Implementing and Evaluating A Rule-based Approach to Querying Regular $\mathcal{EL}+$ Ontologies

Yuting Zhao, Jeff Z. Pan, and Yuan Ren
Department of Computing Science, University of Aberdeen, UK

Abstract

Recent years have witnessed the wide recognition of the importance of ontology and rule in the AI research. In this paper, we report our implementation and evaluation of a rule-based approach to querying regular $\mathcal{EL}+$, a restriction of a well known description logics based ontology language $\mathcal{EL}+$, by only allowing regular role axioms. It is known that, without such a restriction, query answering in $\mathcal{EL}+$ in general is undecidable. In our approach, a regular $\mathcal{EL}+$ ontology is first translated into a logic program which contains a set of rules, and then by forward chaining reasoning the pseudo model of the above logic program is calculated. Query answering for $\mathcal{EL}+$ is rewritten to instance checking in the pseudo model of a logic program. To the best of our knowledge, this is the first report of implementation and evaluation for regular $\mathcal{EL}+$ ontologies.

1 Introduction

According to widely known proposals for a Semantic Web architecture, Description Logics (DLs)-based ontologies will play a key role in the Semantic Web [12]. This has led to considerable efforts to developing a suitable ontology language, culminating in the design of the OWL Web Ontology Language [13], which is now a W3C recommendation. OWL is so successful that OWL 2 (the second version of OWL) is currently undergoing standardisation by the W3C OWL Working Group\(^1\). An interesting addition to this new version of OWL is the definition of three syntactic fragments, called Profiles. Each of these profiles is tailored to support tractable reasoning in a variety of situations.

Rules in the Web have become another mainstream topic these days. Inference rules, such as SWRL rules [6], can be marked up for e-applications, such as e-commerce and e-science. The World Wide Web Consortium (W3C) has set up a Rule Interchange Format (RIF) Working Group\(^2\) to develop a set of dialects, each of which can be used to exchange rules among a category of rule engines. Ontologies and rules are closely related. For example, it has been shown\(^3\) that the RIF Core Dialect can express the inference rules for OWL 2 RL, a rules-oriented profile of the coming OWL 2 standard ontology language.

In this paper, we implement and evaluate a rule-based approach [18] to querying another OWL 2 profile, i.e., OWL 2 EL, which enables polynomial time algorithms for all the basic reasoning tasks. From the system aspect, there exist some efficient implementations of the $\mathcal{EL}+$ [3] subset of OWL 2 EL, such as CEL [2]. In fact, $\mathcal{EL}+$ is perhaps the most widely used subset of OWL 2 EL. Many well known ontologies such as the Systematized Nomenclature of Medicine SNOMED\(^4\) [17], the Gene Ontology (GO)\(^5\) and large parts of the Galen Medical Knowledge Base (GALEN) [15] can be expressed in $\mathcal{EL}+$. Although results from Krötzsch et. al. [9] show that query answering for OWL 2 EL generally is undecidable, Krötzsch et. al. [9] also show that query answering in regular $\mathcal{EL}+$ is decidable. A regular $\mathcal{EL}+$ ontology is an $\mathcal{EL}+$ ontology in which the role hierarchy does not contain cyclic dependencies other than through direct recursion of a single role.

The main idea of the rule-based approach is as follows: a regular $\mathcal{EL}+$ ontology is first translated into a logic program which contains a set of rules, and the pseudo model of the above logic program is calculated by applying forward chaining reasoning. Query answering for $\mathcal{EL}+$ is then reduced to instance checking in the pseudo model of the logic program. To the best of our knowledge, there is no published practical algorithm for querying regular $\mathcal{EL}+$ ontologies. In this paper, we present our implementation in the REL reasoner which support query answering service of regular $\mathcal{EL}+$. We compare REL with a well known reasoner Pellet which is an OWL 2 DL reasoner that does not allow non-distinguished variables.

2 Preliminaries

We assume the reader to be familiar with the basic notions of Description Logic and Logic Programming, and just give brief introduction to the preliminaries. For details please refer [18].

