (FOL) one, in the line of the studies in [6]. This is a task that I am currently investigating.

8. VALIDATION THROUGH PROJECT-RELATED ACTIVITY AND OTHER ISSUES

The aspects that I have discussed in this paper are critical when working with ontologies in real-life OBDA projects. In fact, the need for these tools and methodologies often times is felt only once one has had the chance to undertake this activity, such as the one in which I am currently involved with the Telecom Italia telecommunications company, that leads to dealing with issues that are typical of big data, i.e. data storing, handling, and access.

It is in such contexts that one can appreciate the need for integrated and ontology-based access to data through systems such as Mastro. Furthermore, the instruments that have been discussed in this paper can also be helpful when re-engineering an organization’s information system. In this scenario, an ontology proves to be a valid tool for the design specification of the new system and its data repositories, and it is therefore crucial to be able to rely on the support of tailor-made tools for its representation.

From a practical standpoint, the construction of an OBDA framework in real-life settings poses significant problems in terms of content handling, particularly in the early stages of system development, i.e. during ontology design and documentation. These issues, while not of strong theoretical relevance, determine significant costs in terms of time and effort when not handled within the guidelines of a standardized methodology. It has in fact become apparent that the alignment between ontology and project documentation must be handled in an automated way, through tools that are able to extract information from the ontology, and to generate at least a preliminary documentation. This automated approach is especially significant, and beneficial for both developers and users, when ontology design and domain research are being performed in parallel: it allows the system to automatically reflect, in the documentation, the changes that are made in the modeling of the ontology. Ideally, a work-plan would divide domain research, ontology design, and documentation into different phases of the project. However this division may not always be feasible in practice, particularly when working with large-scale ontologies. In this context, I am contributing to the integration of the tools that are being developed by myself and others to solve these issues into the Mastro system platform, in order to provide the user with an instrument which offers full support for all OBDA-related activities.

Lastly, experience with the design of ontologies that formalize real-world domains has provided the opportunity to identify aspects of domain modeling that commonly occur in different scenarios, and that must be routinely dealt with. It seems natural then to attempt to study these recurring aspects, such as temporally changing information or part-whole relations, and to identify patterns for effectively modeling them. Therefore, some preliminary work is being carried out in this direction, in the hopes of defining solutions that can be applied successfully in both bottom-up or top-down ontology-design approaches.

9. RESULTS AND BENEFITS

There are numerous benefits that may result from the work that I will carry out during my Ph.D. program, both through the improvement of ontological reasoning services and through the development of methodologies and tools for the tasks which are performed in OBDA settings. Intensional reasoning services, such as the classification of the TBox, can aid in checking that ontology design and definition is performed correctly, and that the assertions that compose the TBox do not generate unsatisfiable elements of the ontology. On the other hand, improvements in key aspects of the OBDA framework such as ontology design and visualization will undoubtedly help to advance the OBDA approach towards the fulfillment of its potential in real-life settings. The benefits that will result from these improvements can be found in areas such as the design and re-engineering of information systems, the optimization of user access to information stored in different data sources, and in
10. REFERENCES


