# Soundness Preserving Approximation for TBox Reasoning in $\mathcal{R}$

Ren Yuan, Jeff Z. Pan and Yuting Zhao

Dept. of Computing Science, University of Aberdeen
King's College, Aberdeen AB24 3FX, UK

**Abstract.** TBox reasoning in description logics is hard. For example, reasoning in $\mathcal{SROIQ}$ (i.e. OWL2-DL) is N2ExpTime-complete; even with $\mathcal{R}$, a fragment of $\mathcal{SROIQ}$ supporting $\mathcal{ALC}$ GCIs and role chains, the complexity of reasoning is 2ExpTime-hard. Although various optimisation techniques have been applied, existing tableau-based DL reasoners are still inefficient in dealing with arbitrary GCIs especially when complex role chains present. In this paper, we present a soundness preserving approximation for TBox reasoning in $\mathcal{R}$. The main idea is to convert $\mathcal{R}$ ontologies to $\mathcal{EL}^{+}$ with an additional complement table maintaining the complementary relations between named concepts. Since existing benchmarks do not focus on complex GCIs and RIs, we propose a new set of testing ontologies for TBox reasoning in $\mathcal{R}$ and our preliminary evaluation shows that a naive implementation of our complement-integrated TBox reasoning algorithm outperforms existing reasoners on most of these ontologies.

## 1 Introduction

The family of the description logics (DL) provides a wide range of formalisms with a trade-off between expressiveness and computational difficulty. TBox reasoning in description logics is hard; e.g., DL $\mathcal{SROIQ}$ [6], the adjacent logic of OWL2-DL is N2ExpTime-complete [12]; even with $\mathcal{R}$ (following the notation of $\mathcal{RIQ}$ in [7]), a fragment of $\mathcal{SROIQ}$ supporting $\mathcal{ALC}$ GCIs and role chains, the complexity of reasoning is 2ExpTime-hard [12]. This makes up a major obstacle of applying these expressive languages in large scale systems.

There are mainly two approaches to providing efficient reasoning services. The first approach investigates sophisticated optimisation techniques for tableaux algorithms [5, 8, 9, 11, 22]. Their achievements have been applied in practical DL reasoners such as FaCT++ [21], Pellet [18], Racer [3] and HermiT [17]. One of the major difficulties for tableau algorithms is the high degree of nondeterminism introduced by GCIs. Some techniques, such as *absorption* [11, 22] can reduce GCIs into non-GCIs; however, they are only applicable on some kinds of GCIs.

The second approach is based on transformations. Approximations can be seen as some useful transformations. Selman and Kautz [16] propose the idea of finding the least upper bound and most lower bound for the entire knowledge base or a concept. In practise, such bounds are usually computed in either syntactic or semantic way. *Syntactic Approximation* [2, 4, 15, 19, 23] weaken the ontology into a less expressive DL, while *Semantic Approximation* [14] apply the idea of *Knowledge Compilation* [16] to precompute the entailed axioms. However, naive syntactic approximations can not guarantee the quality, i.e., soundness or completeness of reasoning. Semantic approximation requires the reasoning in the source language, which might require significant preprocessing time. KAON2 [13] reasoner rewrites $\mathcal{SHIQ}(D)$ ontologies into disjunctive datalog programs [10]. It is more dedicated to reasoning with large ABox and the reduction is exponential.

We notice that less expressive DLs, such as $\mathcal{EL}^+$ can deal with large amount of GCIs of restricted patterns very efficiently and its inference patterns can be partially generalised to more expressive DLs by syntactic manipulation of TBox. In this paper, we present a soundness preserving syntactic approximation for TBox reasoning services of the DL $\mathcal{R}$. More precisely, we syntactically approximate an $\mathcal{R}$ TBox to $\mathcal{EL}^+$ [1] TBox. Compared with existing works, we propose a new approximation method together with new reasoning algorithms having the following features:

1. It's a syntax-based approximation, which can be conducted very efficiently.
2. It preserves the complementary relations among concept names with an additional complement table.
3. Soundness-guaranteed TBox reasoning of resulting ontologies can be done in polynomial time.

We implemented a prototype of our approach called REL and evaluate its performance on real world benchmarking ontology and a set of automated generated $\mathcal{R}$ TBox. Preliminary evaluations show that (1) REL outperforms tableaux based reasoners such as Pellet and FaCT++, and (2) its soundness preserving approximation service provides rather complete results, i.e. with high recall.

The rest of the paper is organised as follows: in Section 2 we present our approximation approach with examples; in Section 3 we present a soundness-guaranteed TBox reasoning algorithm for the approximation ontologies; in Section 4 an evaluation of our prototypical implementation is conducted to compare with existing DL reasoners; in Section 5 we summarise our findings and highlight the potential of this study.

## 2 Approximation Approach

In this section, we present our approximation approach by first showing approximation principles with an example, and then formalising these principles into algorithms.