---

\(^1\)http://www.w3.org/2007/OWL/wiki/Document_Overview
\(^2\)http://www.w3.org/2005/rules/wiki/RIF_Working_Group
\(^3\)http://www.w3.org/2005/rules/wiki/OWLRL
\(^4\)http://www.ihtsdo.org/
\(^5\)http://www.geneontology.org/
2.1 The Description Logic \( \mathcal{EL}^+ \)

Description Logics (DLs) [4] are a family of logic-based knowledge representation formalisms used as the underpinning of the standard OWL Web ontology language. They are characterised by the constructors for building complex concept descriptions and role descriptions. In this paper, we focus on the \( \mathcal{EL}^+ \), DL. \( \mathcal{EL}^+ \) supports the following concept constructors:

\[
C ::= A \mid \exists R . C \mid C_1 \sqcap C_2 | \top
\]

where \( A \) is a concept name, \( R \) is a role name.

An \( \mathcal{EL}^+ \) ontology \( \mathcal{O} = (\tau, \mathcal{A}) \) contains a TBox \( \tau \) and an ABox \( \mathcal{A} \). \( \tau \) is a finite set of axioms of the form (i) \( C_1 \sqsubseteq C_2 \), and (ii) \( r_1 \circ r_2 \sqsubseteq R_3 \), where \( C_i \) is a concept description and \( r_i \) is a role name. Axioms are called general concept inclusions (GCIs) and role inclusions (RIs) respectively. \( \mathcal{A} \) is a finite set of assertions of the form (i) \( C(a) \) or (ii) \( r(a, b) \), where \( a, b \) are individual names, \( C \) is a concept description and \( r \) is a role name.

For an \( \mathcal{EL}^+ \) ontology \( \mathcal{O} = (\tau, \mathcal{A}) \), an interpretation \( I \) is a pair \( (\Delta^I, \cdot^I) \), where \( \Delta^I \) is a nonempty domain, and \( \cdot^I \) is a mapping that assigns: (i) to each concept name \( A \) a subset of \( \Delta^I \); (ii) to each role name \( r \) a subset of \( \Delta^I \times \Delta^I \); (iii) to each individual name \( o \in \Delta^I \) an element of \( \Delta^I \). The interpretation function \( \cdot^I \) is extended to give semantics to \( \mathcal{EL}^+ \) concept descriptions as follows: \( \exists R . C = \{ x \in \Delta^I \mid \exists y (x, y) \in R^I \land y \in C^I \} \), \( C_1 \sqcap C_2 = C_1^I \cap C_2^I \). An interpretation \( I \) satisfies a GCI \( C_1 \sqsubseteq C_2 \) if \( C_1^I \subseteq C_2^I \). An interpretation \( I \) satisfies an RI \( r_1 \sqsubseteq r_2 \) if \( r_1^I \subseteq r_2^I \). An interpretation \( I \) satisfies an individual assertion \( C(a) \) if \( a^I \in C^I \). An interpretation \( I \) satisfies an \( \mathcal{EL}^+ \) ontology \( \mathcal{O} \) if it satisfies all GCIs, RIs and individual assertions in \( \mathcal{O} \); in this case, \( I \) is called a model of \( \mathcal{O} \). Let \( O \) be an ontology and \( \phi \) a GCI, a RI or an individual assertion. We say \( O \) entails \( \phi \), denoted by \( O \models \phi \), if all models of \( O \) satisfy \( \phi \).

2.2 Normalisation of \( \mathcal{EL}^+ \)

\[
\begin{array}{c|c|c}
\text{Extended GCIs} & A_1 \sqcap \cdots \sqcap A_n \subseteq B, & A_1 \sqsubseteq \exists R . A_2, \\
& \exists r . A_1 \subseteq B & \exists r . A_2 \subseteq B
\end{array}
\]

\[
\begin{array}{c|c|c}
\text{Extended RIs} & r \sqsubseteq s, & r_1 \circ r_2 \sqsubseteq s
\end{array}
\]

\[
\begin{array}{c|c|c}
\text{ABox assertions} & A(\alpha), & r(\alpha, \beta)
\end{array}
\]

\]

Legends: \( A_i \in CN_i, B \in CN \) and \( r, r_1, r_2, s \) are role names.

