7-2013

SROIQ Syntax Approximation by Using Nominal Schemas

Cong Wang
David Carral
Pascal Hitzler
pascal.hitzler@wright.edu

Follow this and additional works at: http://corescholar.libraries.wright.edu/cse
Part of the Computer Sciences Commons, and the Engineering Commons

Repository Citation
http://corescholar.libraries.wright.edu/cse/216

This Conference Proceeding is brought to you for free and open access by Wright State University's CORE Scholar. It has been accepted for inclusion in Computer Science and Engineering Faculty Publications by an authorized administrator of CORE Scholar. For more information, please contact corescholar@www.libraries.wright.edu.
**SROIQ** Syntax Approximation by Using Nominal Schemas

Cong Wang, David Carral, and Pascal Hitzler
Kno.e.sis Center, Wright State University, Dayton, OH, U.S.A.

**Abstract.** Nominal schemas is a recently introduced extension of description logics which makes it possible to express rules which generalize DL-safe ones. A tractable description logic, $\mathcal{ELROV}_n$, has been identified. This leads us to the question: can we improve approximate reasoning results by employing nominal schemas? In this paper, we investigate how to approximately cast $\text{SROIQ}$ into $\mathcal{ELROV}_n$. Using a datalog-based tractable algorithm, a preliminary evaluation shows that our approach can indeed do approximate $\text{SROIQ}$-reasoning with a high recall.

1 Introduction

Reasoning with large or complex terminology is computationally difficult and is one of the bottlenecks for Semantic Web applications. Most reasoning tasks for ontologies underlying OWL [11] are intractable. Even with small ontologies, sound and complete reasoning is practically infeasible, in particular for applications where quick responses are critical.

This fundamental insight that expressive ontology reasoning is often necessarily of high computational complexity has triggered a line of research which aims at utilizing approximate algorithms, i.e. algorithms which are (provably) not sound and complete, but which nevertheless provide answers which are good enough for practical purposes [6,9,10,24]. This general idea of approximate reasoning is not new and to a certain extent had been studied already before the advent of the Semantic Web [4,25,26]. But the Semantic Web effort with its increased requirements for scalability has recently put this into a focus which this branch of reasoning research has never had before [7,8,12,20,21,22,23,27].

One of the prominent general approaches to approximate reasoning is known as language weakening. Language weakening refers to the idea of rewriting a knowledge base into a language which can be handled more efficiently. Obviously, if the target language has a lower complexity class, this rewriting in general cannot be done without a loss, resulting in an approximate reasoning procedure. In order to limit loss in the translation, it is of advantage if the target language be as expressive as possible while still being of low computational complexity, and hence languages which push expressivity while retaining tractability are natural choices for a language weakening approach.

In this paper, we use $\mathcal{ELROV}_n$ for approximate reasoning over $\text{SROIQ}$ using language weakening. $\mathcal{ELROV}_n$ is essentially a tractable extension of $\mathcal{EL}^{++}$ [2],
Table 1. Normal forms of $\text{SROIQ}$ TBox axioms. $A$, $B$ and $C$ are atomic concept or negations of atomic concepts.

<table>
<thead>
<tr>
<th>$A \sqsubseteq \bot$</th>
<th>$\bot \sqsubseteq C$</th>
<th>$A \sqcap B \sqsubseteq C$</th>
<th>$A \sqsubseteq B \cup C$</th>
<th>$\exists R.A \sqsubseteq C$</th>
<th>$A \sqsubseteq \exists R.C$</th>
<th>$\forall R.A \sqsubseteq C$</th>
<th>$A \sqsubseteq \forall R.C$</th>
<th>$A \sqsubseteq {a}$</th>
<th>${a} \sqsubseteq A$</th>
<th>$\geq nR.A \sqsubseteq C$</th>
<th>$\leq nR.A \sqsubseteq C$</th>
<th>$A \sqsubseteq nR.C$</th>
<th>$A \sqsubseteq \geq nR.C$</th>
</tr>
</thead>
</table>

a.k.a. OWL 2 EL [18], by nominal schemas [17]. As such, $\mathcal{ELROV}_n$ incorporates DL-safe Datalog under Herbrand semantics [14]. We have recently described an efficient procedure to reasoning with $\mathcal{ELROV}_n$ [5] on which we base the evaluations in this paper.

