A graph-based approach for classifying OWL 2 QL ontologies*

Domenico Lembo, Valerio Santarelli, and Domenico Fabio Savo

Dipartimento di Ing. Informatica, Automatica e Gestionale “Antonio Ruberti”
Sapienza Università di Roma
Via Ariosto 25, I-00186 Roma, Italy
{lembo,santarelli,savo}@dis.uniroma1.it

Abstract. Ontology classification is the reasoning service that computes all subsumption relationships inferred in an ontology between concept, role, and attribute names in the ontology signature. OWL 2 QL is a tractable profile of OWL 2 for which ontology classification is polynomial in the size of the ontology TBox. However, to date, no efficient methods and implementations specifically tailored to OWL 2 QL ontologies have been developed. In this paper, we provide a new algorithm for ontology classification in OWL 2 QL, which is based on the idea of encoding the ontology TBox into a directed graph and reducing core reasoning to computation of the transitive closure of the graph. We have implemented the algorithm in the QuOnto reasoner and extensively evaluated it over very large ontologies. Our experiments show that QuOnto outperforms various popular reasoners in classification of OWL 2 QL ontologies.

1 Introduction

Ontology classification is the problem of computing all subsumption relationships inferred in an ontology between predicate names in the ontology signature, i.e., named concepts (a.k.a. classes), roles (a.k.a. object-properties), and attributes (a.k.a. data-properties). 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.

Devising efficient ontology classification methods and implementations is a challenging issue, since classification is in general a costly operation. Most popular reasoners for Description Logic (DL) ontologies, i.e., OWL ontologies, such as Pellet [23], Racer [11], FACT++ [24], and HermiT [9], offer highly optimized classification services for expressive DLs. Various experimental studies show that such reasoners have reached very good performances through the years. However, they are still not able to efficiently classify very large ontologies, such as the full versions of GALEN [22] or of the FMA ontology [10].

Whereas the above tools use algorithms based on model construction through tableau (or hyper-tableau [9]), the CB reasoner [14] for the Horn-$\mathcal{SHIQ}$ DL is

* This paper is an extended abstract of [18].

a *consequence-driven* reasoner. The use of this technique allows CB to obtain an impressive gain on very large ontologies, such as full GALEN. However, the current implementation of the CB reasoner is rather specific for particular fragments of Horn-$\mathcal{SHIQ}$ (and incomplete for the general case) [14]. For example, it does not allow for classification of properties.

Other recently developed tools, such as Snorocket [17], ELK [15], and JCEL [20], are specifically tailored to intensional reasoning over logics of the $\mathcal{EL}$ family, and show excellent performances in classification of ontologies specified in such languages, which are the logical underpinning of OWL 2 EL, one of the tractable profile of OWL 2 [21].

Instead, to the best of our knowledge, ontology classification in the other OWL 2 profiles has received so far little attention. In particular, classification in OWL 2 RL has been investigated only in [16], whereas, to date, 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 [7][1], which constitutes the logical underpinning of OWL 2 QL. Our aim is then to contribute to fill this lack on OWL 2 QL, encouraged also by the fact that such language, like all logics of the *DL-Lite* family, allows for tractable intensional reasoning, and in particular for PTIME ontology classification, as it immediately follows from the results in [7].

In this paper, we thus provide a new method for ontology classification in the OWL 2 QL profile. In our technique, we encode the ontology terminology (TBox) into a graph, and compute the transitive closure of the graph to then obtain the ontology classification. The analogy between simple inference rules in DLs and graph reachability is indeed very natural: consider, for example, an ontology containing the subsumptions $A_1 \sqsubseteq A_2$ and $A_2 \sqsubseteq A_3$, where A_1, A_2, and A_3 are class names in the ontology signature. We can then associate to this ontology a graph having three nodes labeled with 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 $A_1 \sqsubseteq A_3$. On the other hand, things become soon much more complicated when complex (OWL) axioms come into play.

In this respect, we will show that for an OWL 2 QL ontology it is possible to easily construct a graph whose closure constitutes the major sub-task in ontology classification, because it allows us to obtain all subsumptions inferred by the "positive knowledge" specified by the TBox. We will show that the computed classification misses only "trivial" subsumptions inferred by unsatisfiable predicates, i.e., named classes (resp. properties) that always have an empty interpretation in every model of the ontology, and that are therefore subsumed by every class (resp. property) in the ontology signature. We therefore provide an algorithm that, exploiting the transitive closure of the graph, computes all unsatisfiable predicates, thus allowing us to obtain a complete ontology classi-

[1] Not to be confused with the set of DLs studied in [2], which form the *DL-Lite*$_{bool}$ family.

fication. We notice that the presence of unsatisfiable predicates in an ontology is mainly due to errors in the design. However, it is not rare to find such predicates, especially in very large ontologies or in ontologies that are still "under construction". In particular, we could find unsatisfiable concepts even in some benchmark ontologies we used in our experiments (cf. Section 4). Of course, already debugged ontologies might not present such predicates [13,12]. In this case, one can avoid executing our algorithm for computing unsatisfiable predicates.

We have implemented our technique in a new module of QuOnto [1], the reasoner at the base of the Mastro [6,8] system, and have carried out extensive experimentation, focusing in particular on very large ontologies. We have considered a number of well-known ontologies, often used as benchmark for ontology classification, and have suitably approximated in OWL 2 QL those that are out of this language.

