Filling the gap between OWL 2 QL and QuOnto: ROWLKit

Claudio Corona, Marco Ruzzi, and Domenico Fabio Savo

Dipartimento di Informatica e Sistemistica
SAPIENZA Università di Roma
lastname@dis.uniroma1.it

Abstract. Ontologies are nowadays one of the most prominent formalisms used in the area of Semantic Web for knowledge representation. Several efforts have been made by the W3C in order to define OWL (Web Ontology Language) and by developers in order to develop reasoners able to deal with ontologies defined by means of OWL. Unfortunately, the goals of the working groups for standard languages definition rarely meet the ones of reasoners developers and viceversa. In this paper we present *ROWLKit*, a query answering system for ontologies compliant with OWL 2 QL, a profile of OWL 2 that is inspired by *DL-Lite*$_R$, one of the language of the *DL-Lite* family. In particular we show how to extend QUONTO, a pre-existing reasoner for *DL-Lite*$_R$ in order to implement the features of OWL 2 QL not natively supported by QUONTO thus bridging the gap between the standard language and the system implementation. We show the efficacy of our approach presenting experimental results based on the University Ontology Benchmark.

1 Introduction

Ontologies are nowadays one of the most prominent formalisms for knowledge representation and reasoning used in the area of Semantic Web (SW). Several efforts have been made by the W3C in order to define OWL (Web Ontology Language), a standard language for representing ontologies, and by developers in order to build reasoners able to deal with ontologies defined by means of OWL[1]. Unfortunately, the goals of the working groups for standard languages definition rarely meet the ones of systems developers and viceversa. It follows that OWL Full, the most expressive version of the Web Ontology Language, is undecidable and OWL DL, a decidable restriction of OWL Full, requires very high computational costs preventing its use in many practical cases.

In the very last years, several research efforts have been focused on the definition of languages with a limited expressive power but that allow reasoning and query answering tasks to be performed in a very efficient way. In this direction, the *DL-Lite* family [5] of languages based on Description Logics (DLs) [3] has

[1] `http://www.w3.org/TR/owl-ref/`

been proposed, providing languages featuring enough expressive power to allow query answering and reasoning tasks to be first-order reducible. One of the main advantages of *DL-Lite* ontologies is that query answering can be performed in a modular way, first reformulating the user query with the aim of encoding the intensional level of the ontology into the query, and then evaluating such new query over the ontology estensional level generally stored by means of a relational DBMS. The effective practical utility of such languages led on one hand developers to produce reasoners for *DL-Lite* ontologies (es. QuOnto [2], OWLGres [9]), and on the other hand standard language working groups to introduce in OWL 2 a profile called OWL 2 QL[2] covering *DL-Lite*$_R$, a notable member of the *DL-Lite* family, plus some other useful constructs not natively supported by *DL-Lite*$_R$.

In this paper we exploit the knowledge earned developing the QuOnto system, and present the implementation of *ROWLKit*, an efficient query answering engine for OWL 2 QL, based on an extended version of the QuOnto reasoner to perform query rewriting, and using standard DBMS technology to perform final query evaluation. We then conducted some experiments comparing *ROWLKit* with OWLGres, a system for efficient query answering based on *DL-Lite*. In particular our contributions can be summarized as follows: we first highlight the main OWL 2 QL features missing in current QuOnto implementation (Section 2), then show *ROWLKit*, a new reasoner that fully implement the OWL 2 QL features extending the QuOnto reasoner for *DL-Lite*$_R$ (Section 3) and eventually present some experimental results based on the UOBM benchmark [8] (Section 4). We draw some conclusions and future works in Section 5.

2 Preliminaries: OWL 2 QL and *DL-Lite*$_R$

OWL 2 QL is one of the recently presented OWL 2 profiles. This language is based on the *DL-Lite* family of Description Logics (DLs), whose relevant characteristic is that the query answering task is first-order reducible and hence can be entirely delegated to a standard DBMS. In particular the complexity of query answering w.r.t. the size of the ontology extensional level is in AC_0. We now briefly recall the basics of *DL-Lite*$_R$, that is, the language of the *DL-Lite* family that is closer to OWL 2 QL.

As presented in [5], *DL-Lite*$_R$ concepts and roles are formed according to the following syntax:

$$\begin{array}{llll} B \longrightarrow A \mid \exists R & \qquad & R \longrightarrow P \mid P^- \\ C \longrightarrow B \mid \neg B & \qquad & E \longrightarrow R \mid \neg R \end{array}$$

where A denotes an atomic concept, P an atomic role, and P^- the inverse of the atomic role P; B denotes a *basic concept* that can be either an atomic concept or a concept of the form $\exists R$, the standard DL construct of unqualified existential quantification on roles (and their inverse); finally, C denotes a (general) *concept*,

[2] `http://www.w3.org/2007/OWL/wiki/Profiles`

which can be a basic concept or its negation, and E denotes a (general) *role*, which can be a basic role or its negation.