Table 1: Normal form of \( \mathcal{EL}^+ \) ontology

By introducing new concept and role names, a number of straightforward rewriting rules [18] can transform an \( \mathcal{EL}^+ \) ontology \( O \) into a normalized one, such that memberships and subsumption relationships are preserved. Then a \( \mathcal{EL}^+ \) ontology is in normal form as in Table-1.

Let \( O \) be an \( \mathcal{EL}^+ \) ontology, and \( O' \) the normal form of \( O \) by applying rewriting rules. It is easy to see, for any axiom \( \phi \), if \( O \models \phi \) then \( O' \models \phi \). We note that the normalization of an \( \mathcal{EL}^+ \) ontology is linear time complexity, which follows [3].

2.3 Regular \( \mathcal{EL}^+ \)

It is known that conjunctive queries in \( \mathcal{EL}^+ \) are undecidable in general [16; 9]. Fortunately, [9] showed it is decidable on a fragment of \( \mathcal{EL}^+ \) which is called regular. A \( \mathcal{EL}^+ \) ontology is regular if there is a strict partial order \( \prec \) on RIs such that, for all role inclusion axioms \( r_1 \sqsubseteq s \) and \( r_1 \circ r_2 \sqsubseteq s \), we find \( r_1 \prec s \) or \( r_1 = s \) (i = 1, 2).

In this paper we always consider regular \( \mathcal{EL}^+ \) ontology. For a normalized ontology \( O \), we denote the set of concept names by \( N_C \), the set of role names by \( N_R \), and the set of individuals by \( N_I \).

2.4 Query Answering

A conjunctive query is of the form \( q(X) \leftarrow \exists U . \varphi(X, U) \), or simply \( q(X) \leftarrow \varphi(X, U) \), where \( X \) and \( U \) are vectors of distinguished and non-distinguished variables resp., and \( \varphi \) is a conjunct of the form \( A(v), R(v_1, v_2) \), where \( A, R \) are named concepts and named roles resp., \( v, v_1 \) and \( v_2 \) are variables in \( X \) and \( U \), or individual names in the given ontology. If \( v, v_1 \) and \( v_2 \) are not in \( U \), then \( A(v) \) and \( R(v_1, v_2) \) are non-distinguished variable free atoms. If \( X \) is an empty set, we say \( q \) is a boolean query; otherwise, we say \( q \) is a non-boolean query. Theoretically, allowing only named concepts and roles in atoms is not a restriction, as we can always define such named concepts and roles in ontologies. Practically, this should not be an issue as querying against named relations is a usual practice when people query over relational databases. As usual, an interpretation \( I \) satisfies an ontology \( O \) if it satisfies all the axioms in \( O \); in this case, we say \( I \) is a model of \( O \). Given an evaluation \( X \mapsto S \), if every model \( I \) of \( O \) satisfies \( q_{[X \mapsto S]} \), we say \( O \) entails \( q_{[X \mapsto S]} \); in this case, \( S \) is called a solution of \( q \).

3 Translate \( \mathcal{EL}^+ \) knowledge into rules

Given a regular consistent \( \mathcal{EL}^+ \) ontology \( O = (\tau, \mathcal{A}) \), we can translate \( O \) into \( \Pi^0 \), which is a set of horn rules, according to Table-2. We also do not consider “negation as failure” in \( \Pi^0 \) since the original \( \mathcal{EL}^+ \) ontology \( O \) is monotonic.

\[
\begin{array}{c|c}
\mathcal{EL}^+ \text{ axiom} & \text{Rule} \\
\hline
A_1 \sqcap \cdots \sqcap A_n \subseteq B, & r(\alpha, \beta) \leftarrow A(\alpha), (\alpha, \beta)
\end{array}
\]