QuOnto showed better performances, in some cases corresponding to enormous gains, with respect to tableau-based reasoners (in particular, Pellet, Fact++, and HermiT). We also obtained comparable or better results with respect to the CB reasoner, for almost all ontologies considered, but, differently from CB reasoner, we were always able to compute a complete classification. We finally compared QuOnto with ELK, one of the most performing reasoner for $\mathcal{EL}$, for those approximated ontologies that turned out to be both in OWL 2 QL and OWL 2 EL, obtaining similar performances in almost all cases.

We conclude by noticing that, even though we refer here to OWL 2 QL, our algorithms and implementations can be easily adapted to deal with all logics of the *DL-Lite* family mentioned in [7], excluding those allowing for the use of conjunction in the left-hand side of inclusion assertions or the use of n-ary relations instead of binary roles.

The rest of the paper is organized as follows. In Section 2, we provide some preliminaries. In Section 3, we describe our technique for ontology classification in OWL 2 QL. In Section 4, we describe our experimentation, and finally, in Section 5, we conclude the paper.

2 Preliminaries

In this section, we present some basic notions on DL ontologies, the formal underpinning of the OWL 2 language, and on OWL 2 QL. We also recall some notions of graph theory needed later on.

Description Logic Ontologies. We consider a signature Σ, partitioned in two disjoint signatures, namely, Σ_P, containing symbols for predicates, i.e., atomic concepts, atomic roles, atomic attributes, and value-domains, and Σ_C, containing symbols for individual (object and value) constants. Complex concept, role, and attribute expressions are constructed starting from predicates of Σ_P by applying suitable constructs, which vary in different DL languages. Given a DL language $\mathcal{L}$, an $\mathcal{L}$-TBox (or simply a TBox, when $\mathcal{L}$ is clear) over Σ contains universally quantified first-order (FOL) assertions, i.e., axioms specifying general properties of concepts, roles, and attributes. Again, different DLs allow for different axioms.

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. An $\mathcal{L}$-ontology $\mathcal{O}$ is constituted by both an $\mathcal{L}$-TBox $\mathcal{T}$ and an $\mathcal{L}$-ABox $\mathcal{A}$, denoted as $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$.

The semantics of a DL ontology $\mathcal{O}$ is given in terms of FOL interpretations (cf. [4]). We denote with $Mod(\mathcal{O})$ the set of models of $\mathcal{O}$, i.e., the set of FOL-interpretations that satisfy all TBox axioms and ABox assertions in $\mathcal{O}$, where the definition of satisfaction depends on the DL language in which $\mathcal{O}$ is specified. An ontology $\mathcal{O}$ is *satisfiable* if $Mod(\mathcal{O}) \neq \emptyset$. A FOL-sentence ϕ is *entailed* by an ontology $\mathcal{O}$, denoted $\mathcal{O} \models \phi$, if ϕ is satisfied by every model in $Mod(\mathcal{O})$. All the above notions naturally apply to a TBox $\mathcal{T}$ alone.

Traditional intensional reasoning tasks with respect to a given TBox are verification of subsumption and satisfiability of concepts, roles, and attributes [4]. More precisely, a concept C_1 is *subsumed in* $\mathcal{T}$ by a concept C_2, written $\mathcal{T} \models C_1 \sqsubseteq C_2$, if, in every model I of $\mathcal{T}$, the interpretation of C_1, denoted C_1^I, is contained in the interpretation of C_2, denoted C_2^I, i.e., $C_1^I \subseteq C_2^I$ for every $I \in Mod(\mathcal{T})$. Furthermore, a concept C in $\mathcal{T}$ is unsatisfiable, which we wrote as $\mathcal{T} \models C \sqsubseteq \neg C$, if the interpretation of C is empty in every model of $\mathcal{T}$, i.e., $C^I = \emptyset$ for every $I \in Mod(\mathcal{T})$. Analogous definitions hold for roles and attributes.

Strictly related to the previous reasoning tasks is the classification inference service, which we focus on in this paper. Given a signature Σ_P and a TBox $\mathcal{T}$ over Σ_P, such a service allows to determine subsumption relationships in $\mathcal{T}$ between concepts, roles, and attributes in Σ_P. Therefore, classification allows to structure the terminology of $\mathcal{T}$ in the form of a subsumption hierarchy that provides useful information on the connection between different terms, and can be used to speed up other inference services. Here we define it more formally.

Definition 1. *Let $\mathcal{T}$ be a satisfiable $\mathcal{L}$-TBox over Σ_P. We define the $\mathcal{T}$-classification of Σ_P (or simply $\mathcal{T}$-classification when Σ_P is clear from the context) as the set of inclusion assertions defined as follows:*
Let S_1 and S_2 be either two concepts, roles, or attributes in Σ_P. If $\mathcal{T} \models S_1 \sqsubseteq S_2$ then $S_1 \sqsubseteq S_2$ belongs to the $\mathcal{T}$-classification of Σ_P.

The OWL 2 QL Language. The OWL 2 QL language is based on *DL-Lite*$_R$, a DL of the *DL-Lite* family [7]. Differently from *DL-Lite*$_R$, however, besides object properties (i.e., roles), OWL 2 QL allows also for the use of data properties (i.e., attributes), as well as some further constructs, as (ir-)reflexivity on properties. For the sake of presentation, we prefer to not consider here attributes, nor (ir-)reflexivity constraints. This choice does not actually correspond to a real simplification, since in the algorithms proposed in this paper we can treat both attributes and roles essentially in the same way, and our techniques can be applied to full OWL 2 QL ontologies with minimal adaptations. Therefore, in the following, we provide a simplified, German style, syntax for OWL 2 QL, which actually corresponds to that of *DL-Lite*$_R$, whereas refer the reader to [21] for the complete, OWL functional-style syntax of this language[2].

