Towards Scalable Instance Retrieval over Ontologies

Alissa Kaplunova, Ralf Möller, Sebastian Wandelt, Michael Wessel

Hamburg University of Technology, 21079 Hamburg, Germany,
http://www.sts.tu-harburg.de

Abstract. In this paper, we consider the problem of query answering over large multimedia ontologies. Traditional reasoning systems may have problems to deal with large amounts of expressive ontological data (terminological as well as assertional data) that usually must be kept in main memory. We propose to overcome this problem with a new so-called *filter and refine paradigm for ontology-based query answering.*
The contribution of this paper is twofold: (1) For both steps, algorithms are presented. (2) We evaluate our approach on real world multimedia ontologies from the BOEMIE project[1].

1 Introduction

Applying semantic web technologies to enable the semantic retrieval of documents is a hot research topic. We believe that rather expressive DLs such as $\mathcal{SHI}$ are required in order to capture important domain constraints in an ontology (e.g., less expressive DLs may not provide the required expressivity for the modeling problems at hand). Thus, in this paper we focus on the DL $\mathcal{SHI}$ (extensions to larger OWL fragments will be considered in future research).

In general, ontologies tend to be large, both in numbers of concepts as well as in numbers of individuals. Unfortunately, the data complexity of instance retrieval in $\mathcal{SHI}$ (and more expressive DLs) is EXPTIME-complete. Thus, from a computational perspective, instance retrieval with large ontologies (containing lots of instances) may be very hard. Although mature DL/ OWL reasoning systems such as RacerPro exist [1], many reasoning systems for expressive DLs nowadays still work on main memory only. This obviously prevents their usage on very large ontologies, which may contain millions of "facts", so query answering simply runs out of main memory, or even loading of the whole ontology is already impossible.

Recently, query answering in less expressive DLs received great attention. E.g., the QuOnto system [2] is able to perform query answering on secondary memory by taking advantage of (relational) database technology.

In this paper, we propose a pragmatic method to combine the *high performance and data scalability* achieved by the techniques realized in the QuOnto architecture with the *high expressivity and expressivity scalability* realized by state-of-the-art DL reasoners such as RacerPro. We propose a new *filter & refine strategy* for expressive ontologies in order to address the *data- and expressivity scalability problem* [3].

[1] http://www.boemie.org/

This paper is structured as follows. First, the basics of descriptions logics (as far as relevant for this paper) are introduced; i.e., the DLs $\mathcal{SHI}$ and *DL-Lite*, as well as basic inference problems. Then, we describe the novel approximation algorithm which reformulates $\mathcal{SHI}$ ontologies as *DL-Lite* ontologies for the filter step. We then apply a novel partitioning algorithm for the refine step and perform a preliminary evaluation of our framework applied to the AEO ontology. Open problems are discussed and provide motivation for future research. Finally comes the conclusion and some discussion of related and future work.

This paper is accompanied by a technical report[4] containing full proofs and additional details.

2 Basics and Guiding Example

In the following part we will define mathematical notions, which are relevant for the remaining paper.

The Description Logic $\mathcal{SHI}$ We asume the syntax and semantics of the description logic $\mathcal{SHI}$ (also called $\mathcal{ALCHI}_{\mathcal{R}^+}$) and *DL-Lite*$_{\mathcal{F}}$ as defined in [5] and [6].

With $Ind(\mathcal{A})$ we denote the set of individuals occurring in $\mathcal{A}$. We say that $\mathcal{O}$ is *inconsistent*, denoted with $INC(\mathcal{O})$, if there exists no model for $\mathcal{O}$. We say that $\mathcal{O}$ is *consistent*, denoted with $CON(\mathcal{O})$, if there exists at least one model for $\mathcal{O}$. Given an individual a and an atomic concept C, we have $\langle \mathcal{T}, \mathcal{A} \rangle \models a : C$ iff $INC(\langle \mathcal{T}, \mathcal{A} \cup \{a : \neg C\} \rangle)$.

By *instance retrieval for concept* C, we obtain all individuals $a \in Ind(\mathcal{A})$, s.t. we have $\langle \mathcal{T}, \mathcal{A} \rangle \models a : C$. We denote the set of instances for a given concept C with $\mathsf{concept\_instances}(C, \mathcal{A}, \mathcal{T})$.

In the following we define some additional notions, which will be used throughout the remaining part of the paper. A $\exists$-*constraint* is a concept description of the shape $\exists R.C$, s.t. C is an arbitrary concept description. A $\forall$-*constraint* is a concept description of the shape $\forall R.C$, s.t. C is an arbitrary concept description.

The *subsumption hierarchy* (so-called *taxonomy*) of parents and children for each concept name can be obtained by classification. For $\mathcal{SHI}$ ontologies it is possible to compute the subsumption hierarchy in advance given only the TBox $\mathcal{T}$, i.e. without the ABox $\mathcal{A}$. This is possible since $\mathcal{SHI}$ does not allow the use of nominals. With $\sqsubseteq_{\mathcal{T}}$: $N_C \times N_C$ we denote the precomputed taxonomy obtained by classification, e.g., we have $\sqsubseteq_{\mathcal{T}} (C, D)$ iff $\mathcal{O} \models C \sqsubseteq D$ for atomic concepts C and D. The role hierarchy of a $\mathcal{SHI}$-ontology can be computed in advance given the TBox $\mathcal{T}$ only as well. With $\sqsubseteq_{\mathcal{R}}$: $N_R \times N_R$ we denote the precomputed role hierarchy, e.g. we have $(R, S) \in \sqsubseteq_{\mathcal{R}}$ iff $\mathcal{O} \models R \sqsubseteq S$ for roles R and S.