The plan of this paper is as follows. In Section 2 we recall the languages $\text{SROIQ}$ and $\mathcal{ELROV}_n$. In Section 3 we describe our approximate compilation of $\text{SROIQ}$ into $\mathcal{ELROV}_n$. In Section 4 we recall our $\mathcal{ELROV}_n$ reasoning approach from [5]. In Section 5 we describe our implementation and evaluation results. In Section 6 we conclude.

2 Preliminaries

In this section, we introduce the description logics (DLs) $\text{SROIQ}$ and $\mathcal{ELROV}_n$. The latter includes the new constructor from [17], nominal schemas, which we use to approximate some features of $\text{SROIQ}$.

A signature $\Sigma = \langle \Sigma_I, \Sigma_C, \Sigma_R, \Sigma_S \rangle$ consists of mutually disjoint finite sets of atomic roles role names $\Sigma_R$, atomic concepts $\Sigma_C$, and individuals individual $\Sigma_I$, together with a distinguished subset $\Sigma_S \subseteq \Sigma_R$ of simple atomic roles. The set of roles (over $\Sigma$) is $\mathcal{R} := \Sigma_R \cup \{R^\rightarrow \mid R \in \Sigma_R\}$; the set of simple roles is $\mathcal{S} := \Sigma_S \cup \{S^\rightarrow \mid S \in \Sigma_S\}$. A role chain is an expression of the from $R_1 \ldots \ldots \ldots R_n$ with $n \geq 1$ and each $R_i \in \mathcal{R}$. The function $\text{inv}(\cdot)$ is defined on roles by $\text{inv}(R) := R^\rightarrow$ and $\text{inv}(R^\rightarrow) := R$ where $R \in \mathcal{R}$, and extended to role chains by $\text{inv}(R_1 \ldots \ldots \ldots R_n) := \text{inv}(R_n) \ldots \ldots \ldots \text{inv}(R_1)$.

The set $\mathcal{C}$ of $\text{SROIQ}$ concepts (over $\Sigma$) is defined recursively as follows:

$$\mathcal{C} := \Sigma_C[\{\Sigma_I\}] \sqcap \mathcal{C} \sqcup \mathcal{C} \sqcap \neg \mathcal{C} \sqcap \exists R.C \sqcap \forall R.C \sqcap \geq nS.C \sqsubseteq \leq nS.C \sqsubseteq \exists S\text{Self}$$

A TBox is a finite set of general concept inclusions (GCIs) of the form $C \sqsubseteq D$ where $C, D \in \mathcal{C}$. A $\text{SROIQ}$ TBox can be normalized such that it only contains the normal forms in Table 1 [1].

Satisfiability checking of $\text{SROIQ}$ ontologies is in $\text{N2ExpTime}$ [13]. Given a disjunctive assertion $(C \sqcup D)(s)$, the tableau algorithm [13] nondeterministically guesses that either $C(s)$ or $D(s)$ holds, which can give rise to exponential behavior. Although the absorption technique and the hypertableaux approach [19] reduce the cost of this nondeterminism, it is still a considerable performance bottleneck.

1 It was called $\mathcal{SROELV}_n$ in [17].
**SROIQ** defines simple roles and role regularity to ensure decidability [13]. However, since we will later approximately cast **SROIQ** into **ELROV**$_n$, which is free of these restrictions, we do not have to concern ourselves with them for the purposes of this paper. **ELROV**$_n$ extends **EL**++ with nominal schemas (see [5,17] for details). To deal with the new constructor, we extend the signature to $\Sigma = \langle \Sigma_I, \Sigma_C, \Sigma_R, \Sigma_V \rangle$, where $\Sigma_V$ is a set of variables. A nominal schema is a concept of the form $\{x\}$ where $x \in \Sigma_V$. Semantically, these variables can only bind to known individuals. The $n$ in **ELROV**$_n$ is a global bound on the number of different nominal schemas which can occur in any axiom in a knowledge base—this restriction guarantees tractability. The set of $C$ of **ELROV**$_n$ concepts is defined as follows:

$$C := \Sigma_C|\{\Sigma_I\}|\{\Sigma_V\}|C \cap C|\exists R.C|\exists S.Self$$

To give an example, consider the first-order rule

$$R_1(x, y) \land R_2(y, z) \land R_3(x, z) \rightarrow R(x, z)$$

which cannot be translated faithfully into **SROIQ**. By limiting the variable $z$ in the sense that it can bind only to known individuals (such variables are called DL-safe [16]), we can express this rule in **ELROV**$_n$ as

$$\exists R_1.\exists R_2.\{z\} \land \exists R_3.\{z\} \subseteq \exists R.\{z\}.$$ 

If $a_1, \ldots, a_k$ are all the known individuals in the knowledge base, then this axiom can also be expressed using the $k$ **SROIQ**-axioms

$$\exists R_1.\exists R_2.\{a_i\} \land \exists R_3.\{a_i\} \subseteq \exists R.\{a_i\}$$

where $i$ ranges from 1 to $k$. This kind of conversion, called full or naive grounding, of nominal schemas into classical description logics is, however, computationally infeasible [5] even for **ELROV**$_n$, which is of PTime complexity [17]. In [5], we thus presented a datalog-based algorithm for **ELROV**$_n$ which avoids full grounding, and have also shown experimentally that the algorithm is efficient.

### 3 Approximation

For our approximation of **SROIQ** by **ELROV**$_n$, we use a number of different techniques, some of which are borrowed from existing literature. The key ideas are as follows.

- We rewrite mincardinality restrictions into maxcardinality restrictions or approximate using an existential.
- We rewrite universal quantification into existential quantification.
- We approximate maxcardinality restrictions using functionality.
- We approximate inverse roles and functionality using nominal schemas.
- We approximate negation using class disjointness.
Algorithm 1 Approximation Algorithm

1: normalize the $SROIQ$ KB into normal forms;
2: for each concept $C$ do
3:    introduce a fresh concept $\neg(C)$;
4:    add axiom $C \sqcap \neg(C) \sqsubseteq \bot$;
5: end for
6: for each role $R^{-}$ appearing in KB do
7:    introduce a fresh role $\text{inv}(R)$;
8:    add $\{x\} \sqcap \exists R.\{y\} \sqsubseteq \{y\} \sqcap \exists \text{inv}(R).\{x\}$;
9: end for
10: for each axiom $a$ in TBox do
11:    if $a$ is of type $A \sqsubseteq C$ then
12:        add axiom $\neg(A) \sqsubseteq \neg(C)$;
13:    else if $a$ is of type $A \sqsubseteq B \sqcup C$ then
14:        add axiom $\neg(B) \sqcap \neg(C) \sqsubseteq \neg(A)$;
15:    else if $a$ is of type $A \sqsubseteq \forall R.C$ then
16:        add axiom $\exists R.\neg(C) \sqsubseteq \neg(A)$;
17:        add axiom $\exists \text{inv}(R).A \sqsubseteq C$ and $\{x\} \sqcap \exists R.\{y\} \sqsubseteq \{y\} \sqcap \exists \text{inv}(R).\{x\}$;
18:    else if $a$ is of type $\forall R.A \sqsubseteq C$ then
19:        add axiom $\neg(C) \sqsubseteq \exists R.\neg(A)$;
20:    else if $a$ is of type $C \sqsubseteq nR.A$ then
21:        add axiom $\neg(C) \sqsubseteq \exists R.A$;
22:    else if $a$ is of type $C \sqsubseteq \leq nR.A$ then
23:        add axiom $\neg(C) \sqcap \exists R.\{(z1) \sqcap A\} \sqcap \exists R.\{(z2) \sqcap A\} \sqsubseteq \exists U.((\{z1\} \sqcap \{z2\})$;
24:    else if $a$ is of type $\leq nR.A \sqsubseteq C$ then
25:        add axiom $\neg(C) \sqsubseteq \exists R.A$;
26:    else if $a$ is of type $\geq nR.A \sqsubseteq C$ then
27:        add axiom $\neg(C) \sqcap \exists R.\{(z1) \sqcap A\} \sqcap \exists R.\{(z2) \sqcap A\} \sqsubseteq \exists U.((\{z1\} \sqcap \{z2\})$;
28:    else
29:        add axiom $a$;
30: end if
31: end for

– We approximate disjunction using conjunction.