[2] Notice that (a)symmetric roles allowed in OWL 2 QL, even though not explicitly mentioned, can be easily expressed in the syntax that we consider.

Expressions in OWL 2 QL are formed according to the following syntax:

$$\begin{array}{ll} B \longrightarrow A \mid \exists Q & \quad Q \longrightarrow P \mid P^- \\ C \longrightarrow B \mid \neg B \mid \exists Q.A & \quad R \longrightarrow Q \mid \neg Q \end{array}$$

where: A and P are symbols in Σ_P denoting respectively an *atomic concept* and an *atomic role*; P^- denotes the inverse of P; $\exists Q$, 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. In the following, we call B a *basic concept*, and Q a *basic role*.

An OWL 2 QL TBox $\mathcal{T}$ is a finite set of axioms of the form $B \sqsubseteq C$ and $Q \sqsubseteq R$, where the former denote subsumptions between concepts, and the latter subsumptions between roles. We call *positive inclusions* axioms of the form $B_1 \sqsubseteq B_2$, $B_1 \sqsubseteq \exists Q.A$, and $Q_1 \sqsubseteq Q_2$, and *negative inclusions* axioms of the form $B_1 \sqsubseteq \neg B_2$ and $Q_1 \sqsubseteq \neg Q_2$.

The semantics of OWL 2 QL ontologies and TBoxes is given in the standard way [21,4].

As for OWL 2 QL ABoxes, we do not present them here, since we concentrate on intensional reasoning, and refer the interested reader to [21].

Graph Theory Notions. In this paper we use the term *digraph* to refer to a directed graph. We assume that a digraph $\mathcal{G}$ is a pair $(\mathcal{N}, \mathcal{E})$, where $\mathcal{N}$ is a set of elements called *nodes*, and $\mathcal{E}$ is a set of ordered pairs (s, t) of nodes in $\mathcal{N}$, called *arcs*, where s is denoted the *source* of the arc, and t the *target* of the arc.

The transitive closure $\mathcal{G}^* = (\mathcal{N}, \mathcal{E}^*)$ of a digraph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is a digraph such that there is an arc in $\mathcal{E}^*$ having a node s as source and a node t as target if and only if there is a path from s to t in $\mathcal{G}$ [5]. Let $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a digraph, and let n be a node in $\mathcal{N}$. We denote with $\mathsf{predecessors}(n, \mathcal{G})$ the set of nodes p_n in $\mathcal{N}$ such that there exists in $\mathcal{E}$ an arc (p_n, n).

3 $\mathcal{T}$-classification in OWL 2 QL

In this section we describe our approach to computing, given a signature Σ_P and an OWL 2 QL TBox $\mathcal{T}$ over Σ_P, the $\mathcal{T}$-classification of Σ_P.

In OWL 2 QL, a subsumption relation between two concepts or roles in Σ_P, can be inferred by a TBox $\mathcal{T}$ if and only if (*i*) $\mathcal{T}$ contains such subsumption; (*ii*) $\mathcal{T}$ contains a set of positive inclusion assertions that together entail the subsumption; or (*iii*), trivially, the subsumed concept or role is unsatisfiable in $\mathcal{T}$. The above observation is formalized as follows.

Theorem 1. *Let $\mathcal{T}$ be an OWL 2 QL TBox containing only positive inclusions, and let S_1 and S_2 be two atomic concepts or two atomic roles. $S_1 \sqsubseteq S_2$ is entailed by $\mathcal{T}$ if and only if at least one of the following conditions holds:*

1. *a set $\mathcal{P}$ of positive inclusions exists in $\mathcal{T}$, such that $\mathcal{P} \models S_1 \sqsubseteq S_2$;*
2. *$\mathcal{T} \models S_1 \sqsubseteq \neg S_1$.*

Given a OWL 2 QL TBox $\mathcal{T}$ over a signature Σ_P, we use $\Phi_\mathcal{T}$ and $\Omega_\mathcal{T}$ to denote two sets of positive inclusions of the form $S_1 \sqsubseteq S_2$, with $S_1, S_2 \in \Sigma_P$, such that $\Phi_\mathcal{T}$ contains only positive inclusions for which statement 1 holds, and $\Omega_\mathcal{T}$ contains only positive inclusions for which statement 2 holds. It is easy to see that $\Phi_\mathcal{T}$ and $\Omega_\mathcal{T}$ are not disjoint. From Definition 1 and Theorem 1 it follows that the $\mathcal{T}$-classification coincides with the union of the sets $\Phi_\mathcal{T}$ and $\Omega_\mathcal{T}$.

In the following, we describe our approach to the computation of the $\mathcal{T}$-classification by firstly computing the set $\Phi_\mathcal{T}$, and then computing the set $\Omega_\mathcal{T}$.