As usual, an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ expressed in any logic of the *DL-Lite* family is constituted by a *TBox* $\mathcal{T}$ and an *ABox* $\mathcal{A}$, where the first component specifies general properties of concepts and roles, and the second component specifies the instances of concepts and roles.

A *TBox* is formed by a set of *inclusion assertions* of the form: $B \sqsubseteq C$ and $R \sqsubseteq E$, that specify the properties of concepts and roles, stating that every instance of a concept (resp., role) is also an instance of another concept (resp., role). Precisely we refer to assertions of the form $B_1 \sqsubseteq B_2$ (resp. $R_1 \sqsubseteq R_2$) as *positive inclusion assertions* (PIs), and to assertions of the form $B_1 \sqsubseteq \neg B_2$ (resp. $R_1 \sqsubseteq \neg R_2$) as *negative inclusion assertions* (NIs). Therefore general concepts (resp., roles) may only occur on the right-hand side of inclusion assertions.

An *ABox* is formed by a set of *membership assertions* on atomic concepts and on atomic roles, of the form $A(a)$ and $P(a, b)$, stating respectively that the object denoted by the constant a is an instance of A and that the pair of objects denoted by the pair of constants (a, b) is an instance of the role P.

A *union of conjunctive queries* (UCQ) q over a *DL-Lite$_R$* ontology $\mathcal{O}$ is an expression of the form

$$q(\boldsymbol{x}) \leftarrow \exists \boldsymbol{y_1}.conj_1(\boldsymbol{x}, \boldsymbol{y_1}) \vee \cdots \vee \exists \boldsymbol{y_n}.conj_n(\boldsymbol{x}, \boldsymbol{y_n})$$

where $\boldsymbol{x}$ are the so-called *distinguished variables*, $\boldsymbol{y_1}, \ldots, \boldsymbol{y_n}$ are existentially quantified variables called the *non-distinguished* variables, and each $conj_i(\boldsymbol{x}, \boldsymbol{y_i})$ is a conjunction of atoms of the form $\alpha(z)$, $\beta(z, z')$, $z = z'$, where α (resp., β) is a unary (resp., binary) atomic symbol occurring in $\mathcal{O}$, and z, z' are constants in $\mathcal{O}$ or variables in $\boldsymbol{x}$ or $\boldsymbol{y_i}$ for some $i \in \{1, \ldots, n\}$. Given an interpretation $\mathcal{I}$, q^I is the set of tuples of domain elements that, when assigned to the distinguished variables $\boldsymbol{x}$, make the formula $\exists \boldsymbol{y_1}.conj_1(\boldsymbol{x}, \boldsymbol{y_1}) \vee \cdots \vee \exists \boldsymbol{y_n}.conj_n(\boldsymbol{x}, \boldsymbol{y_n})$ true in $\mathcal{I}$ (cf. [1]). Then, the set $cert(q, \mathcal{O})$ of *certain answers to* q *over* $\mathcal{O}$ is the set of tuples $\boldsymbol{a}$ of constants appearing in $\mathcal{O}$ such that $\boldsymbol{a}^{\mathcal{I}} \in q^{\mathcal{I}}$, for every model $\mathcal{I}$ of $\mathcal{O}$. In the following, instead of query answering, we consider, w.l.o.g., boolean *query entailment* $\mathcal{O} \models q$, i.e., checking whether $\langle\rangle \in cert(q, \mathcal{O})$, where q is a Boolean UCQ (that is a UCQ with no distinguished variables) and $\langle\rangle$ denotes the empty tuple.

We now briefly highlight the more relevant differences between OWL 2 QL and the language implemented in QuOnto, which have led to the implementation of a new system called *ROWLKit* (see Section 3) by suitably extending the QuOnto reasoner. Such differences can be summarized as follows:

(*i*) existential quantification to a class (*ObjectSomeValuesFrom*) and existential quantification to a data range (*DataSomeValuesFrom*);
(*ii*) symmetric property axioms (*SymmetricObjectProperty*) and asymmetric property axioms (*AsymmetricObjectProperty*);
(*iii*) reflexive property axioms (*ReflexiveObjectProperty*) and irreflexive property axioms (*IrreflexiveObjectProperty*).

Notice that, although (*i*) and (*ii*) are not natively supported by QuOnto they are actually expressible in *DL-Lite$_R$* by suitably processing the intensional level of the ontology. Conversely, reflexivity and irreflexivity axioms requires the QuOnto system to be extended with brand new features as will be shown in the next section.