### 2.1 Principles

$\mathcal{R}$ supports two types of TBox axioms: general concept inclusion (GCIs) and complex role inclusion (RIs). Because RIs are the same in $\mathcal{R}$ and $\mathcal{EL}+$, and they are independent from GCIs, we preserve all the RIs in the approximation.

The approximation of the GCIs is mainly about the approximation of concept expressions. In $\mathcal{R}$, a concept is inductively defined by constructs as follows:

$$C := \top|\bot|A|\neg C|C \sqcap D|C \sqcup D|\exists r.C|\forall r.C$$

while in $\mathcal{EL}^+$, a concept is inductively defined by constructs as:

$$C := \top|A|C \sqcap D|\exists r.C$$

where $A$ is an atomic concept, $r$ is a role name.

Intuitively, an $\mathcal{R}$ construct belongs to one of the following sets: $\{\top, \bot\}$, $\{A, \neg C\}$, $\{C \sqcap D, C \sqcup D\}$, $\{\exists r.C\}$, $\{\forall r.C\}$. It's easy to observe that the former construct in each set is $\mathcal{EL}^+$-compatible, and the latter is equivalent to some negation of the former, which means for an arbitrary $\mathcal{R}$ concept expression, either itself or its complement can be composed by legal $\mathcal{EL}^+$ constructs. So we can represent non-$\mathcal{EL}^+$ part by replacing its appearance with a concept name and preserve its semantics by adding axioms defining its complement. From these observations we generalise our principles of approximation as follows:

1. Representing non-$\mathcal{EL}^+$ parts with new concept names. And, particularly, using a symbol $\perp\!\!\!\perp$ to represent the unique approximation of $\bot$.
2. Maintaining the semantics of non-$\mathcal{EL}^+$ concepts by definitions of their complements.
3. Using an additional complement table $CT$ to maintain the complementary relations between concept names.
4. Preserving all the *RI*s.
5. Asserting additional subsumptions in reasoning to recover the semantics of approximated concept expressions.

An example is as follows:

*Example 1.* $\mathcal{T} = \{Koala \sqsubseteq \forall eat.EucalyptLeave,\ EucalyptLeave \sqsubseteq VegetarianFood,\ \forall eat.VegetarianFood \sqsubseteq Herbivore\}$

Following principle 1, we represent non-$\mathcal{EL}^+$ concepts $\forall eat.EucalyptLeave$ and $\forall eat.VegetarianFood$ by new names $eatEucalyptLeave$ and $eatVegetarianFood$, respectively. Following principle 2, we define $neatEucalyptLeave \equiv \exists eat.nEucalyptLeave$ and $neatVegetarianFood \equiv \exists eat.nVegetarianFood$ to represent the complements of $eatEucalyptLeave$ and $eatVegetarianFood$ respectively. Then recursively, we introduce $nEucalyptLeave$ and $nVegetarianFood$ to represent non-$\mathcal{EL}^+$ concepts $\neg EucalyptLeave$ and $\neg VegetarianFood$ respectively. Thus the TBox becomes:

$\mathcal{T}' = \{eatVegetarianFood \sqsubseteq Herbivore, Koala \sqsubseteq eatEucalyptLeave, EucalyptLeave \sqsubseteq VegetarianFood,\ neatEucalyptLeave \equiv \exists eat.nEucalyptLeave,\ neatVegetarianFood \equiv \exists eat.nVegetarianFood\}$

Following principle 3, we build

$CT = \{(eatEucalyptLeave, neatEucalyptLeave), (EucalyptLeave, nEucalyptLeave), (eatVegetarianFood, neatVegetarianFood), (VegetarianFood, nVegetarianFood)\}$

Following principle 5, in such a knowledge base, reasoning can infer $Koala \sqsubseteq Herbivore$ as follows:

*EucalyptLeave* $\sqsubseteq$ *VegetarianFood* $\rightarrow$ *nVegetarianFood* $\sqsubseteq$ *nEucalyptLeave* $\rightarrow$ *neatVegetarianFood* $\sqsubseteq$ *neatEucalyptLeave* $\rightarrow$ *eatEucalyptLeave* $\sqsubseteq$ *eatVegetarianFood* $\rightarrow$ *Koala* $\sqsubseteq$ *Herbivore*.

### 2.2 Algorithms

Given an $\mathcal{R}$ TBox $\mathcal{T}$, we first generate a set $S(\mathcal{T})$ of concept expressions appearing in $\mathcal{T}$ as follows:

1. Initialise $S(\mathcal{T})$ by an $\{\top\}$.
2. If $C$ is refereed in $T$, then add $C$ into $S(\mathcal{T})$.
3. For each $C \in S(\mathcal{T})$, add $\neg C$ into $S(\mathcal{T})$.
4. For each $C \in S(\mathcal{T})$, if $C$ is a conjunction (disjunction), add all its conjuncts (disjuncts) into $S(\mathcal{T})$; if $C$ is a existential (universal) restriction, add its filler into $S(\mathcal{T})$.
5. Go back to 3 and repeat until no more changes can be made.