Computation of $\Phi_\mathcal{T}$. Given an OWL 2 QL TBox $\mathcal{T}$, in order to compute $\Phi_\mathcal{T}$, we encode the set of positive inclusions in $\mathcal{T}$ into a digraph $\mathcal{G}_\mathcal{T}$ and compute the transitive closure of $\mathcal{G}_\mathcal{T}$ in such a way that each subsumption $S_1 \sqsubseteq S_2$ in $\Phi_\mathcal{T}$ corresponds to an arc (S_1, S_2) in such transitive closure, and vice versa. The following constructive definition describes the appropriate fashion to obtain the digraph TBox representation for our aims.

Definition 2. *Let $\mathcal{T}$ be an OWL 2 QL TBox over a signature Σ_P. We call the digraph representation of $\mathcal{T}$ the digraph $\mathcal{G}_\mathcal{T} = (\mathcal{N}, \mathcal{E})$ built as follows:*

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

The idea is that each node in the digraph $\mathcal{G}_\mathcal{T}$ represents a basic concept or a basic role, and each arc models a positive inclusion, i.e., a subsumption, contained in $\mathcal{T}$, where the source node of the arc represents the left-hand side of the subsumption and the target node of the arc represents the right-hand side of the subsumption. Observe that for each role inclusion assertion $P_1 \sqsubseteq P_2$ in the TBox $\mathcal{T}$, we also represent as nodes and arcs in the digraph $\mathcal{G}_\mathcal{T}$ the entailed positive inclusions $P_1^- \sqsubseteq P_2^-$, $\exists P_1 \sqsubseteq \exists P_2$, and $\exists P_1^- \sqsubseteq \exists P_2^-$.

Let $\mathcal{T}$ be an OWL 2 QL TBox and let $\mathcal{G}_\mathcal{T} = (\mathcal{N}, \mathcal{E})$ be its digraph representation. We denote with $\mathcal{G}_\mathcal{T}^* = (\mathcal{N}, \mathcal{E}^*)$ the transitive closure of $\mathcal{G}_\mathcal{T}$. Note that by definition of digraph transitive closure, for each node $n \in \mathcal{N}$ there exists in $\mathcal{E}^*$ an arc (n, n). Moreover, in what follows, we denote with $\alpha(\mathcal{E}^*)$ the set of arcs $(S_1, S_2) \in \mathcal{E}^*$ such that both terms S_1 and S_2 denote in $\mathcal{T}$ either two atomic concepts or two atomic roles. Then, the following property holds.

Theorem 2. *Let $\mathcal{T}$ be an OWL 2 QL TBox and let $\mathcal{G}_\mathcal{T} = (\mathcal{N}, \mathcal{E})$ be its digraph representation. Let S_1 and S_2 be two atomic concepts or two atomic roles. An inclusion assertion $S_1 \sqsubseteq S_2$ belongs to $\Phi_\mathcal{T}$ if and only if there exists in $\alpha(\mathcal{E}^*)$ an arc (S_1, S_2).*

We can then easily construct an algorithm, called ComputeΦ, that, taken as input an OWL 2 QL TBox $\mathcal{T}$, first builds the digraph $\mathcal{G}_\mathcal{T} = (\mathcal{N}, \mathcal{E})$ according

```
**Algorithm:** computeUnsat
**Input:** an OWL 2 QL TBox $\mathcal{T}$
**Output:** a set of concept and role expressions
$\mathsf{Emp} \leftarrow \emptyset$;
foreach negative inclusion $S_1 \sqsubseteq \neg S_2 \in \mathcal{T}$ do                /* step 1 */
    foreach $n_1 \in \mathsf{predecessors}(S_1, \mathcal{G}^*_{\mathcal{T}})$ do
        foreach $n_2 \in \mathsf{predecessors}(S_2, \mathcal{G}^*_{\mathcal{T}})$ do
            if $n_1 = n_2$
            then $\mathsf{Emp} \leftarrow \mathsf{Emp} \cup \{n_1\}$;
            if $(n_1 = \exists Q^-$ and $n_2 = A)$ or $(n_2 = \exists Q^-$ and $n_1 = A)$
            then $\mathsf{Emp} \leftarrow \mathsf{Emp} \cup \{\exists Q.A\}$;
$\mathsf{Emp}' \leftarrow \emptyset$;
while $\mathsf{Emp} \neq \mathsf{Emp}'$ do                                          /* step 2 */
    $\mathsf{Emp}' \leftarrow \mathsf{Emp}$;
    foreach $S \in \mathsf{Emp}'$ do
        foreach $n \in \mathsf{predecessors}(S, \mathcal{G}^*_{\mathcal{T}})$ do
            $\mathsf{Emp} \leftarrow \mathsf{Emp} \cup \{n\}$;
            if $n = P$ or $n = P^-$ or $n = \exists P$ or $n = \exists P^-$
            then $\mathsf{Emp} \leftarrow \mathsf{Emp} \cup \{P, P^-, \exists P, \exists P^-\}$;
            if there exists $B \sqsubseteq \exists Q.n \in \mathcal{T}$
            then $\mathsf{Emp} \leftarrow \mathsf{Emp} \cup \{\exists Q.n\}$;
return Emp.
```

Fig. 1: The algorithm computeUnsat($\mathcal{T}$)

to Definition 2, then computes its transitive closure, and finally returns the set $\Phi_{\mathcal{T}}$, which contains an inclusion assertion $S_1 \sqsubseteq S_2$ for each arc $(S_1, S_2) \in \alpha(\mathcal{E}^*)$.

According to Theorem 2, ComputeΦ is sound and complete with respect to the problem of computing $\Phi_{\mathcal{T}}$ for any OWL 2 QL TBox $\mathcal{T}$ containing only positive inclusions.