3 ROWLKit

ROWLKit is a simple graphic user interface enabling the user to reason and pose query over ontologies written in the OWL 2 QL profile of OWL 2. It allows to load an ontology, both locally stored or accessible via URI, and check for compliance with the OWL 2 QL language restrictions. Basic consistency check, reasoning services and mainly sound and complete query answering are the services provided by *ROWLKit*. *ROWLKit* is written in JAVA, uses the OWLAPI[3] as input file parser, and makes the user able to choose one of the supported DBMS for storing the extensional level of the ontology and for the evaluation of union of conjunctive queries written in the SPARQL[4] query language. The reasoning module of *ROWLKit* is based on QuOnto [2], an efficient reasoner for DL-Lite ontologies. Due to the differences existing between the OWL 2 QL and the language implemented inQuOnto, the reasoning module of *ROWLKit* has been built by extending and modifying the QuOnto reasoner. In particular, in this section we show how we choose to implement in *ROWLKit* the support for qualified existential quantification, reflexive and irreflexive property axioms, and symmetric and asymmetric property axioms.

3.1 Native handling of qualified existential quantification

OWL 2 QL provides the use of qualified existential quantification on object (and data) property as super-class in the *SubClassOf* axioms, with the qualifying class being a named class. In terms of DLs, it means that it is possible to have inclusion assertions of the form $B \sqsubseteq \exists R.A$ in the TBox. As shown in [4], these kind of TBox assertion can be handled by means of a preprocessing step that substitute them through the use of unqualified existential quantification, auxiliary roles, and inclusions between roles. More in details, in [4], each inclusion assertion α in the TBox such that α is of the form $B \sqsubseteq \exists R.A$ is replaced with $\mathit{translation}(\alpha) = \{B \sqsubseteq \exists R_{aux}, R_{aux} \sqsubseteq R, \exists R^{-}_{aux} \sqsubseteq A\}$, with R_{aux} appearing only in $\mathit{translation}(\alpha)$. This approach has two disadvantages: (i) it changes the ontology alphabet, since it adds new auxiliary roles; (ii) the output of PerfectRef [5] may contain conjunctive queries (CQs) involving auxiliary role atoms (causing the CQ in which they appear to be empty). An obvious approach to the second disadvantage is to postprocess the output of PerfectRef by deleting all the CQs containing at least one auxiliary role atom. Conversely, our approach aims to: (i) preserve the original ontology alphabet (ii) keep clean the output of PerfectRef, without the need of performing any postprocess on its output. The second point is achieved by introducing new rewriting rules in PerfectRef,

[3] http://owlapi.sourceforge.net/

[4] http://www.w3.org/TR/rdf-sparql-query/

in order to natively handle *SubClassOf* axioms involving qualified existential quantification as super-class. We start the presentation of such approach by giving some preliminary definitions.

We say that an argument of an atom in a query is *bound* if it corresponds to either a distinguished variable, i.e., a variable appearing in the head of the query, or a shared variable, i.e., a variable occurring at least twice in the query body, or a constant. Instead, an argument of an atom in a query is *unbound* if it corresponds to a non-distinguished non-shared variable and, as usual, it is denoted by the symbol '_'.

We say that a PI I is *applicable to a set of atoms Σ* if:

- I is not of the form $B \sqsubseteq \exists R.A$ and Σ contains one atom $A(x)$ and I has A in its right-hand side;
- I is not of the form $B \sqsubseteq \exists R.A$ and Σ contains one atom $P(x_1, x_2)$ and if: (i) $x_2 = \_$ and the right-hand side of I is $\exists P$, or (ii) $x_1 = \_$ and the right-hand side of I is $\exists P^-$, or (iii) I is a role inclusion assertion and its right-hand side is either P or P^-;
- I is of the form $B \sqsubseteq \exists R.A$ and Σ is formed as follow:
  - $\Sigma = \{R(x, y), A(y)\}$, with y appearing only in Σ;
  - $\Sigma = \{R(x, \_), A(\_)\}$;
  - $\Sigma = \{R(x, \_)\}$
  - $\Sigma = \{A(\_)\}$

When I is applicable to Σ, we indicate with $ngr(\Sigma, I)$ the atom obtained from the set of atoms Σ by applying I. Formally:

Definition 1. *Let I be an inclusion assertion that is applicable to the set of atoms Σ. Then, $ngr(\Sigma, I)$ is the atom defined as follows:*