Table 2: Translating \( \mathcal{EL}^+ \) axioms into rules

Following the common understanding in Logic Programming, in Table-2 \( \alpha \) and \( \beta \) in rule\(_0\) and rule\(_7\) are constants. \( A(\alpha) \) and \( r(\alpha, \beta) \) are facts. We call a rule \( r \) is grounded, if all of the variables in \( r \) are replaced by constants. Given a set of atoms \( M \), for a grounded rule \( r; p_1, \cdots, p_m \leftarrow q_1, \cdots, q_n \), where \( p_i \) (\( 1 \leq i \leq m \)) and \( q_j \) (\( 1 \leq j \leq n \)) are atoms, we say \( r \) is satisfied by \( M \), denoted by \( M \models r \).
iff we have: for any \( j \ (1 \leq j \leq n) \) \( q_j \in M \), then for any \( i \ (1 \leq i \leq m) \) \( p_i \in M \). Let \( \Pi \) be a set of rules, we say \( M \) satisfies \( \Pi \), denoted by \( M \models \Pi \), if \( M \) satisfies any rule \( r \in \Pi \). In this case we say \( M \) is a model of \( \Pi \). Let \( \phi \) be a logic expression, for any model \( M \) of \( \Pi \), if \( M \models \phi \), then we say \( \Pi \) entails \( \phi \), and denote this by \( \Pi \models \phi \).

We note that the translation is linear w.r.t. the size of the ontology.

For rule2 in Table-2, in order to represent some elements “unknown”, we introduce a unique function symbol \( f_{i,B}(x) \) for each axiom of the form \( A \subseteq \exists r.B \), where \( i \) is the index of the concept inclusion in the left side. sometime we skip \( B \) in the function symbol, if it is clear or not important. We also use the kind of convenience like, \( f_1^{2}(x) \) for denoting \( f_1(f_1(x)) \), and \( f_2^{3}(x) \) for \( f_1(f_2(f_2(f_2(x)))) \), etc.

A function symbol \( f_{i,B}(x) \) will be grounded during the forward chaining reasoning, which will be introduced in next section. Grounded function symbol, e.g. \( f_{i,B}(\alpha) \), will be used in answering queries.

4 Forward chaining reasoning in the rules

Since function symbols are involved in the rules, traditional logic programming approach is not suitable to support reasoning among the rules. Here we introduce a forward chaining approach to compute the pseudo model, in order to capture the semantics, of a set of rules generated from last section.

Let \( O = \langle T, A \rangle \) be a regular consistent \( \mathcal{EL}+ \) ontology in normal form, and \( \Pi^O \) the logic program obtained by applying translating rule in Table-2 on \( O \). Let \( M_{\text{pseudo}} = \{ \langle C \mid \text{if } C \in N_{C} \} \cup \{ \tau \mid \text{if } r \in N_{R} \} \} \) be the pseudo model of \( O \), where we use \( \tau \) to represent a pseudo model for each concept name \( C \), and \( \tau \) for each role name \( r \). We also use a representative symbol \( X_{C} \) for each concept name \( C \) as a value in the pseudo model. In the following algorithms, we use \( \alpha \) (or \( \beta \)) to denote an individual of \( O \), or a grounded function symbol like \( f_{1}(\alpha) \). We also use \( g_{i}(\cdot) \) to present an arbitrary function or function composition like \( f_{1}(f_{2}(f_{3}(\cdot))) \). We also use \( g_{i}^{-1} \) to remove outer function(s) from a combined function symbol. For example, let \( g_{1}(\cdot) = f_{1}(f_{2}(f_{3}(\cdot))) \), and \( g_{2}(\cdot) = f_{4}(\cdot) \), so \( g_{2}^{-1}(g_{1}(\alpha)) = f_{3}(f_{2}(f_{1}(\alpha))) \).

The algorithm needs some explanation and clarification.

- After initialization and collection primary facts in step-1 and step-2, the algorithm does normal forward chain reasoning with rules in the form of rule1, rule2, rule4, rule5 and extends the pseudo model (step-4~8).

- In step-9~20 the algorithm deals with rule2 which contains function symbol. If a loop is detected, the algorithm makes a breaking, to introduce a pattern composed by representative symbol and function symbol to remember this loop. And extend the pseudo model accordingly (step-10~16). Otherwise just extended the pseudo model with function symbols (step-18).

- The algorithm terminates if \( M_{\text{pseudo}} \) does not change.

Theorem 1 The Algorithm A-1 terminates in polynomial time w.r.t. the size of the ontology.