Computation of $\Omega_{\mathcal{T}}$. We first observe that, according Definition 2, no node corresponding to a qualified existential role is created in the TBox digraph representation. This kind of node is indeed not useful for computing $\Phi_{\mathcal{T}}$. Differently, if one aims to identify every cause of unsatisfiability, the creation of nodes corresponding to a qualified existential role is needed. This is due to the fact that a TBox may entail that a qualified existential role $\exists P.A$ is unsatisfiable, even in case of satisfiability of $\exists P$. Specifically, this may occur in two instances: (*i*) if the TBox $\mathcal{T}$ entails the assertion $\exists P^- \sqsubseteq \neg A$, and (*ii*), the TBox $\mathcal{T}$ entails $A \sqsubseteq \neg A$. Clearly, in both cases the concept $\exists P.A$ is unsatisfiable. We therefore modify here Definition 2 by substituting Rule 5 with the following one:

5^*. for each concept inclusion $B_1 \sqsubseteq \exists Q.A \in \mathcal{T}$, $\mathcal{N}$ contains the node $\exists Q.A$, and $\mathcal{E}$ contains the arcs $(B_1, \exists Q.A)$ and $(\exists Q.A, \exists Q)$;

From now on, we adopt the digraph representation built according to Definition 2, where rule 5^* replaces rule 5. Given one such TBox $\mathcal{T}$ over a signature Σ_P, the algorithm computeUnsat given in Figure 1 returns all unsatisfiable concepts and roles in Σ_P, by exploiting the transitive closure of the digraph representation of $\mathcal{T}$.

Before describing the algorithm, we recall that, given a digraph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ and a node $n \in \mathcal{N}$, the set $\mathsf{predecessors}(n, \mathcal{G}^*)$ contains all those nodes n' in $\mathcal{N}$ such that $\mathcal{G}^*$ contains the arc (n', n), which means that there exists a path from n' to n in $\mathcal{G}$. Also, it can be shown that $\mathcal{G}^*_{\mathcal{T}}$ allows in fact to obtain all subsumptions between satisfiable *basic* concepts or roles, in the sense that the TBox $\mathcal{T}$ infers one such subsumption $S_1 \sqsubseteq S_2$ if and only if there is an arc (S_1, S_2) in $\mathcal{E}^*$. Then, the two steps that compose the algorithm proceed as follows:

Step 1 Let S be either a concept expression or a role expression. We have that for each $S^i \in \mathsf{predecessors}(S, \mathcal{G}^*_{\mathcal{T}})$ the TBox $\mathcal{T}$ entails $S^i \sqsubseteq S$. Hence, given a negative inclusion assertion $S_1 \sqsubseteq \neg S_2$, for each $S_1^i \in \mathsf{predecessors}(S_1, \mathcal{G}^*_{\mathcal{T}})$ and for each $S_2^j \in \mathsf{predecessors}(S_2, \mathcal{G}^*_{\mathcal{T}})$, $\mathcal{T} \models S_1^i \sqsubseteq \neg S_2^j$. Therefore, for each negative inclusion $S_1 \sqsubseteq \neg S_2 \in \mathcal{T}$, the algorithm computes the set $\mathsf{predecessors}(S_1, \mathcal{G}^*_{\mathcal{T}})$ and $\mathsf{predecessors}(S_2, \mathcal{G}^*_{\mathcal{T}})$ and is able to: (i) recognize as unsatisfiable all those concepts and roles whose corresponding nodes occur in both the set $\mathsf{predecessors}(S_1, \mathcal{G}^*_{\mathcal{T}})$ and $\mathsf{predecessors}(S_2, \mathcal{G}^*_{\mathcal{T}})$, and (ii) identify those unsatisfiable qualified existential roles $\exists Q.A$ whose inverse existential role node $\exists Q^-$ occurs in $\mathsf{predecessors}(S_1, \mathcal{G}^*_{\mathcal{T}})$ (resp. $\mathsf{predecessors}(S_2, \mathcal{G}^*_{\mathcal{T}})$) and whose concept node A occurs in $\mathsf{predecessors}(S_2, \mathcal{G}^*_{\mathcal{T}})$ (resp. $\mathsf{predecessors}(S_1, \mathcal{G}^*_{\mathcal{T}})$), which indeed implies $\exists Q^- \sqsubseteq \neg A$ and therefore unsatisfiability of $\exists Q.A$.

Step 2 Further unsatisfiable concepts and roles are identified by the algorithm through a cycle in which: (i) if a concept or role S is in Emp, then all the expressions corresponding to the nodes in $\mathsf{predecessors}(S, \mathcal{G}^*_{\mathcal{T}})$ are in Emp. This captures propagation of unsatisfiability through chains of positive inclusions; (ii) if at least one of the expressions $P, P^-, \exists P, \exists P^-$ is in Emp, then all four expressions are in Emp; (iii) for each expression $\exists Q.A$ in $\mathcal{N}$, if $A \in \mathsf{Emp}$, then $\exists Q.A \in \mathsf{Emp}$. We notice that the algorithm stops cycling when no new expressions of the form $\exists Q$ or $\exists Q.A$ are added to Emp (indeed, in this case only a single further iteration may be needed).