We then assign names to these concepts by a function $n$ which assigns each atomic concept in $S(\mathcal{T})$ (including $\top$) to itself, and each complex concept expression a unique name that does not appear in $S(\mathcal{T})$.

We further define these names by a function $d$ which assigns each conjunction $C \equiv \bigsqcap C_i \in S(\mathcal{T})$ in $S(\mathcal{T})$ an axiom $n(C) \equiv \bigsqcap n(C_i)$, each existential restriction $C \equiv \exists r.D$ in $S(\mathcal{T})$ an axiom $n(C) \equiv \exists r.n(D)$.

With these names and definitions, we approximate $\mathcal{T}$ by Algorithm A-1. Its input is an $\mathcal{R}$ TBox $\mathcal{T}$. Its output is $(AS, CT)$ with $AS$ an $\mathcal{EL}^{+}$ TBox and $CT$ a set of paired concept names.

---

**Algorithm A-1**: OntoApprox($\mathcal{T}$)

1: $AS := \emptyset$
2: $CT := \emptyset$
3: **for** each GCI $C \sqsubseteq D \in \mathcal{T}$ **do**
4:     $AS := AS \cup \{n(C) \sqsubseteq n(D)\}$
5: **end for**
6: **for** each concept $C \in S(\mathcal{T})$ **do**
7:     $CT := CT\{(n(C), n(\neg C)), (n(\neg C), n(C))\}$
8:     **if** $C$ is a conjunction or existential restriction **then**
9:         $AS := AS \cup \{d(C)\}$
10:     **end if**
11: **end for**
12: **for** each RI $\beta \in \mathcal{T}$ **do**
13:     $AS := AS \cup \{\beta\}$
14: **end for**
15: normalise $AS$

---

This algorithm needs some explanation:

- By step-1 and step-2, $AS$ and $CT$ are initialised by empty sets.
- Step-4 rewrite the GCI with the named concepts.

– Step-7 updates $CT$.
– Step-8 and 9 maintain the definition of some concepts.
– Step-13 preserve all the RIs.
– Step-15 normalise $AS$ as a classical $\mathcal{EL}^+$ ontology [1]. It's important to point out here that such a normalisation will not introduce any new concept name, because in step-4, all the GCIs have already been approximated into form $A \sqsubseteq B$ or $A \equiv C$, where $A, B$ are concept names and $C$ is either $A_1 \sqcap \ldots \sqcap A_n$ or $\exists r.A$.

After the execution of A-1, $AS$ is a normalised $\mathcal{EL}^+$ ontology. For every concept name $A \in CN_{AS}$, there exists $B$ such that $(A, B) \in CT$ or $(B, A) \in CT$. The complexity of A-1 is described by the following theorem:

**Theorem 1.** *Given an $\mathcal{R}$ TBox, $\mathcal{T}$ and $N_{\alpha,\mathcal{T}}$ the number of axioms in $\mathcal{T}$, Algorithm A-1 will terminate in $O(N_{\alpha,\mathcal{T}})$ time in worst case.*

*Proof.* Algorithm A-1 is linearly w.r.t. $N_{\alpha,\mathcal{T}} + |S(\mathcal{T})|$, where $|S(\mathcal{T})|$ is also linear w.r.t. $N_{\alpha,\mathcal{T}}$.

We call the pair of $(AS, CT)$ an $\mathcal{EL}^+_C$ ontology to indicate that it is an $\mathcal{EL}^+$ ontology plus a complement table. As Algorithm A-1 shows, $CT$ actually contains pairs of complementary named concepts. In the following, for each $A$ appear in $(AS, CT)$, we use $CT(A)$ to represent the complement of $A$, i.e. $(A, CT(A)) \in CT$.

Following Theorem 1 and the algorithms, we immediately know that $|AS| = O(N_{\alpha,\mathcal{T}})$ and $|CT| = O(N_{\alpha,\mathcal{T}})$. Also, the approximation is additive, which means $OntoApprox(\mathcal{T}_1 \cup \mathcal{T}_2) = OntoApprox(\mathcal{T}_1) \cup OntoApprox(\mathcal{T}_2)$ when any concept expression $C$ has the same $n(C)$ in these three approximations.

## 3 Soundness-preserving $\mathcal{EL}^+_C$ TBox Reasoning

We define entailment in an $\mathcal{EL}^+_C$ ontology $O = (AS, CT)$ as: $O \models \alpha$ *iff* $AS \cup \{A \equiv \neg B | (A, B) \in CT\} \models \alpha$. Given $CN_{AS}$ the set of concept names, $RN_{AS}$ the set of role names, TBox reasoning in $O$ yields, for each $C \in CN_{AS}$, a subsumer set $S(C) = \{X | O \models C \sqsubseteq X\}$, for each $r \in RN_{AS}$, a relation set $R(r) = \{(X, Y) | O \models X \sqsubseteq \exists r.Y\}$.