5 Query answering

Let \( O \) be a regular consistent \( \mathcal{EL}+ \) ontology in normal form, and \( \Pi^O \) the logic program obtained by applying translating rule in Table-2 on \( O \). Let \( M_{\text{pseudo}} \) be the pseudo model of \( \Pi^O \) by applying Algorithm A-1, and \( N_{EI} \) the set of extended individuals which contains all of the function symbols and representative symbols in \( \Pi^O \) and its pseudo model. For a conjunctive query \( q(X) \leftarrow \exists U \cdot \varphi(X, U) \), an answer of \( \Pi^O \) is a set of matching \( \pi \), such that (i) \( \pi \vdash \varphi \); (ii) \( \pi \vdash u \leftarrow N_{I} \cup N_{EI} \); (iii) \( M_{\text{pseudo}} \models \varphi(\pi[X], \pi[U]) \).

In the following algorithms, we use \( \alpha \) (or \( \beta \)) to denote an individual of \( O \), or a grounded function symbol like \( f_{1}(\alpha) \). We also use \( g_{i}(\cdot) \) to present an arbitrary function or function composition like \( f_{1}(f_{2}(f_{3}(\cdot))) \). We also use \( g_{i}^{-1} \) to remove outer function(s) from a combined function symbol. For example, let \( g_{1}(\cdot) = f_{1}(f_{2}(f_{3}(\cdot))) \), and \( g_{2}(\cdot) = f_{4}(\cdot) \), so \( g_{2}^{-1}(g_{1}(\alpha)) = f_{3}(f_{2}(f_{1}(\alpha))) \).

Algorithm A-2 Matching returns all answers of query \( q \) according to the pseudo model \( M_{\text{pseudo}} \) of \( \Pi^O \). Here \( q \) is a conjunctive query, and \( \pi \) is a (partial) assignment. We use \( \pi(q) \) to denote a query resulted from \( q \) by replacing each (distinguish and non-distinguish) variable \( y \) with \( \alpha \) if there is \( y \mapsto \alpha \in \pi \). Matching recursively loads itself via algorithm Extend, in which it combines the solutions with the new (partial) assignments. Matching and Extend also load Valid, which checks if the current query \( q \) is unsatisfiable according to pseudo model \( M_{\text{pseudo}} \). For example, if \( \text{Man}(Mary) \) in \( q \) but \( \text{Man}(Mary) \notin \text{Man} \), then it is not Valid.

Algorithm A-2: Matching(q, M\text{pseudo})

1: Initialize the set of solutions \( S = \emptyset \);
2: if \( q \) has any variable then
3: for each \( r(\alpha, x) \in q \), and each \( r(\alpha, \beta) \in M_{\text{pseudo}} \) do
4: \( S = \text{Extend}(q/x/\beta), M_{\text{pseudo}}, \pi, S) \);
5: end for
6: for each \( r(\alpha, x) \in q \), \( r(X_{A_{j}}, g_{j}(X_{A_{j}})) \in M_{\text{pseudo}} \) do
7: \( S = \text{Extend}(q/x/\alpha), M_{\text{pseudo}}, \pi, S) \);
8: end for
9: for each \( r(\alpha, \beta) \in q \), \( r(X_{A_{j}}, g_{j}(X_{A_{j}})) \in M_{\text{pseudo}} \) do
10: \( S = \text{Extend}(q/x/\beta), M_{\text{pseudo}}, \pi, S) \);
11: end for
12: for each \( r(\alpha, x) \in q \), \( r(X_{A_{j}}, g_{j}(X_{A_{j}})) \in M_{\text{pseudo}} \) do
13: \( S = \text{Extend}(q/x/\alpha), M_{\text{pseudo}}, \pi, S) \);
14: end for
15: return \( M_{\text{pseudo}} \)
Matching end for
M = Answer( and to meet the Student then M
The algorithms for query answering terminate in \( c \) removes the bindings of non-
for \( g \) M end if
worksFor
then >R ⊑ \( \pi \) [bench-
M if \( q \) M end for
is the time used for normal-
C end if
M R = R, g in Query4 and can be matched to either a relation
\( \pi \) \( R \) = \( R \), \( g \) in .
else the length of the query and size of the
expressive power and our
\( \pi \) \( C \) are
\( \pi \) \( \circ \)
worksFor
ο subOrganizationOf ⊆ memberOf
(2) worksFor ο subOrganizationOf ⊆ worksFor.