It easy to see that, by virtue of the fact that the size of the set $\mathcal{N}$ of the digraph representation of the TBox $\mathcal{T}$ is finite, $\mathsf{computeUnsat}(\mathcal{T})$ terminates, and that the number of executions of the while cycle is less than or equal to $|\mathcal{N}|$.

The following theorem shows that algorithm $\mathsf{computeUnsat}$ can be used for computing the set containing all the unsatisfiable concepts and roles in $\mathcal{T}$.

Theorem 3. *Let $\mathcal{T}$ be an OWL 2 QL TBox and let S be either an atomic concept or an atomic role in Σ_P. $\mathcal{T} \models S \sqsubseteq \neg S$ if and only if $S \in \mathsf{computeUnsat}(\mathcal{T})$.*

We call $\mathsf{Compute\Omega}$ the algorithm that, taken $\mathcal{T}$ as input, returns $\Omega_{\mathcal{T}}$ by making use of $\mathsf{computeUnsat}$.

The following theorem, which is a direct consequence of Theorem 2 and of Theorem 3, states that our technique is sound and complete with respect to the problem of classifying an OWL 2 QL TBox.

Theorem 4. *Let $\mathcal{T}$ be an OWL 2 QL TBox and let S_1 and S_2 be either two atomic predicates. $\mathcal{T} \models S_1 \sqsubseteq S_2$ if and only if $S_1 \sqsubseteq S_2 \in \mathsf{Compute\Phi}(\mathcal{T}) \cup \mathsf{Compute\Omega}(\mathcal{T})$.*

Ontology	Concepts	Roles	Attributes	Original DL fragment	Original axioms	Owl 2 QL axioms	Negative inclusions
Mouse	2753	1	0	*ALE*	3463	3463	0
Transportation	445	89	4	*ALCH(D)*	931	931	317
DOLCE	209	313	4	*SHOIN(D)*	1736	1991	45
AEO	760	47	16	*SHIN(D)*	3449	3432	1957
Gene	26225	4	0	*SH*	42655	42655	3
EL-Galen	23136	950	0	*ELH*	46457	48026	0
Galen	23141	950	0	*ALEHIF+*	47407	49926	0
FMA 1.4	6488	165	0	*ALCOIF*	18612	18663	0
FMA 2.0	41648	148	20	*ALCOIF(D)*	123610	118181	0
FMA 3.2.1	84454	132	67	*ALCOIF(D)*	88204	84987	0
FMA-OBO	75139	2	0	*ALE*	119558	119558	0

Table 1: In the table the Original and OWL 2 QL axioms fields indicate respectively the total number of axioms in the original version of the ontology and in the OWL 2 QL-approximated version. The Negative inclusion field reports the number of negative inclusions in the OWL 2 QL-approximated version.

4 Implementation and Evaluation

By exploiting the results presented in Section 3, we have developed a Java-based OWL 2 QL classification module for the QuOnto reasoner [1,6,8].

This module computes the classification of an OWL 2 QL TBox $\mathcal{T}$ by adopting the technique described in Section 3. In this implementation the transitive closure of the digraph $\mathcal{G}_\mathcal{T}$ is based on a breadth first search through $\mathcal{G}_\mathcal{T}$. In the implementation we have considered all aspects of OWL 2 QL which were ignored in the theoretical discussion presented in the previous sections (see Section 2).

We have performed comparative experiments, where QuOnto was tested against several popular ontology reasoners. Specifically, during our test we compared ourselves with the Fact++ [24], Hermit [9], and Pellet [23] OWL reasoners, and with the CB [14] Horn $\mathcal{SHIQ}$ reasoner, and with the ELK [15] reasoner for those ontologies that are also in OWL 2 EL.

The ontology suite used during testing includes twenty OWL ontologies, assembled from the TONES Ontology Repository[3] and from other independent sources. The six reasoners exhibited negligible differences in performance for the majority of the smaller tested ontologies, so we will only discuss the ontologies which offered interesting results, meaning those on which reasoning times are significantly different for at least a subset of the reasoners.

These ontologies include: the Mouse ontology; the Transportation ontology[4]; the Descriptive Ontology for Linguistic and Cognitive Engineering (DOLCE) [19]; the Athletic Events Ontology (AEO)[5]; the Gene Ontology (GO) [3]; two versions of the GALEN ontology [22]; and four versions of the Foundational Model of Anatomy Ontology (FMA) [10].

Because QuOnto is an OWL 2 QL reasoner, each benchmark ontology not in OWL 2 QL was preprocessed prior to classification in order to fit OWL 2 QL expressivity. Therefore, every OWL expression which cannot be expressed

[3] http://owl.cs.manchester.ac.uk/repository/
[4] http://www.daml.org/ontologies/409
[5] http://www.boemie.org/deliverable_d_3_5

Ontology	QuOnto	FaCT++	HermiT	Pellet	CB	ELK
Mouse	0.156	0.282	0.296	0.179	0.159	0.246
Transportation	0.150	0.045	0.163	0.151	0.195	0.343
DOLCE	1.327	0.245	25.619	1.696	1.358	—
AEO	0.650	0.743	0.920	0.647	0.605	—
Gene	1.255	1.400	3.810	2.803	1.918	1.419
EL-Galen	2.788	109.835	7.966	50.770	2.446	1.205
Galen	4.600	145.485	34.608	*timeout*	2.505	—
FMA 1.4	0.688	*timeout*	93.781	*timeout*	1.243	—
FMA 2.0	4.111	*out of memory*	*out of memory*	*timeout*	7.142	—
FMA 3.2.1	4.146	4.576	11.518	24.117	4.976	—
FMA-OBO	4.827	*timeout*	50.842	16.852	7.433	4.078