### 3.1 Completion rules

TBox reasoning in $AS$ alone can be done by $\mathcal{EL}^+$ classification [1]. However, due to the absence of knowledge maintained by $CT$, performing $\mathcal{EL}^+$ reasoning in $AS$ without considering $CT$ will lose much information. We therefore propose several additional completion rules to capture the semantics of $CT$ and the $\perp\!\!\!\perp$ as shown in Table 1.

**R6** realises axiom $A \sqcap \neg A \sqsubseteq \bot$. **R7** asserts the reverse subsumption between concepts to supplement the absence of negation. **R8** builds up the relations between conjuncts of a conjunction, *e.g.* $A \sqcap B \sqsubseteq \bot$ implies $A \sqsubseteq \neg B$. **R9** deals with $\bot$ in existential restrictions, *e.g.* $A \sqsubseteq \exists r.\bot$ implies $A \sqsubseteq \bot$.

We show the application of some rules with the following example:

**Table 1.** Additional completion rules

| |
|---|
| **R6** If $A, B \in S(X)$, $A = CT(B)$ and $⫫ \notin S(X)$ then $S(X) := S(X) \cup \{⫫\}$ |
| **R7** If $A \in S(B)$ and $CT(B) \notin S(CT(A))$ then $S(CT(A)) := S(CT(A)) \cup \{CT(B)\}$ |
| **R8** If $A_1 \sqcap \ldots \sqcap A_i \sqcap \ldots \sqcap A_n \sqsubseteq ⫫$, $A_1, \ldots, A_{i-1}, A_{i+1}, \ldots, A_n \in S(X)$ and $CT(A_i) \notin S(X)$ then $S(X) := S(X) \cup \{CT(A_i)\}$ |
| **R9** If $(A, ⫫) \in R(r)$ and $⫫ \notin S(A)$ then $S(A) := S(A) \cup \{⫫\}$ |

*Example 2.* $\mathcal{O} = \{A \sqsubseteq \forall r.\exists s.B,\ B \sqsubseteq \bot,\ \neg C \sqsubseteq \exists r.\top\}$

Obviously, we can infer $A \sqsubseteq C$ from the above TBox.

*OntoApprox*((*O*)) will yield the following results:

$AS = \{X_2 \equiv \exists s.B,\ X_5 \equiv \exists r.X_3,\ A \sqsubseteq X_4,\ B \sqsubseteq ⫫,\ X_8 \equiv \exists r.\top,\ X_7 \sqsubseteq X_8\ \}$ and

$CT = \{(B, X_1),\ (X_1, B),\ (X_2, X_3),\ (X_3, X_2),\ (X_4, X_5),\ (X_5, X_4),\ (A, X_6),\ (X_6, A),\ (⫫, \top),\ (\top, ⫫),\ (C, X_7),\ (X_7, C),\ (X_8, X_9),\ (X_9, X_8)\}$

Intuitively, $A \sqsubseteq C$ can be inferred by reasoning in following steps:

1. $X_2 \equiv \exists s.B, B \sqsubseteq ⫫ \rightarrow X_2 \equiv \exists s.\ ⫫ \rightarrow_{R9} X_2 \sqsubseteq ⫫ \rightarrow_{R7} \top \sqsubseteq X_3$;
2. $\top \sqsubseteq X_3, X_8 \equiv \exists r.\top \rightarrow X_8 \sqsubseteq \exists r.X_3$;
3. $X_8 \sqsubseteq \exists r.X_3, X_5 \equiv \exists r.X_3 \rightarrow X_8 \sqsubseteq X_5$;
4. $X_8 \sqsubseteq X_5, X_7 \sqsubseteq X_8 \rightarrow X_7 \sqsubseteq X_5 \rightarrow_{R7} X_4 \sqsubseteq C$;
5. $X_4 \sqsubseteq X_C, A \sqsubseteq X_4 \rightarrow A \sqsubseteq C$

### 3.2 Abstract algorithm

The extra completion rules can be considered as introductions of new normalised $\mathcal{EL}^+$ axioms in addition to $AS$. For example, **R7** introduces a new axiom $CT(A) \sqsubseteq CT(B)$ given known subsumption $B \sqsubseteq A$. In the following we propose an abstract algorithm which performs $\mathcal{EL}^+_C$ TBox reasoning by realising the completion rules with incremental reasoning of $\mathcal{EL}^+$ [20].

The algorithm computes these sets by processing corresponding axioms. For each name concept or existential restriction appearing on LHS of some axiom, the algorithm maintains a Ô set while for each name concept $A$, the algorithm maintains a FIFO *queue*($A$).