1. *if $\Sigma = \{A(x)\}$ and $I = A_1 \sqsubseteq A$, then $ngr(\Sigma, I) = A_1(x)$;*
2. *if $\Sigma = \{A(x)\}$ and $I = \exists P \sqsubseteq A$, then $ngr(\Sigma, I) = P(x, \_)$;*
3. *if $\Sigma = \{A(x)\}$ and $I = \exists P^- \sqsubseteq A$, then $ngr(\Sigma, I) = P(\_, x)$;*
4. *if $\Sigma = \{P(x, \_)\}$ and $I = A \sqsubseteq \exists P$, then $ngr(\Sigma, I) = A(x)$;*
5. *if $\Sigma = \{P(x, \_)\}$ and $I = \exists P_1 \sqsubseteq \exists P$, then $ngr(\Sigma, I) = P_1(x, \_)$;*
6. *if $\Sigma = \{P(x, \_)\}$ and $I = \exists P_1^- \sqsubseteq \exists P$, then $ngr(\Sigma, I) = P_1(\_, x)$;*
7. *if $\Sigma = \{P(\_, x)\}$ and $I = A \sqsubseteq \exists P^-$, then $ngr(\Sigma, I) = A(x)$;*
8. *if $\Sigma = \{P(\_, x)\}$ and $I = \exists P_1 \sqsubseteq \exists P^-$, then $ngr(\Sigma, I) = P_1(x, \_)$;*
9. *if $\Sigma = \{P(\_, x)\}$ and $I = \exists P_1^- \sqsubseteq \exists P^-$, then $ngr(\Sigma, I) = P_1(\_, x)$;*
10. *if $\Sigma = \{P(x_1, x_2)\}$ and either $I = P_1 \sqsubseteq P$ or $I = P_1^- \sqsubseteq P^-$, then $ngr(\Sigma, I) = P_1(x_1, x_2)$;*
11. *if $\Sigma = \{P(x_1, x_2)\}$ and either $I = P_1 \sqsubseteq P^-$ or $P_1^- \sqsubseteq P$, then $ngr(\Sigma, I) = P_1(x_2, x_1)$;*
12. *if $\Sigma = \{R(x, y), A(y)\}$ with y appearing only in Σ and $I = B \sqsubseteq \exists R.A$, then $ngr(\Sigma, I) = B(x)$;*
13. *if $\Sigma = \{R(x, \_), A(\_)\}$, then $ngr(\Sigma, I) = B(x)$;*
14. *if $\Sigma = \{R(x, \_)\}$ and $I = B \sqsubseteq \exists R.A$, then $ngr(\Sigma, I) = B(x)$;*
15. *if $\Sigma = \{A(\_)\}$ and $I = B \sqsubseteq \exists R.A$, then $ngr(\Sigma, I) = B(\_)$;*

Note that the rewriting rules 1-11 correspond to the ones performed by the function gr in PerfectRef, as described in [5]. In the following, we define a new algorithm, called NewPerfectRef, obtained by extending PerfectRef with the rewriting rules 12-15.

By [4] and [5], it follows that when the TBox contains qualified existential quantifications appearing in the right-hand side of PIs, query answering can be handled in a sound and complete way by transforming those PIs into pure *DL-Lite*$_R$ PIs, and by simply considering the old algorithm PerfectRef. We now show that, considering the original TBox, in which no transformations are performed and no auxiliary roles are introduced, also NewPerfectRef is sound and complete. In particular, we show that disregarding the CQs containing auxiliary roles, the outputs of PerfectRef and NewPerfectRef are equivalent. Intuitively, considering the reformulation tree generated by PerfectRef on a input CQ q, we show that for every CQ q_{out}, different from q and without auxiliary roles, a CQ q_{anc} (ancestor query), without auxiliary roles, too, is generated before q_{out} in PerfectRef, such that NewPerfectRef, taken in input q_{anc}, produces q_{out} in one reformulation iteration. After applying for a finite number of times the same property on q_{anc}, we finally obtain q.

In the following, we denote by $\mathcal{T}_{ex}$ a set $\{\alpha_1, \ldots, \alpha_m\}$ of PIs of the form $B \sqsubseteq \exists R.A$, by $\mathcal{T}_r$ a generic *DL-Lite*$_R$ TBox (hence not including PIs of the form $B \sqsubseteq \exists R.A$), and by $\mathcal{T}_{ex}^{'}$ the set of *DL-Lite*$_R$ PIs obtained by transforming the assertions in $\mathcal{T}_{ex}$ as described in [4]. Moreover, $cq_1 \equiv cq_2$ means that the CQ cq_1 is *isomorphic* (or *equivalent*) to the CQ cq_2[6].

Lemma 1. *Given two TBoxes $\mathcal{T} = \mathcal{T}_r \cup \mathcal{T}_{ex}$ and $\mathcal{T}_t = \mathcal{T}_r \cup \mathcal{T}_{ex}^{'}$, and given two different CQs defined over the alphabet of $\mathcal{T}$ q, q_{out}, then $q_{out} \in$ PerfectRef(T_t, q) iff a CQ $q_{anc} \neq q_{out}$, defined over the alphabet of $\mathcal{T}$, exists in PerfectRef(T_t, q) such that*

1. *an atom a exists in q_{out} and a PI I exists in $\mathcal{T}$ such that $q_{anc} \equiv q_{out}[a/\Sigma]$ and $q_{anc}[\Sigma/ngr(\Sigma, I)] \equiv q_{out}$; or*
2. *a couple of atoms a_1, a_2 exists in q_{anc} such that* $\mathrm{reduce}(q_{anc}, a_1, a_2) \equiv q_{out}$.