After the above modification we will obtain a \( \mathcal{EL}^+ \) benchmark for conjunctive query answering. The number of concepts and properties will be the same as original LUBM but the semantics are different. In later of this section we will further extend it for purpose of completeness comparison.

6.2 Evaluation on Performance
There are 14 test queries over the original LUBM. Among them, test queries 2, 4, 5, 8 and 12 refer to "memberOf" and "worksFor" thus are of special interests to us. In order to generalize these queries into situations of multiple universities, we replace the http://www.Department0.University0.edu in query 4 and 5 by an independent variable of type Department, and replace the http://www.University0.edu in query 8 and 12 by an independent variable of type University. Therefore the selectivity of these queries becomes low and the solution set should be much larger than before. The original variables are treated as distinguishable variables in the queries while the variables introduced by us are treated as non-distinguishable variables.

The results are shown in Table 3, in Which U is the number of universities in the ABox, \( T_P \) is the time used for normalization and \( M_{pseudo} \) generation. The time unit is second.

From the above table we can see that, for queries with simple atoms, like Query5, and queries with small hierarchies, like Professor in Query4 and Chair in Query12, the processing is fast. While for queries with many variables, complex relations and large hierarchies such as Student, Department and University, etc. in Query2 and Query8,
the processing is much slower. Also the variables we have introduced make the query answering much harder. If we still use http://www.University0.edu in Query8 instead of a non-distinguishable variable, it takes only 77.155 second for 5 universities.

Table 3: Query time with increasing number of universities.

<table>
<thead>
<tr>
<th>U</th>
<th>Time (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>6.543</td>
</tr>
<tr>
<td>2</td>
<td>29.104</td>
</tr>
<tr>
<td>3</td>
<td>60.355</td>
</tr>
<tr>
<td>4</td>
<td>121.703</td>
</tr>
<tr>
<td>5</td>
<td>187.397</td>
</tr>
</tbody>
</table>

Figure 1: Trend of Query time.

6.3 Evaluation on Completeness
In this subsection, we evaluate the completeness of query service of regular $\mathcal{EL}^+$ ontologies. However, there existing no other query engine for regular $\mathcal{EL}^+$; the closest tool available is Pellet, which is an OWL 2 DL reasoner that also provide some restricted query answering service. It is well know that query answering in OWL 2 DL is still an open problem in terms of decidability; in other words, there exist no decision procedure for querying OWL 2 DL ontologies. Pellet only supports a restricted form of conjunctive queries, i.e., treating non-distinguish variables as distinguished ones.

In order to illustrate the difference between Pellet’s restricted support for regular $\mathcal{EL}^+$ and REL’s sound and complete service. Let us have a look at a small part of the benchmark ontology:

$$\text{ResearchAssistant} \sqsubseteq \exists \text{worksFor. ResearchGroup}$$
$$\text{ResearchGroup} \sqsubseteq \exists \text{researchProject. Research}$$
$$\text{ResearchAssistant}(a), \text{ResearchAssistant}(b), \text{ResearchAssistant}(c), \text{Research}(P1),$$
$$\text{ResearchGroup}(G1), \text{ResearchGroup}(G2),$$
$$\text{worksFor}(a, G1), \text{worksFor}(b, G2)$$
$$\text{researchProject}(G1, P1)$$

Note that the first subsumption already in the original LUBM benchmark, while the second subsumption is newly introduced into LUBM$\mathcal{EL}^+$. In LUBM$\mathcal{EL}^+$, for each instance of ResearchAssistant, we make it $\text{worksFor a ResearchGroup}$ with a probability. Similarly, for each instance of ResearchGroup, we make it $\text{researchProject a Research}$ with a probability. After the modification, we submit the following query to the three query engines. The results are illustrated in Table 4.

$$q(x) \leftarrow \text{ResearchAssistant}(x),$$
$$\text{worksFor}(x, y),$$
$$\text{researchProject}(y, z).$$

For the above example ontology, Pellet returns $\{a\}$ due to $\text{ResearchAssistant}(a), \text{worksFor}(a, G1)$, and $\text{researchProject}(G1, P1)$. REL returns the sound and complete answer, i.e., $\{a, b, c\}$. In example shows, by treating non-distinguish variables with distinguished ones, Pellet is not complete in query answering regular $\mathcal{EL}^+$ ontologies; REL, however, returns all of the solutions. The following table shows the answers from Pellet and REL for the above query over the LUBM$\mathcal{EL}^+$ benchmark:

Table 4: Recall and Query time with REL and Pellet.