A pseudocode description is given in Algorithm 1, we explain the relevant parts in more detail below. Role chain axioms are left untouched, as are axioms which can already directly be expressed in $\mathcal{ELROV}_n$. We drop the soundness proof, since one can easily find out that our approach is sound but incomplete.

3.1 Approximation of Inverse Role and Functionality

Since $\mathcal{ELROV}_n$ can express DL-safe Datalog rules, all rule-like axioms in $SROIQ$ can be approximated easily in $\mathcal{ELROV}_n$.

For role inclusion axioms of the form $R \sqsubseteq S^{-}$, the first-order logic rule is $R(x, y) \rightarrow S(y, x)$. By restricting the variables to nominals, we obtain $\text{nom}(x) \land \text{nom}(y) \land R(x, y) \rightarrow S(y, x)$, where nom(x) is defined by the collection of facts...
nom(a_i) for each individual a_i. The latter rule can be expressed by means of the nominal schema axiom,
\{x\} \cap \exists R.\{y\} \sqsubseteq \{y\} \cap \exists S.\{x\}
where x and y are nominal schemas. This axiom will be later translated into datalog rule,
nom(x), nom(y), triple(x, R, y) \rightarrow triple(y, S, x)
where we can clearly see that the rule expresses the inverse role with restricting variable bounded to known individuals.

Similarly, for a functionality axiom \( C \sqsubseteq 1 \cdot R \cdot D \), we can cast it into
\( C \sqcap \exists R.\{\{z\} \sqcap \{D\}\} \sqsubseteq \exists U.\{\{z\} \sqcap \{2\}\} \)
where \( U \) is the universal role. This axiom will be translated into two datalog rules:
\( nom(z1), nom(z2), inst(x, C), inst(x, D), triple(x, R, z1), triple(x, R, z2) \rightarrow inst(z1, z2) \)
\( nom(z1), nom(z2), inst(x, C), inst(x, D), triple(x, R, z1), triple(x, R, z2) \rightarrow inst(z2, z1) \)
Briefly, it means if there are two triples \( triple(x, R, z1) \) and \( triple(x, R, z2) \), then \( z1 \) and \( z2 \) must be same. (See details of translation in [5].)

Since \( A \sqsubseteq \forall R \cdot C \) is the same as \( \exists R^-.A \sqsubseteq C \), we can approximate \( A \sqsubseteq \forall R \cdot C \) by adding
\( \exists inv(R).A \sqsubseteq C \)
and
\( \{x\} \cap \exists R.\{y\} \sqsubseteq \{y\} \cap \exists inv(R).\{x\}. \)

Furthermore, for each axiom \( A \sqsubseteq n \cdot R \cdot C \), we reduce it to \( A \sqsubseteq 1 \cdot R \cdot C \), such that it can be approximated through the nominal schema axiom
\( A \cap \exists R.\{x\} \sqcap \exists R.\{y\} \sqsubseteq \exists U.\{\{x\} \sqcap \{2\}\} \).

3.2 Approximation of Negation and Disjunction

Our approach for approximating negation is derived from [23]. In brief, we add a fresh concept \( neg(C) \) for each concept \( C \) in KB, and add the axiom \( neg(C) \cap C \sqsubseteq \bot \) to express that the negation of \( C \) and \( C \) are disjoint. Furthermore, we rewrite the following axioms by using their De Morgan equivalent axioms and replace \( \neg C \) by the fresh concept \( neg(C) \).