Proof. $\Rightarrow$ If $q_{out} \not\equiv q$, then a CQ q^* must exist in PerfectRef(T_t, q) such that one of the following conditions holds:

– two atoms a_1 and a_2 exist in q^* such that *reduce*$(q^*, a_1, a_2) \equiv q_{out}$. In this case, known as a *reduce* step[5], since the condition 2 holds for q^*, $q_{anc} = q^*$;
– an auxiliary role atom a of the form $R_{aux}(x, \_)$ appears in q^* such that $q_{out} = q^*[a/gr(a, B \sqsubseteq \exists R_{aux})]$. In this case, in order to find q_{anc}, we have to go back the way one came to the atom $R_{aux}(x, \_)$ in q^* through PerfectRef. The CQ q^* may be obtained by means of a *reduce* step performed by PerfectRef on a CQ q_1^*, in which two atoms a_1 and a_2 involving R_{aux} appear, such that *reduce*$(q_1^*, a_1, a_2) \equiv q^*$. Note that, in order to let the *reduce* step produce q^*, the two atoms a_1 and a_2 must be such that the variable appearing as the first argument of the one is different from the variable appearing as the right argument of the other one, and both the variables appearing as

right argument in a_1 and a_2 must not occur elsewhere. Obviously, these kind of *reduce* steps may be repeated for a whatever, but finite, number n of iterations of PerfectRef, so to lead to a CQ q^*_n containing a set Γ_{aux} of $n+1$ atoms involving R_{aux}. To summarize, in this backwards path of q^* inside $\mathsf{PerfectRef}(T_t, q)$, if no *reduce* steps are performed, Γ_{aux} will contain only $R_{aux}(x, \_)$; if n *reduce* steps are performed, Γ_{aux} will contain $n+1$ atoms involving R_{aux}, respecting the constraint that for each couple (not necessarily different) of atoms $R_{aux} \in \Gamma_{aux}$, the variable appearing as left argument of one atom is different from the variable appearing as right argument of the other one, and all the variables appearing as right argument do not occur outside Γ_{aux}. Now, each atom R_{aux} in Γ_{aux} must have been obtained by applying a rewriting step involving one among $\exists R^-_{aux} \sqsubseteq A$ and $R_{aux} \sqsubseteq R$. In this way, we obtain a new set of atoms Γ by performing $n+1$ steps of PerfectRef, through which each R_{aux} atom in Γ_{aux} is backwards rewritten in a R atom or in a A atom. Let us indicate with q_Γ the so obtained CQ appearing in $\mathsf{PerfectRef}(T_t, q)$ containing Γ. Note that q_Γ does not contain anymore auxiliary roles. In the following, we denote by α the PI $B \sqsubseteq \exists R.A$ in $\mathcal{T}$ such that R_{aux} is the auxiliary role introduced in *translation*$(B \sqsubseteq \exists R.A)$ in $\mathcal{T}'_{ex}$. The set of atoms Γ must assume one of the following forms:

1. $\Gamma = R(x_1, y_1), ..., R(x_{n+1}, y_{n+1})$, x_i (y_i) denoting the variable appearing as left (right) argument in the i-th R_{aux} atom in Γ_{aux}. Now, if q_Γ is in $\mathsf{PerfectRef}(T_t, q)$, then, by applying n iterations of PerfectRef, $\mathsf{PerfectRef}(T_t, q)$ will also contain a CQ q_R such that $q_R \equiv q^*[R_{aux} \setminus R]$. Since $ngr(\{R(x, \_)\}, \alpha) = B(x)$ (rule 14), $q_{anc} = q_R$;
2. $\Gamma = A(y_1), ..., A(y_{n+1})$, y_i denoting the variable appearing as right argument in the i-th R_{aux} atom in Γ_{aux}. In this case, by following a way similar to the previous case, $\mathsf{PerfectRef}(T_t, q)$ will contain a CQ q_A such that $q_A \equiv q^*[R_{aux}(x, \_)/A(\_)]$. Since $ngr(\{A(\_)\}, \alpha) = B(\_)$ (rule 15), $q_{anc} = q_A$;
3. $\Gamma = R(x_1, y_1), ..., R(x_m, y_m), A(y_{m+1}), ..., A(y_n)$, y_i (x_i) denoting the variable appearing as right (left) argument in the i-th R_{aux} atom in Γ_{aux}. In this case, by following a way similar to both the previous cases, $\mathsf{PerfectRef}(T_t, q)$ will contain a CQ q_{RA} such that $q_{RA} \equiv q^*[R_{aux}(x, \_)/\{R(x_i, y_j)A(y_k)\}]$. Since $ngr(\{R(x_i, y_j), A(y_k)\}, \alpha) = B(x_i)$ (rule 12 if $y_j = y_k$, rule 13 otherwise), $q_{anc} = q_{RA}$;

– an atom a not of the form $R_{aux}(x, \_)$ exists in q^* such that $q_{out} = q^*[a/gr(a, I)]$, for some $I \in \mathcal{T}_r$. In this case, since $ngr(\{a\}, I) = gr(a, I)$, $q_{anc} = q^*$;

$\Leftarrow$ Let q_{anc} and q_{out} be two different CQ defined over the alphabet of $\mathcal{T}$ and appearing in $\mathsf{PerfectRef}(T_t, q)$. Then:

- if the condition 1 holds, since the rules 1-15 can be trivially simulated by PerfectRef, q_{out} is in $\mathsf{PerfectRef}(T_t, q)$;
- if the condition 2 holds, since the *reduce* steps are performed by PerfectRef, too, q_{out} is in $\mathsf{PerfectRef}(T_t, q)$. □

Corollary 1. *If in Lemma 1 we have that q_{anc} is equivalent to q, then a CQ q' exists in $\mathsf{NewPerfectRef}(T, q)$ such that q_{out} is equivalent to q'.*

Proof. Trivial. The step from q_{anc} to q_{out} in Lemma 1 just corresponds to an iteration step of NewPerfectRef. It follows that NewPerfectRef, on input $\mathcal{T}$ and q, is able to reach a CQ equivalent to q_{out}.

Theorem 1. *Given two TBoxes $\mathcal{T} = \mathcal{T}_r \cup \mathcal{T}_{ex}$ and $\mathcal{T}_t = \mathcal{T}_r \cup \mathcal{T}_{ex}^{'}$, and given a CQ q defined over the alphabet of $\mathcal{T}$, then, disregarding the CQs in $\mathsf{PerfectRef}(T_t, q)$ in which at least one auxiliary role appears, $\mathsf{PerfectRef}(T_t, q)$ and $\mathsf{NewPerfectRef}(T, q)$ are equivalent.*

Proof. We first prove that $\mathsf{PerfectRef}(T_t, q) \subseteq \mathsf{NewPerfectRef}(T, q)$. By contradiction.

Let q_t be a CQ appearing in $\mathsf{PerfectRef}(T_t, q)$ different from q such that a CQ q_n does not exist in $\mathsf{NewPerfectRef}(T, q)$ such that $q_t \subseteq q_n$. From Lemma 1, $q_t \in \mathsf{PerfectRef}(T_t, q)$ iff a CQ $q_{anc} \neq q_t$, defined over the alphabet of $\mathcal{T}$, exists in $\mathsf{PerfectRef}(T_t, q)$ such that

1. an atom a exists in q_t and a PI exists in $\mathcal{T}$ such that $q_{anc} \equiv q_t[a/\Sigma]$ and $q_{anc}[\Sigma/ngr(\Sigma, I)] \equiv q_t$; or
2. a couple of atoms a_1, a_2 exist in q_{anc} such that $reduce(q_{anc}, a_1, a_2) \equiv q_t$.

Note that for q_{anc} there are two possibilities:

1. $q_{anc} \equiv q$. In this case, from Corollary 1 it follows that a CQ q_t' exists in $\mathsf{NewPerfectRef}(T, q)$ such that $q_t' = q_t$. Contradiction.
2. $q_{anc} \not\equiv q$. In this case, being q_{anc} a CQ belonging to $\mathsf{PerfectRef}(T_t, q)$, and defined over the alphabet of $\mathcal{T}$, Lemma 1 holds recursively by considering q_{anc} as the new q_t, until $q_{anc} \equiv q$. Again, from Corollary 1 it follows that a CQ q_t' exists in $\mathsf{NewPerfectRef}(T, q)$ such that $q_t' \equiv q_t$, for every q_t for which Lemma 1 has held. Contradiction.

We now prove that $\mathsf{NewPerfectRef}(T, q) \subseteq \mathsf{PerfectRef}(T_t, q)$. According to Lemma 1, each rewriting rules in NewPerfectRef can be simulated (in at most three iterations) by PerfectRef. □

Theorem 2. *Given a TBox $\mathcal{T} = \mathcal{T}_r \cup \mathcal{T}_{ex}$ and a CQ q defined over the alphabet of $\mathcal{T}$, then $\mathsf{NewPerfectRef}(\mathcal{T}, q)$ is sound and complete.*

Proof. Let $\mathcal{T}_t$ be the *DL-Lite*$_R$ TBox obtained by replacing $\mathcal{T}_{ex}$ with $\mathcal{T}_{ex}^{'}$. Since $\mathsf{PerfectRef}(\mathcal{T}_t, q)$ is sound and complete, from Theorem 1 it follows that $\mathsf{NewPerfectRef}(\mathcal{T}, q)$ is sound and complete, too. □

Note that all the results above can be extended so that they hold even for existential quantification on data property appearing as super-class in the *SubClassOf* axioms.

3.2 Symmetric and asymmetric object property axioms

To support *SymmetricObjectProperty* axiom and *AsymmetricObjectProerty* axiom it is not necessary to modify the reformulation algorithm, since those axioms can be treated by translating them into suitable intensional axioms. To be more precise:

- each axiom *SymmetricObjectProperty(R)* in the ontology is treated as if it was an axiom *SubObjectProperty(R InverseOf(R))*, which in *DL-Lite*$_R$ can be written as $R \sqsubseteq R^-$;
- each axiom *AsymmetricObjectProperty(R)* in the ontology is treated as if it was an axiom *DisjointObjectProperties(R InverseOf(R))*, which in *DL-Lite*$_R$ can be written as $R \sqsubseteq \neg R^-$.