Given a $\mathcal{EL}^+_C$ ontology $\mathcal{O} = (AS, CT)$, the algorithm is initialised as follows:

1. $\forall C \in CN_{AS}, S(C) = \{C\} \cup \{\top\}, S(⫫) = CN_{AS}$;
2. $\forall r \in RN_{AS}, R(r) = \emptyset$;
3. if $A_1 \sqcap \ldots \sqcap A_n \sqsubseteq B \in AS$, then add $A_1, \ldots, A_{i-1} \sqcap A_{i+1}, \ldots, A_n \rightarrow B$ into $\hat{O}(A_i)$;
4. if $A \sqcap \exists r.B \in AS$, then add $\exists r.B \in$ into $\hat{O}(A)$;
5. if $\exists r.A \sqcap B \in AS$, then add $B \in$ into $\hat{O}(\exists r.A)$;

6. $\forall A \in CN_{AS}$, queue($A$)=$\hat{O}(A) \cup \hat{O}(\top)$;

Then, for all $A \in CN_{AS}$ such that $\perp\!\!\!\perp \notin S(A)$, Algorithm A-2 is applied to all the entries $X \in$queue($A$) until no more changes can be made.

---

**Algorithm A-2**: Process($A, X$)

1: **if** $X = B_1, \ldots, B_n \rightarrow B$ and $B \notin S(A)$ **then**
2:     **if** $B_1, \ldots, B_n \in S(A)$ **then**
3:         AddSubsumer($A$,$B$)
4:     **end if**
5:     **if** $\perp\!\!\!\perp \in S(B)$ **then**
6:         **for** $CT(B_i) \notin S(A)$ and $B_1, \ldots, B_{i-1}, B_{i+1} \ldots, B_n \in S(A)$ **do**
7:             AddSubsumer($A$,$CT(B_i)$)
8:         **end for**
9:     **end if**
10: **end if**
11: **if** $X = \exists r.B$ **then**
12:     **if** $\perp\!\!\!\perp \in S(B)$ **then**
13:         AddSubsumer($A$, $\perp\!\!\!\perp$)
14:     **else**
15:         **if** $(A, B) \notin R(r)$ **then**
16:             Process-new-edge($A$, $r$, $B$)
17:         **end if**
18:     **end if**
19: **end if**

---

The algorithm needs some explanations:

- In step-3, Algorithm A-3 is called to add $B$ as a subsumer of $A$.
- Step-5 to Step-9 realise **R8**.
- Step-12 to Step-13 realise **R9**.
- In step-16, Algorithm *Process-new-edge* in [1] is called.

During the execution of Algorithm A-2, Algorithm A-3 is called whenever a new subsumer is found. Its input are the subsumee and subsumer, respectively.

---

**Algorithm A-3**: AddSubsumer($A$,$B$)

$S(A) = S(A) \cup \{B\}$
queue($CT(B)$) =queue($CT(B)$) $\cup \{\rightarrow CT(A)\}$
$\hat{O}(CT(B))$ =$\hat{O}(CT(B)) \cup \{\rightarrow CT(A)\}$
**if** $\perp\!\!\!\perp \notin S(B)$ and $CT(B) \notin S(A)$ and $CT(A) \notin S(B)$ **then**
    queue($A$)=queue($A$)$\cup\hat{O}(B)$
    **for** all concept names $A'$ and role name r with $(A', A) \in R(r)$ **do**
        queue($A'$)=queue($A'$)$\cup\hat{O}(\exists r.B)$
    **end for**
**else**
    AddSubsumer($A$,$\perp\!\!\!\perp$)
    **for** all concept names $A'$ and role name r with $(A', A) \in R(r)$ **do**
        AddSubsumer($A'$,$\perp\!\!\!\perp$)

```
13:     end for
14: end if
```

Some explanations of the above algorithm:

- Step-2 and step-3 realise **R7** with incremental reasoning. The application of **R7** will always introduce a new axioms $CT(B) \sqsubseteq CT(A)$. Therefore we generate a new entry $\rightarrow CT(A)$ for $\hat{O}(CT(B))$ and initialise it into queue($CT(B)$) accordingly.
- Step-4 checks the condition of **R6**. Such a condition is important because once we realise that a concept is subsumed by $\perp\!\!\!\perp$, we immediately know that it's subsumed by any concept.
- Step-10 to Step-13 realise **R6** and **R9**

Post-processing should be done so that $\forall A \in CN_{AS}$ and $\perp\!\!\!\perp \in S(A)$, $S(A) := CN_{AS}$ and $R(r) := R(r) \cup \{(A, B)\}$ for all $r \in RN_{AS}$ and $B \in CN_{AS}$. Also, $A \in S(\top)$ should be subsumer of all the concept. The complexity of the reasoning is described by the following theorem:

**Theorem 2.** *Given an $\mathcal{EL}_C^+$ ontology $\mathcal{O} = (AS, CT)$ the computation of the S sets an R sets for all its named concepts and named roles will terminate in polynomial time w.r.t. $|CN_{AS} \cup RN_{AS}|$.*

*Proof.* The complexity of A-3 immediately follow the polynomial complexity of $\mathcal{EL}^+$ classification [1] and incremental reasoning [20].