Table 2: Classification times of benchmark OWL 2 QL ontologies by QuOnto and other tested reasoners.

by OWL 2 QL axioms was approximated from the ontology specifications. This approximation follows this procedure: each axiom in the ontology is fed to an external reasoner, specifically Hermit, and every OWL 2 QL-compliant axiom that is implied from that axiom, between the ontology symbols that appear in it, is added to the OWL 2 QL-approximated ontology. For instance, the OWL assertion EquivalentClasses(ObjectUnionOf(:Male :Female) :Person) is approximated by the two assertions SubClassOf(:Male :Person) and SubClassOf(:Female :Person). Note that, as is the case in this example, the OWL 2 QL-approximated ontology may contain a greater number of axioms than the original ontology. Table 1 shows that the Mouse, Transportation, Gene, and FMA-OBO ontologies are in OWL 2 QL, and thus do not need approximation, while AEO and FMA 1.4 are subject to minimal changes by the approximation procedure.

During the tests for each reasoner, classification was performed on the OWL 2 QL-compliant versions of the ontologies resulting from the above described preprocessing. Metrics about the ontologies are reported in Table 1.

All tests were performed on a DELL Latitude E6320 notebook with Intel Core i7-2640M 2.8Ghz CPU and 4GB of RAM, running Microsoft Windows 7 Premium operating system, and Java 1.6 with 2GB of heap space. Classification timeout was set at one hour, and aborting if maximum available memory was exhausted. All figures reported in Table 2 are in seconds, and, because classification results are subject to minor fluctuation, particularly when dealing with large ontologies, are the average of 3 classifications of the respective ontologies with each reasoner. The following versions of the OWL reasoners were tested: Fact++ v.1.5.3, HermiT v.1.3.6, Pellet v.2.3.0, CB v.12, and ELK v.0.3.2.

In our test configuration, the classifications of the FMA 2.0 ontology by the Hermit and FaCT++ reasoners terminate due to an out-of-memory error. In [9], classification of this ontology by the Hermit reasoner is performed successfully, but classification time far exceeds the one registered by QuOnto.

The results of the experiments are summarized in Table 2. These results confirm that the performance offered by QuOnto compares favorably to other reasoners for almost all tested ontologies. Classification for even the largest of the tested ontologies, i.e., the FMA-OBO and FMA 3.2.1 ontologies, is performed in under 5 seconds, and memory space issues were never experienced during our tests with QuOnto. For some test cases, the gap in performance between QuOnto and other reasoners is sizeable: for instance, classification by Pellet of the Galen and FMA (1.4 and 2.0) and by FaCT++ of the FMA (1.4 and OBO) ontologies exceeds the predetermined timeout limit of one hour.

Detailed analysis of the results provided in Table 2 shows that only the CB and ELK reasoners consistently display comparable performances to QuOnto, which is fastest for all ontologies which feature only positive inclusions, with the exception of the EL-Galen, Galen, and FMA-OBO ontologies. The CB reasoner, which is the best-performing reasoner for the Galen ontology, does not however always perform complete classification. For instance, it does not compute property hierarchies. The ELK reasoner instead is slower than QuOnto for three out of the five ontologies also in OWL 2 EL, showing instead markedly better performance for EL-Galen.

Furthermore, if, as it is usually the case, an ontology does not present unsatisfiable predicates, the computation of such predicates through the exploration of all negative inclusions can be avoided. This is the case for ontologies such as DOLCE and AEO, for which computation of the set $\Phi_{\mathcal{T}}$ of positive inclusion assertions resulting from the transitive closure of $\mathcal{G}_{\mathcal{T}}$ is performed respectively in 0.347 and 0.384 seconds, fastest among tested reasoners. Instead, for ontologies such as Pizza and Transportation, which feature respectively 2 and 62 unsatisfiable atomic concepts, the identification of all such predicates is unavoidable, and the resulting set of trivial inclusion assertions must be added to $\Omega_{\mathcal{T}}$.

5 Conclusions

The research presented in this paper can be extended in various directions. First of all, in the implementation of our technique we have adopted a *naive* algorithm for computing the digraph transitive closure. We are currently experimenting more sophisticated and efficient techniques for this task. We are also working to optimize the procedure through which we identify unsatisfiable predicates. Finally, we are working to extend our technique to compute all inclusions that are inferred by the TBox (which, in OWL 2 QL, are a finite number). In this respect, we notice that through $\mathcal{G}^*_{\mathcal{T}}$ it is already possible to obtain the classification of all basic concepts, basic roles, and attributes, and not only that of predicates in the signature, and that, with slight modifications of computeUnsat, we can actually obtain the set of all negative inclusions inferred by an OWL 2 QL TBox. The remaining challenge is to devise an efficient mechanism to obtain all inferred positive inclusions involving qualified existential roles and attribute domains.

Acknowledgments. This research has been partially supported by the EU under FP7 project Optique (grant n. FP7-318338), and by the EU under FP7-ICT project ACSI (grant no. 257593).