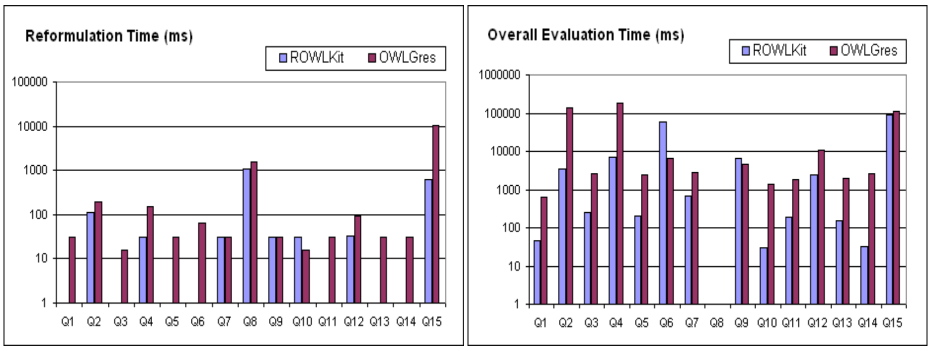
3.3 Reflexive and irreflexive object property axioms

We now present how *ROWLKit* deals with the *IrreflexiveObjectProperty* axiom. We point out that, since (i) the asymmetry axiom on a property implies the irreflexivity axiom on the same property; and (ii) the asymmetry axiom influences only the consistency check on the ontology; the irreflexivity axiom influences only the consistency check, too. It means that to treat an irreflexivity axiom on a property on which an asymmetry axiom does not exist, we need only to identify a suitable CQ q_I to add to the boolean union of CQs used to check the consistency of the ontology (see [5]). We now show how to identify such a CQ. Let $\mathcal{O}$ be a consistent OWL 2 QL ontology without irreflexivity axioms, α be an irreflexivity axiom on a object property OP, and suppose we add α to $\mathcal{O}$. We want to know if $\mathcal{O}' = \mathcal{O} \cup \alpha$ is consistent. We have that $\mathcal{O}'$ is unsatisfiable iff $\mathcal{O} \models \neg\alpha$. In other words, the ontology $\mathcal{O}'$ is unsatisfiable iff $\mathcal{O} \models \exists x.OP(x,x)$. Therefore, $\mathcal{O}'$ is consistent iff the CQ $q_I()$:-$OP(x,x)$ is false over $\mathcal{O}$. Note that the boolean CQ q_I can also be seen as a *denial constraint* [7] over $\mathcal{O}$.

Unlike the irreflexivity axiom, the reflexivity axiom is not painless from the reformulation algorithm point of view. Intuitively, a reflexivity axiom on an object property OP acts as if the extensional level of the ontology would contain a certain number of object property assertion axioms $OP(x,x)$, being in OWL 2 the reflexivity axiom defined over the (non empty) domain of interpretation. For the lack of space, we focus on the case of boolean CQs. As for generic CQs, the handling of irreflexivity axiom can be conveniently extended. Informally, if an object property is defined as reflexive, then every atom of the kind $OP(x,x)$ can be deleted from the query, since it is implied to be true by the ontology.

We now describe more precisely how the reflexivity axiom can be handled in PerfectRef. When PerfectRef analyzes a CQ q, for each atom α involving an object property OP in the body of q such that OP (or $InverseOf(OP)$) is a reflexive object property, if:

1. α is of the form $OP(x,x)$ (or $OP('a','a')$), then the atom is deleted from q;
2. α is of the form $OP(x,y)$, then a new CQ is obtained from q, and added to PerfectRef, by applying and propagating the unification $x = y$;



(a) Query reformulation time (b) Overall query evaluation time

Fig. 1. Experimental results

3. α is of the form $OP(x,'a')$ (or $OP('a',x)$), then a new CQ is obtained from q, and added to PerfectRef, by applying and propagating the unification $x='a'$;

Note that, differently from what happened without reflexivity axioms, the version of PerfectRef supporting the reflexivity axioms returns an output in which some CQs may have an empty body. If PerfectRef$(\mathcal{T}, q)$ contains some CQs with an empty body, then it simply means that $\langle \mathcal{T}, \mathcal{A} \rangle \models q$.

4 Experiments with *ROWLKit*

In this Section we presents the results of some experiments we conducted with *ROWLKit* in order to test the practical applicability of the techniques presented in Section 3. To this aim we set up an experimental scenario based on the University Ontology Benchmark (UOBM)[8], an extension of the well known Lehigh University Benchmark[5] (LUBM), that provide a test ontology using a set of constructs that are relevant for OWL 2 QL and hence for *ROWLKit*.