### 3.3 Reasoning Service

Given the original ontology $O_1$ and its approximation $O_2 = OntoApprox(O_1)$, we can provide various reasoning services:

- The entailment checking of an arbitrary GCI is realised as: $O_1 \models C \sqsubseteq D$ if $O_2 \models CApprox(C) \sqsubseteq CApprox(D)$.
- The unsatisfiability of an concept expression $C$ can be realised by the entailment checking of $O_1 \models_? C \sqsubseteq \bot$, which will be reduced to entailment checking $O_2 \models_? CApprox(C) \sqsubseteq \perp\!\!\!\perp$.
- The inconsistency checking of $O_1$ can be realised by entailment checking $O_1 \models_? \top \sqsubseteq \bot$. Therefore, the problem of ontology consistency in $O_1$ can also be reduced to entailment checking $O_2 \models_? \top \sqsubseteq \perp\!\!\!\perp$ in $O_2$.
- Incremental reasoning with a temporal ontology $O_{temp}$: $O_1 \cup O_{temp}$ can be approximated into $OntoApprox(O_1 \cup O_{temp})$, which is equivalent to $O_2 \cup OntoApprox(O_{temp})$, whose taxonomy can be computed incrementally by adopting the incremental reasoning algorithm in [20].

The quality of the approximation and the reasoning is guaranteed by the following theorem:

**Theorem 3.** *Given an $\mathcal{R}$ ontology $O_1$, approximate it into $O_2$ with Algorithm A-1. For any $A, B \in CN_{O_1}$, $O_1 \models A \sqsubseteq B$ if $B \in S(A)$ can be inferred from $O_2$ by Algorithm A-2.*

Proof (sketch): It is easy to see the approximation by Algorithm A-1 is an equivalent syntactic transformation, because it is reversible. The completion rules implemented by Algorithm A-2 are obviously soundness-guaranteed.

**Incompleteness** Our extra completion rules process each axiom in $AS$ individually. In Algorithm A-2 we also process each queue entry individually. This helps keeping the reasoning tractable but some information that can only be derived from interaction of multiple entries will be lost:

*Example 3.* $\mathcal{T} = \{A \sqcap \neg B \sqsubseteq C, A \sqcap B \sqsubseteq C, D \sqsubseteq \exists r.\neg C, \exists r.B \sqsubseteq E, \exists r.\neg A \sqsubseteq E\}$

Obviously, we have $\mathcal{T} \models A \sqsubseteq C$ and thus $D \sqsubseteq E$. However, if we approximate it into $(\{X_1 \equiv A \sqcap nB, X_2 \equiv A \sqcap B, X_3 \equiv \exists r.nC, X_4 \equiv \exists r.nA, \ldots\}, \{(B, nB), \ldots\})$ and initialise Algorithm A-1, we will have queue$(A) = \{nB \rightarrow X_1, B \rightarrow X_2\}$. Obviously, $B$ and $nB$ are not subsumers of $A$ thus we can't further infer $C \in S(A)$.

This can be solved by resolution: $nB \rightarrow X_1 \in$ queue$(A)$ implies $A \sqsubseteq fc(nB) \sqcup X_1$ thus $A \sqsubseteq B \sqcup C$. similarly we have $A \sqsubseteq nB \sqcup C$. Together we can infer $A \sqsubseteq C$.

In order to further infer $D \sqsubseteq F$. A new axiom $\exists r.(B \sqcup \neg A) \equiv \exists r.B \sqcup \exists r.\neg A$ has to be added into $\mathcal{T}$ and approximated for incremental reasoning.

Although we can't guarantee completeness, we will see in next section that the recall is high, at least for our benchmark ontologies.

## 4 Evaluation

We implement our approximation and reasoning algorithm as a functionality of our REL reasoner for $\mathcal{EL}^+$, which is a component of our TrOWL Tractable Reasoning infrastructure of OWL[1].

To evaluate its performance, We compare REL with FaCT++ v1.2.3 and Pellet 2.0.0 rc5. The experiments are conducted in an environment of Microsoft Windows XP SP3 with 2.66 GHz CPU and 1G memory allocated to JVM 1.6.0.07.

In order to test its effects on difficult TBox with complex GCIs, we create our own set of 16 $\mathcal{R}$ ontologies [2]. In these ontologies, the size of any conjunction or disjunction is at most 5, the depth of a concept expression on the lhs or rhs of a GCI is at most 4. The domain and range of any role is a disjunction of depth at most 3. These ontologies are split into two sets, i.e. S1 and S2. In S1, ontologies have increasing number of concept names and GCIs. $|RN|$ is fixed to 10, number of simple RIs and complex RIs are about 10. In S2, ontologies have increasing $|RN|$. $|CN|$ is fixed to 20, number of GCIs is fixed to 30, number of RIs is increasing with the $|RN|$ . The performance metrics for REL include approximation time,

[1] `http://trowl.eu/`

[2] `http://www.abdn.ac.uk/~csc303/benchmark/RBenchmarkTest.zip`

reasoning time and completeness. All the time are measured in seconds. To evaluate the completeness, we check the subsumption between each pair of named concepts and count the number of discovered subsumptions.