\begin{align*}
(1) & \quad A \sqsubseteq B \sqcup C \Rightarrow \neg B \cap \neg C \sqsubseteq \neg A \Rightarrow neg(C) \sqsubseteq neg(A) \\
(2) & \quad A \sqsubseteq \forall R \cdot C \Rightarrow \exists R.\neg C \sqsubseteq \neg A \Rightarrow \exists R.neg(C) \sqsubseteq neg(A) \\
(3) & \quad \forall R.A \sqsubseteq C \Rightarrow \neg C \sqsubseteq \exists R.\neg A \Rightarrow neg(C) \sqsubseteq \exists R.neg(A) \\
(4) & \quad \leq n \cdot R \cdot A \sqsubseteq C \Rightarrow \neg C \sqsupset n \cdot R \cdot A \Rightarrow neg(C) \sqsupset n \cdot R \cdot A \\
(5) & \quad \geq n \cdot R \cdot A \sqsubseteq C \Rightarrow \neg C \sqsubset n \cdot R \cdot A \Rightarrow neg(C) \sqsubset n \cdot R \cdot A
\end{align*}
Table 2. Evaluation ontologies for our algorithm

<table>
<thead>
<tr>
<th>Ontology</th>
<th>Classes</th>
<th>Annotation P.</th>
<th>Data P.</th>
<th>Object P.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rex⁵</td>
<td>552</td>
<td>10</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>Spatial⁴</td>
<td>106</td>
<td>13</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>Xenopus⁵</td>
<td>710</td>
<td>19</td>
<td>0</td>
<td>5</td>
</tr>
</tbody>
</table>

Note that we can always reduce \( C \sqsupseteq n R.A \) to \( C \sqsubseteq \exists R.A \). Then, for the last two axioms (4) and (5), we reduce them to \( neg(C) \sqsubseteq \exists R.A \) and \( neg(C) \sqsubseteq 1 R.A \). Following the ideas in [12,27], for \( A \sqsubseteq B \sqcup C \), it can be reduced to \( A \sqsubseteq B \sqcap C \), i.e., \( A \sqsubseteq B \) and \( A \sqsubseteq C \), falling into unsound but complete results. We will attempt to combine this idea with the approach in the paper. Briefly, combining unsound results and incomplete results to achieve higher precise and recall.

4 Reasoning over \( E L R O V_n \)

We briefly recall the algorithm for reasoning over \( E L R O V_n \) presented in [5], and the evaluation results presented therein. The algorithm actually imposes some restrictions on \( E L R O V_n \) which are described in detail in [5] and which cause no problem for our approximation approach.

The algorithm itself is based on results presented in [15]. Following this approach, for every \( E L R O V_n \) knowledge base \( KB \) we can construct a Datalog program \( P_{KB} \) that can be used for reasoning over \( KB \). The Datalog program \( P_{KB} \) contains facts which are translated from all the DL normal forms (Figure 1) and rules (Figure 2). [5] contains a correctness proof.

The evaluation reported in [5] was performed using the Java-based Datalog reasoner IRIS² [3], and we compared it to a full grounding approach for which we also used IRIS. We used suitable ontologies from the TONES repository, see Table 2 for some basic metrics, and artificially added named individuals and axioms using nominal schemas. Results are listed in Table 3. In our approach, the number of nominal schemas per axioms had almost no effect on the runtime, thus indicating that the approach performs very well indeed.

² http://iris-reasoner.org/
³ http://obo.cvs.sourceforge.net/checkout/obo/obo/ontology/physicochemical/physicochemical\_rei.\_obo
⁴ http://obo.cvs.sourceforge.net/checkout/obo/obo/ontology/anatomy/caro/spatial.\_obo
Table 3. Evaluation, IRIS reasoning time listed only (no pre-processing, no load time), in ms. The "No ns" column refers to the running with no nominal schemas, while k ns refers to the use of k nominal schemas in an axiom. Times in brackets are for full grounding, for comparison. If not listed, full grounding was OOM (Out of Memory)