<table>
<thead>
<tr>
<th>No. Solutions / Recall</th>
<th>Time (in seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>U</td>
<td>REL</td>
</tr>
<tr>
<td>1</td>
<td>547 / 100%</td>
</tr>
<tr>
<td>2</td>
<td>1296 / 100%</td>
</tr>
<tr>
<td>3</td>
<td>1951 / 100%</td>
</tr>
<tr>
<td>4</td>
<td>2597 / 100%</td>
</tr>
<tr>
<td>5</td>
<td>5001 / 100%</td>
</tr>
</tbody>
</table>

The above Table-4 shows that, for the given query, (i) the recall of Pellet various from 5.5% to 7.6%; i.e., Pellet only returns a small part of the answer sets, and (ii) the performance of REL is comparable to that of Pellet, even though it produces the exact (i.e., much larger) answer sets.

7 Related Works
Kazakov [8] presents a saturation base decision procedure to convert an $\mathcal{EL}$ TBox into a set of clauses and check concept subsumption accordingly. However, this approach does not support ABox conversion. KAON2 [7] provides an exponential reduction of $\mathcal{SHIQ}$ into disjunctive Datalog programs for ABox reasoning, but it does not support querying with non-distinguish variables. ELP [10] is a rule-based tractable knowledge representation language, it generalizes the known tractable description logics $\mathcal{EL}^+$ and DLP to transform an $\mathcal{EL}^+$ rule base RB into an equisatisfiable normal form $\mathcal{EL}^+$ RB′. However ELP only supports satisfiability checking, which is known to be much easier than general conjunctive query answering for $\mathcal{EL}$ and $\mathcal{EL}^+$. Hector et. al. in [14] propose a query rewriting approach for $\mathcal{ELHT}$ ontologies, in which the TBox and query are put together to translate into a Datalog program. Because the query has been fixed already, the functional symbols (very similar to those used in our approach) are easily identified and replaced. The resulted Datalog query is finally archived. The disadvantage of this approach is that the processing of TBox and queries are not separable. Krötzsch et. al. [9] theoretically indicate query answering in $\mathcal{EL}^+$ ontology is undecidable. They also propose a NFA automata based on computing canonical model; this approach can be used to check the answers of a query to $\mathcal{EL}^+$ ontology. The authors of [11] present an approach to
convert an $\mathcal{EL}$ ontology query task into a database query task. They also use representative symbols $x_C$ and $(x_C, x_D)$, for concept $C$ and $C \sqsubseteq \exists r. D$ respectively, in building the canonical model $I_K$. Since function symbols are not used in this approach, they need some efforts on unraveling the canonical model $I_K$ and query rewriting.

The work we present in this paper is different from existing works. The pseudo model that we introduced in this paper is also different from canonical model or universal model, in the sense that the former is defined in the individual layer, while the latter is defined in the layer of elements of domain. To the best of our knowledge, this is the first report of implementation and evaluation for regular $\mathcal{EL}^+$ ontologies.

8 Conclusions

In this paper, we have presented a rule-based approach to querying regular $\mathcal{EL}^+$ ontologies. To the best of our knowledge, this is the first report on practical algorithms and implementation of a regular $\mathcal{EL}^+$ engine. Our preliminary evaluation indicates that our approach work on rather large ontologies (e.g. 700,000 individuals for 5 universities in LUBM). In the future, we will work on optimisations to make the algorithms more scalable.

Acknowledgements

This work has been partially supported by the European Project Marrying Ontologies and Software Technologies (MOST ICT 2008-216691).

References