Pellet timed-out or clashed in all the 16 ontologies. Results of REL and FaCT++ are illustrated in Table 2, in which recall is calculated as the ratio of

**Table 2.** Evaluation Results Against FaCT++

| $\mathcal{S}_1$ | $\|CN\|$ | $GCIs$ | $T_{approximation}$ | $T_{reasoning}$ | Recall | $T_{FaCT++}$ |
|---|---|---|---|---|---|---|
| R1 | 10 | 40 | 0.016 | 0.047 | 68.9% | 0.797 |
| R2 | 10 | 40 | 0.016 | 0.031 | 100% | 0.015 |
| R3 | 10 | 40 | 0.031 | 0.094 | N/A | - |
| R4 | 20 | 40 | 0.015 | 0.032 | 100% | 0.578 |
| R5 | 20 | 40 | 0.016 | 0.047 | N/A | - |
| R6 | 20 | 40 | 0.016 | 0.031 | 100% | 3.25 |
| R7 | 20 | 40 | 0.031 | 0.031 | 100% | 0.437 |
| R8 | 20 | 50 | 0.032 | 0.078 | N/A | - |
| R9 | 20 | 50 | 0.031 | 0.109 | N/A | - |
| R10 | 20 | 50 | 0.031 | 0.032 | 100% | 0.61 |
| $\mathcal{S}_2$ | $\|RN\|$ | | $T_{approximation}$ | $T_{reasoning}$ | Recall | $T_{FaCT++}$ |
| R11 | 100 | | 0.094 | 1.047 | 86.7% | 1.109 |
| R12 | 100 | | 0.094 | 0.875 | 98.1% | 2.907 |
| R13 | 100 | | 0.078 | 0.938 | 87.3% | 4.3 |
| R14 | 300 | | 0.234 | 7.281 | N/A | - |
| R15 | 300 | | 0.319 | 10.563 | 100% | 2.125 |
| R16 | 300 | | 0.234 | 5.969 | N/A | - |

subsumptions discovered by REL against those discovered by FaCT++. – means FaCT++ timed-out or run out of memory.

As we can see, some ontologies are extremely difficult in contrast to their small size. By looking in depth into the ontologies, we see the differences as:

- R2 is quite easy for both REL and FaCT++ because it contains many explicit equivalence between concept names.
- R3 has relatively complex domain and range with disjunctions, which are difficult to optimise for tableau algorithms.
- R4 and R7 are relatively easy because they are actually shallow ontologies with only a few implicit subsumptions.
- R1, R5, R6, R8 and R9 are difficult for FaCT++ because they all have many explicit or implicit $A \sqsubseteq \exists r.\top$ axioms, which will lead to large expansions for tableau algorithm.
- R11, R12 and R13 are actually shallow with a few implicit subsumptions, although the number of roles is not small.

In order to evaluate the usability of our approximative reasoning approach on real world ontologies, we test REL on the wine and cyc ontology. More precisely, we first remove all the ABox axioms from these ontologies, and then use Pellet, FaCT++ and REL to classify them. The results are given in Tab.3. In order to justify the complement-integrated reasoning algorithm, we also use a $\mathcal{EL}^+$ reasoner to classify the approximated $\mathcal{EL}^+$ TBox solely without taking complement table into account.

**Table 3.** Evaluation on Real World Ontology

| Ontology | DL | FaCT++ | Pellet | REL | Recall | $\mathcal{EL}^+$ Recall |
|---|---|---|---|---|---|---|
| OWL Guide Wine | $\mathcal{SHOIN}$ | 0.344 | 1.234* | 0.094 | 96.8% | 95.8% |
| Cyc | $\mathcal{ALCHF}$ | 0.422 | 98.765 | 4.109 | 100% | 1.2% |

Comparison shows that, REL can perform efficiently on real world ontologies as well and the recall is rather high (more than 90%). Also the complement-integrated reasoning algorithm can significantly improve the recall on particular ontologies.

## 5 Conclusion & Future Work

In this paper we presented our approach to approximating description logic $\mathcal{R}$ into $\mathcal{EL}^+$ with an additional complement table. The approximation method we proposed is a syntax-based transformation which is very efficient and preserving the complementary relations. With additional completion rules and abstract algorithms that we proposed, the reasoning of resulted ontology is sound and tractable. Preliminary evaluation results showed that our approach can outperform existing DL reasoners on some difficult $\mathcal{R}$ ontologies.

The follow-up of the current work will be the investigation of the completeness and the optimisation of the implementation. Potential extension of this method aims at including more expressiveness for both source and target languages: a more comprehensive approximation from OWL2-DL to OWL2-EL is under investigation. We expect our work to push one step forward to dealing with complex GCIs and RIs in very expressive DLs, and to build a bridge between the $\mathcal{ALC}$ and $\mathcal{EL}$ families of DLs.

## Acknowledgements

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

## References

1. Franz Baader, Carsten Lutz, and Boontawee Suntisrivaraporn. Is tractable reasoning in extensions of the description logic el useful in practice? In *Proceedings of the 2005 International Workshop on Methods for Modalities (M4M-05)*, 2005.