<table>
<thead>
<tr>
<th>Ontology</th>
<th>number of individuals</th>
<th>no ns</th>
<th>1 ns</th>
<th>2 ns</th>
<th>3 ns</th>
<th>4 ns</th>
<th>5 ns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rex</td>
<td>100</td>
<td>263</td>
<td>263 (321)</td>
<td>267 (972)</td>
<td>273</td>
<td>275</td>
<td>259</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>480</td>
<td>518 (1753)</td>
<td>537 (OOM)</td>
<td>538</td>
<td>545</td>
<td>552</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>2904</td>
<td>2901 (133179)</td>
<td>3120 (OOM)</td>
<td>3165</td>
<td>3192</td>
<td>3296</td>
</tr>
<tr>
<td>Spatial</td>
<td>100</td>
<td>22</td>
<td>191 (222)</td>
<td>201 (1163)</td>
<td>198</td>
<td>202</td>
<td>207</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>134</td>
<td>417 (1392)</td>
<td>415 (OOM)</td>
<td>421</td>
<td>431</td>
<td>432</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>1322</td>
<td>1792 (96437)</td>
<td>1817 (OOM)</td>
<td>1915</td>
<td>1888</td>
<td>1997</td>
</tr>
<tr>
<td>Xenopus</td>
<td>100</td>
<td>62</td>
<td>332 (383)</td>
<td>284 (1629)</td>
<td>311</td>
<td>288</td>
<td>280</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>193</td>
<td>538 (4751)</td>
<td>440 (OOM)</td>
<td>430</td>
<td>456</td>
<td>475</td>
</tr>
<tr>
<td></td>
<td>10000</td>
<td>1771</td>
<td>2119 (319013)</td>
<td>1843 (OOM)</td>
<td>1886</td>
<td>2038</td>
<td>2102</td>
</tr>
</tbody>
</table>

5 Implementation and Evaluation

We realized the implementation based on the $\mathcal{ERLOV}_n$ datalog-based reasoner [5]. All experiments were conducted on a laptop with a 2.4GHz Intel Core i7-3630QM processor and 8GB RAM operated by Windows 7 64-bit system with Java VM v.1.7.0. We set time out of 1 hour and Java heap space of 1GB. The ontologies were chose from Oxford Ontologies Repository 6, in Table Table 4. To evaluate the performance in practice, we also compared with mainstream reasoners Pellet 2.3.07, FaCT++ 1.6.28 and HermiT 1.3.79. The reasoning task is classification, therefore recall equals the number of subsumption relations between concepts divides its correct number. Since our approach needs some individual to fire the datalog rules, we add one unique dummy individual for each concepts if the testing ontology does not contain individuals. Therefore, we can check subsumption relations by tracking those dummy individuals.

The experiment, Table 5 , shows our approach has good recalls but fails when conducting very large ontologies. The reason is that IRIS reasoner has a difficulty to run with large number of rules or facts. However, with a quicker datalog reasoner or a more efficient reasoner that supports nominal schemas, we believe it will achieve a better result. Also, since the number of rules (Figure 2) are fixed, we do not need a full powerful Datalog reasoner. We can specifically program the rules to improve the efficiency.

To be noticed, the approximation in this paper can be done by HermiT reasoner since HermiT can handle $DL$-safe rules and the rules can directly be

6 http://www.cs.ox.ac.uk/isg/ontologies/
7 http://clarkparsia.com/pellet/
8 http://owl.man.ac.uk/factplusplus/
9 http://www.hermi-reasoner.com/
```
C(a) ↦→ {subClass(a, D)}  R(a, b) ↦→ {subEx(a, R, b, b)}
⊤ ⊑ C ↦→ {top(C)}  A ⊑ ⊥ ↦→ {bot(A)}
{a} ⊑ C ↦→ {subClass(a, C)}  A ⊑ {c} ↦→ {subClass(A, c)}
A ⊑ C ↦→ {subclass(A, C)}  A ∩ B ⊑ C ↦→ {subConj(A, B, C)}
∃R. Self ⊑ C ↦→ {subSelf(R, C)}  A ⊑ ∃R. Self ↦→ {supSelf(A, R)}
R ⊑ C × D ↦→ {supProd(R, C, D)}
```

**Fig. 1.** Input Translation \( I_V \)

**Table 4.** Evaluation ontologies for our algorithm, the No. denotes the number order of Oxford Repository. Since the implementation does not support datatype property, any ontologies containing datatype properties are not chosen here.