We decided to compare our techniques with the one implemented in OWLGres since, as far as we know, it is the only system besides QuOnto admittedly based on $DL\text{-}Lite_R$ that provides query answering techniques over ontologies with a big number of individuals. As for the ontology used for experiments we point out that, since UOBM is expressed by means of OWL DL, we manually deleted from the UOBM ontology all constructs not expressible in the OWL 2 QL profile (e.g. transitive object properties and functional object properties). The resulting ontology, whose language is fully supported by both OWLGres and *ROWLKit*, contains about 280.000 individuals. We used Derby[6] as relational DBMS for handling the extensional level of the ontology and to perform

[5] `http://swat.cse.lehigh.edu/projects/LUBM/`
[6] `http://db.apache.org/derby/`

final evaluation of the rewritten queries, since currently is the only DBMS natively supported by both the systems involved in the tests. The size of the DBMS storing the individuals is 80 MB for OWLGres and 160 MB for *ROWLKit*.

We performed the experiments running 15 queries (all the 14 queries of the UOBM benchmark, plus a custom query) on both systems, measuring the query reformulation and overall query evaluation times. Figure 1 shows the results. First of all we noticed that the reformulation step is performed faster by *ROWLKit* in all the queries except query Q_{10}. We argue that the native management of the qualified existential quantification on object and data properties provides two benefits: (*i*) in general a lower number of subclass assertion implies a lower number of queries generated during the reformulation process, and hence in the final rewriting and (*ii*) we don't have to further process the rewritten query in order to delete useless queries. As a consequence of such a new technique, the number of rewritten queries generated by *ROWLKit* is always sensibly lower, or at least equal, w.r.t. the number of queries generated by OWLGres. Secondly, the overall query evaluation process is performed faster by *ROWLKit* in almost all queries: this is mainly due to the way *ROWLKit* stores the individuals into the DBMS. However, OWLGres appears to be more efficient from the disk usage point of view. It is important to note that, due to the high complexity of the structure of query Q_8, the corresponding rewriting cannot be evaluated over the Derby database although we verified that other DBMS can handle it.

As a final remark we point out that all queries from Q_1 to Q_{14} returned the same number of answers on both systems, whereas *ROWLKit* retrieved 34 answers for query Q_{15}, against the 31 retrieved by OWLGres. This is due to the fact that *ROWLKit* implements a sound and complete query answering algorithm whereas OWLGres is sound and complete under the ground semantics only.

5 Conclusion and Future Works

In this paper we presented the main techniques implemented in the *ROWLKit* system. *ROWLKit* is a new system, based on an extension of the QuOnto reasoner, suitably developed for answering queries over ontologies expressed by means of OWL 2 QL, one of the recently proposed OWL2 profiles.

In particular we highlighted the main differences between OWL 2 QL and *DL-Lite*$_R$, the core language which QuOnto is based on, and extended the rewriting technique of QuOnto in order to natively handle some of the new features provided with OWL 2 QL. In order to test the effectiveness of the presented techniques, we performed some query answering tests based on the UOBM benchmark, and compared the performance of *ROWLKit* and OWLGres, another system for query answering based on the *DL-Lite*$_R$ language. The results of experiments confirmed that the reformulation step is a crucial task for such kind of systems and has to be performed as efficiently as possible, in order to lighten the workload of the DBMS in charge for the final evaluation of the queries over the instance level.

As future works we plan to extend *ROWLKit* in several directions: first of all we plan to broaden the set of DBMS supported by the system for the individuals storage and perform a comparison between such DBMS. Then we plan to extend to non-boolean conjunctive queries the technique for the handling of reflexive object and data properties presented in Section 3, and implement new optimizations in the reformulation process. Eventually, is also worth investigating the different ways of storing the ontology individuals into the DBMS in order to speed up the final query evaluation process.

References

1. Serge Abiteboul, Richard Hull, and Victor Vianu. *Foundations of Databases*. 1995.
2. Andrea Acciarri, Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, Mattia Palmieri, and Riccardo Rosati. QuOnto: QUerying ONTOlogies. pages 1670–1671, 2005.
3. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter F. Patel-Schneider, editors. *The Description Logic Handbook: Theory, Implementation and Applications*. 2003.
4. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Data complexity of query answering in description logics. pages 260–270, 2006.
5. Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. Tractable reasoning and efficient query answering in description logics: The *DL-Lite* family. 39(3):385–429, 2007.
6. Chandra Chekuri and Anand Rajaraman. Conjunctive query containment revisited. pages 56–70, 1997.
7. Claudio Corona, Emma Di Pasquale, Antonella Poggi, Marco Ruzzi, and Domenico Fabio Savo. When OWL met DL-Lite... In *SWAP*, 2008.
8. Li Ma, Yang Yang, Zhaoming Qiu, Guo Tong Xie, Yue Pan, and Shengping Liu. Towards a complete owl ontology benchmark. In *ESWC*, pages 125–139, 2006.
9. Markus Stocker and Michael Smith. Owlgres: A scalable owl reasoner. In *OWLED*, 2008.