2. Perry Groot, Heiner Stuckenschmidt, and Holger Wache. Approximating description logic classification for semantic web reasoning. In Asunción Gómez-Pérez and Jérôme Euzenat, editors, *ESWC*, volume 3532 of *Lecture Notes in Computer Science*, pages 318–332. Springer, 2005.
3. Volker Haarslev and Ralf Möller. RACER system description. In *IJCAR '01: Proceedings of the First International Joint Conference on Automated Reasoning*, pages 701–706, London, UK, 2001. Springer-Verlag.
4. Pascal Hitzler and Denny Vrandecic. Resolution-based approximate reasoning for owl dl. In Y. Gil et al., editor, *Proceedings of the 4th International Semantic Web Conference, Galway, Ireland, November 2005*, volume 3729 of *Lecture Notes in Computer Science*, pages 383–397. Springer, Berlin, NOV 2005.
5. Ian Horrocks. Reasoning with expressive description logics: Theory and practice. In *In: Andrei Voronkov, (ed) Proc. 18th Int. Conf. on Automated Deduction (CADE-18)*, pages 1–15. Springer, 2002.
6. Ian Horrocks, Oliver Kutz, and Ulrike Sattler. The even more irresistible sroiq. In Patrick Doherty, John Mylopoulos, and Christopher A. Welty, editors, *KR 2006*, pages 57–67. AAAI Press, 2006.
7. Ian Horrocks and Ulrike Sattler. Decidability of shiq with complex role inclusion axioms. *Artif. Intell.*, 160(1):79–104, 2004.
8. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for expressive description logics. In *LPAR '99: Proceedings of the 6th International Conference on Logic Programming and Automated Reasoning*, pages 161–180, London, UK, 1999. Springer-Verlag.
9. Ian Horrocks, Ulrike Sattler, and Stephan Tobies. Practical reasoning for very expressive description logics. *Logic Journal of the IGPL*, 8:2000, 2000.
10. Ullrich Hustadt, Boris Motik, and Ulrike Sattler. Reducing shiq-description logic to disjunctive datalog programs. In *KR 2004*, pages 152–162, 2004.
11. Horrocks I. and Tobies S. Optimisation of terminological reasoning. Technical report, 2000.
12. Yevgeny Kazakov. SRIQ and SROIQ are harder than SHOIQ. In *DL 2008*, 2008. to appear.
13. B. Motik. Practical dl reasoning over large aboxes with kaon2.
14. Jeff Z. Pan and Edward Thomas. Approximating OWL-DL Ontologies. In *the Proc. of the 22nd National Conference on Artificial Intelligence (AAAI-07)*, pages 1434–1439, 2007.
15. Marco Schaerf and Marco Cadoli. Tractable reasoning via approximation. *Artificial Intelligence*, 74:249–310, 1995.
16. Bart Selman and Henry Kautz. Knowledge compilation and theory approximation. *J. ACM*, 43(2):193–224, 1996.
17. Rob Shearer, Boris Motik, and Ian Horrocks. Hermit: A highly-efficient owl reasoner. In Catherine Dolbear, Alan Ruttenberg, and Ulrike Sattler, editors, *OWLED 2008*, volume 432 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2008.
18. E Sirin, B Parsia, BC Grau, A Kalyanpur, and Y Katz. Pellet: A practical OWL-DL reasoner. *Journal of Web Semantics*, 5(2):51–53, June 2007.
19. Heiner Stuckenschmidt and Frank van Harmelen. Approximating terminological queries. In *FQAS '02: Proceedings of the 5th International Conference on Flexible Query Answering Systems*, pages 329–343, London, UK, 2002. Springer-Verlag.
20. Boontawee Suntisrivaraporn. Module extraction and incremental classification: A pragmatic approach for $\mathcal{EL}^+$ ontologies. In Sean Bechhofer, Manfred Hauswirth, Joerg Hoffmann, and Manolis Koubarakis, editors, *Proceedings of the 5th European*

*Semantic Web Conference (ESWC'08)*, volume 5021 of *Lecture Notes in Computer Science*, pages 230–244. Springer-Verlag, 2008.
21. D. Tsarkov and I. Horrocks. Fact++ description logic reasoner: System description. In *Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2006)*, volume 4130 of *Lecture Notes in Artificial Intelligence*, pages 292–297. Springer, 2006.
22. Dmitry Tsarkov, Ian Horrocks, and Peter F. Patel-Schneider. Optimizing terminological reasoning for expressive description logics. *J. Autom. Reason.*, 39(3):277–316, 2007.
23. Holger Wache, Perry Groot, and Heiner Stuckenschmidt. Scalable instance retrieval for the semantic web by approximation. In Mike Dean, Yuanbo Guo, Woochun Jun, Roland Kaschek, Shonali Krishnaswamy, Zhengxiang Pan, and Quan Z. Sheng, editors, *WISE Workshops*, volume 3807 of *Lecture Notes in Computer Science*, pages 245–254. Springer, 2005.