References

1. A. Acciarri, D. Calvanese, G. D. Giacomo, D. Lembo, M. Lenzerini, M. Palmieri, and R. Rosati. QuOnto: Querying Ontologies. In M. Veloso and S. Kambhampati, editors, *Proc. of AAAI 2005*, pages 1670–1671. AAAI Press/The MIT Press, 2005.
2. A. Artale, D. Calvanese, R. Kontchakov, and M. Zakharyaschev. The *DL-Lite* family and relations. *J. of Artificial Intelligence Research*, 36:1–69, 2009.
3. M. Ashburner, C. Ball, J. Blake, D. Botstein, H. Butler, J. Cherry, A. Davis, K. Dolinski, S. Dwight, J. Eppig, et al. Gene Ontology: tool for the unification of biology. *Nature genetics*, 25(1):25, 2000.
4. F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. Cambridge University Press, 2nd edition, 2007.
5. J. Bang-Jensen and G. Z. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer, 2nd edition, 2008.
6. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodriguez-Muro, R. Rosati, M. Ruzzi, and D. F. Savo. The MASTRO system for ontology-based data access. *Semantic Web J.*, 2(1):43–53, 2011.
7. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, and R. Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. *J. of Automated Reasoning*, 39(3):385–429, 2007.
8. G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, R. Rosati, M. Ruzzi, and D. F. Savo. MASTRO: A reasoner for effective ontology-based data access. In *Proc. of ORE-2012*, volume 858 of *CEUR,* ceur-ws.org, 2012.
9. B. Glimm, I. Horrocks, B. Motik, R. Shearer, and G. Stoilos. A novel approach to ontology classification. *J. of Web Semantics*, 14:84–101, 2012.
10. C. Golbreich, S. Zhang, and O. Bodenreider. The foundational model of anatomy in OWL: Experience and perspectives. *J. of Web Semantics*, 4(3):181–195, 2006.
11. V. Haarslev and R. Möller. RACER system description. In R. Goré, A. Leitsch, and T. Nipkow, editors, *Proc. of IJCAR 2001*, volume 2083 of *LNCS*, pages 701–706. Springer, 2001.
12. Q. Ji, P. Haase, G. Qi, P. Hitzler, and S. Stadtmüller. RaDON - repair and diagnosis in ontology networks. In L. Aroyo, P. Traverso, F. Ciravegna, P. Cimiano, T. Heath, E. Hyvönen, R. Mizoguchi, E. Oren, M. Sabou, and E. P. B. Simperl, editors, *Proc. of ESWC 2009*, volume 5554 of *LNCS*, pages 863–867. Springer, 2009.
13. A. Kalyanpur, B. Parsia, E. Sirin, and J. A. Hendler. Debugging unsatisfiable classes in OWL ontologies. *J. of Web Semantics*, 3(4):268–293, 2005.
14. Y. Kazakov. Consequence-driven reasoning for Horn $\mathcal{SHIQ}$ ontologies. In C. Boutilier, editor, *Proc. of IJCAI 2009*, pages 2040–2045. AAAI press, 2009.
15. Y. Kazakov, M. Krötzsch, and F. Simancik. Concurrent classification of $\mathcal{EL}$ ontologies. In L. Aroyo, C. Welty, H. Alani, J. Taylor, A. Bernstein, L. Kagal, N. F. Noy, and E. Blomqvist, editors, *Proc. of ISWC 2011*, volume 7031 of *LNCS*, pages 305–320. Springer, 2011.
16. M. Krötzsch. The not-so-easy task of computing class subsumptions in OWL RL. In P. Cudré-Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J. X. Parreira, J. Hendler, G. Schreiber, A. Bernstein, and E. Blomqvist, editors, *Proc. of ISWC 2012*, volume 7649 of *LNCS*, pages 279–294. Springer, 2012.
17. M. Lawley and C. Bousquet. Fast classification in Protégé: Snorocket as an OWL 2 EL reasoner. In T. Meyer, M. Orgun, and K. Taylor, editors, *In Proc. of AOW 2010*, volume 122 of *CRPIT*, pages 45–50. ACS, 2010.

18. D. Lembo, V. Santarelli, and D. F. Savo. Graph-based Ontology Classification in OWL 2 QL. In *Proc. of ESWC 2013*, 2013. (to appear).
19. C. Masolo, S. Borgo, A. Gangemi, N. Guarino, A. Oltramari, and L. Schneider. The wonderweb library of foundational ontologies and the DOLCE ontology. Technical Report D17, WonderWeb, 2002.
20. J. Mendez, A. Ecke, and A. Turhan. Implementing completion-based inferences for the $\mathcal{EL}$-family. In *Proc. of DL 2011*, volume 745 of *CEUR,* `ceur-ws.org`, 2011.
21. B. Motik, B. Cuenca Grau, I. Horrocks, Z. Wu, A. Fokoue, and C. Lutz. OWL 2 Web Ontology Language – Profiles (2nd edition). W3C Recommendation, World Wide Web Consortium, Dec. 2012. Available at `http://www.w3.org/TR/owl2-profiles/`.
22. J. Rogers and A. Rector. The GALEN ontology. *Medical Informatics Europe (MIE 96)*, pages 174–178, 1996.
23. E. Sirin, B. Parsia, B. C. Grau, A. Kalyanpur, and Y. Katz. Pellet: A practical OWL-DL reasoner. *J. of Web Semantics*, 5(2):51–53, 2007.
24. D. Tsarkov and I. Horrocks. FaCT++ description dogic reasoner: System description. In U. Furbach and N. Shankar, editors, *Proc. of IJCAR 2006*, volume 4130 of *LNCS*, pages 292–297. Springer, 2006.