<table>
<thead>
<tr>
<th>No.</th>
<th>Ontology</th>
<th>expressivity</th>
<th>Classes</th>
<th>Object Properties</th>
<th>Individuals</th>
<th>TBox</th>
<th>RBox</th>
<th>ABox</th>
</tr>
</thead>
<tbody>
<tr>
<td>00004</td>
<td>BAMS</td>
<td>SHFL</td>
<td>1110</td>
<td>12</td>
<td>0</td>
<td>18813</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>00015</td>
<td>DOLCE</td>
<td>SHFL</td>
<td>37</td>
<td>70</td>
<td>0</td>
<td>279</td>
<td>76</td>
<td>0</td>
</tr>
<tr>
<td>00039</td>
<td>GALEN</td>
<td>ALEHIF</td>
<td>3097</td>
<td>403</td>
<td>0</td>
<td>14661</td>
<td>111</td>
<td>0</td>
</tr>
<tr>
<td>00040</td>
<td>GO</td>
<td>SRIQ</td>
<td>58882</td>
<td>220</td>
<td>0</td>
<td>130376</td>
<td>104</td>
<td>0</td>
</tr>
<tr>
<td>00103</td>
<td>Gardiner</td>
<td>ALEQ</td>
<td>176</td>
<td>10</td>
<td>187</td>
<td>263</td>
<td>0</td>
<td>187</td>
</tr>
<tr>
<td>00387</td>
<td>OBO</td>
<td>SRIQ</td>
<td>27412</td>
<td>215</td>
<td>34</td>
<td>55998</td>
<td>87</td>
<td>157897</td>
</tr>
</tbody>
</table>

added to the input ontology in functional style. But, HermiT doesn’t have specific reasoning procedure for \( \mathcal{EL} \)-families, such that reasoning for \( \mathcal{EL} \) is not its advantage. Moreover, there are \( \mathcal{ELROV}_n \)axioms which cannot be expressed as \( DL\)-safe rules, e.g., \( ∃R.\{z\} ⊑ ∃T.∃S.\{z\} \). Moreover,

### 6 Conclusions and Future Work

We have described an approximate reasoning procedure for \( SROIQ \) which utilizes the tractable nominal-schemas-based \( \mathcal{ELROV}_n \) using a language weakening approach. We have also provided an experimental evaluation which shows the feasibility of this setting.

Going forward, there are several directions which we intend to explore. On the one hand, we will be looking into variants on how to obtain the weakened language, in the spirit of [27], and will attempt to further tweak and optimize our approach. On the one hand, we will be looking into incremental methods
Fig. 2. Deduction Rules $P_V$
Table 5. Evaluation, reasoning time of each reasoner, in ms. N/A denotes that the datalog-based reasoner corrupts with too many loading rules.

<table>
<thead>
<tr>
<th>Ontology</th>
<th>HermiT</th>
<th>Fact++</th>
<th>Pellet</th>
<th>Ours</th>
<th>Ours Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>BAMS</td>
<td>3</td>
<td>2</td>
<td>10</td>
<td>107</td>
<td>100%</td>
</tr>
<tr>
<td>DOLCE</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>53</td>
<td>100%</td>
</tr>
<tr>
<td>GALEN</td>
<td>4</td>
<td>2</td>
<td>17</td>
<td>7840</td>
<td>90.8%</td>
</tr>
<tr>
<td>GO</td>
<td>36</td>
<td>75</td>
<td>59</td>
<td>N/A</td>
<td>N/A</td>
</tr>
<tr>
<td>GardinerCorpus</td>
<td>14</td>
<td>6</td>
<td>17</td>
<td>89</td>
<td>92.3%</td>
</tr>
<tr>
<td>OBO</td>
<td>34</td>
<td>61</td>
<td>139</td>
<td>N/A</td>
<td>N/A</td>
</tr>
</tbody>
</table>

which use the approximate reasoning results as starting point and subsequently compute correct results in all or at least most cases.

Acknowledgements This work was supported by the National Science Foundation under award 1017225 III: Small: TROn – Tractable Reasoning with Ontologies.

References