An atomic concept D is a synonym for a concept description C if we have $\mathcal{T} \models C \sqsubseteq D$ and $\mathcal{T} \models D \sqsubseteq C$. With $\mathsf{synonyms}(C, \mathcal{T})$ we denote the set of atomic concepts, which are synonyms for concept C with respect to $\mathcal{T}$. With $\mathsf{parents}(C, \mathcal{T})$ ($\mathsf{children}(C, \mathcal{T})$) we denote the set of atomic concepts which are more general (specific) than a given concept C.

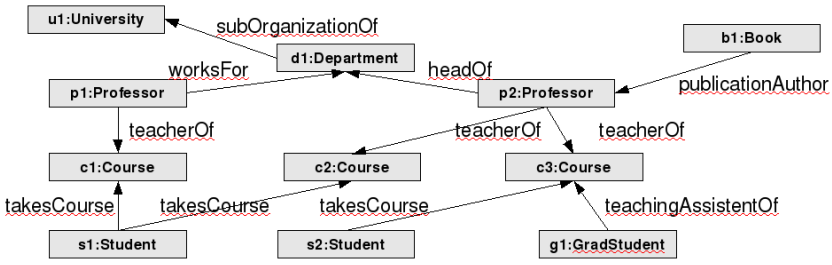

Fig. 1. Guiding Example: ABox $\mathcal{A}_{EX}$ for ontology $\mathcal{O}_{EX}$

Example 1. In the following we define an example ontology, which is used throughout the remaining part of the paper. The ontology is a simplified version of the LUBM [7]. Let $\mathcal{O}_{EX} = \langle \mathcal{T}_{EX}, \mathcal{A}_{EX} \rangle$, s.t.

$$\begin{aligned}\mathcal{T}_{EX} = \{\, & Chair \dot{\equiv} \exists headOf.Department \sqcap Person, \\ & Professor \dot{\sqsubseteq} Faculty, Book \dot{\sqsubseteq} Publication, \\ & GraduateStudent \dot{\sqsubseteq} Student, Student \dot{\equiv} Person \sqcap \exists takesCourse.Course, \\ & \top \dot{\sqsubseteq} \forall teacherOf.Course, \exists teacherOf.\top \dot{\sqsubseteq} Faculty, Faculty \dot{\sqsubseteq} Person, \\ & \top \dot{\sqsubseteq} \forall publicationAuthor^{-}.(Book \sqcup ConferencePaper), \\ & headOf \dot{\sqsubseteq} worksFor, worksFor \dot{\sqsubseteq} memberOf, memberOf \dot{\equiv} member^{-} \,\}\end{aligned}$$

The ABox $\mathcal{A}_{EX}$ is shown in Fig. 1. Please note that individual $p2$ is an non-obvious instance (i.e. we need to perform reasoning) of concept $Chair$, since $p2$ has an outgoing $headOf$-edge to a $Department$ and every $Professor$ is necessarily a $Person$.

3 Terminological Approximation - The Filter Step

Definition of Approximation Let us start with some basic definition. First we define the notion of an approximation of a TBox $\mathcal{T}$:

Definition 1. *For two TBoxes $\mathcal{T}_1$ and $\mathcal{T}_2$, $\mathcal{T}_2 \models \mathcal{T}_1$ iff all models of $\mathcal{T}_2$ are also models of $\mathcal{T}_1$.*

Definition 2. *Let $\mathcal{T}_1$ be a TBox in some DL $\mathcal{DL}$. A $\mathcal{T}_2$ is called an* approximation of $\mathcal{T}_1$ *iff a) $\mathcal{T}_2$ is a $\mathcal{DL}'$ TBox, with $\mathcal{DL}' \subseteq \mathcal{DL}$, and b) $\mathcal{T}_2 \models \mathcal{T}_1$ holds*[2]*.*

TBox entailment is decidable if $\mathcal{DL}$ is decidable, since $\mathcal{T}_2 \models \mathcal{T}_1$ iff for all $C \dot{\sqsubseteq} D \in \mathcal{T}_1$, $C \sqcap \neg D$ is unsatisfiable w.r.t. $\mathcal{T}_2$. Note that this is well-defined, since we assume $\mathcal{DL}' \subseteq \mathcal{DL}$.

After all, our intention for this definition is that instance retrieval over $\mathcal{A}$ w.r.t. $\mathcal{T}_2$ shall be complete, but possibly unsound compared with instance retrieval w.r.t. $\mathcal{A}$ and $\mathcal{T}_1$:

Proposition 1. *Let $\mathcal{A}$ be an ABox which contains only* atomic *concept assertions, i.e., for all $i : C \in \mathcal{A}$, C is an atomic concept: $C \in N_{CN}$. Let D be an atomic query concept, the concept whose instances shall be retrieved. Let $\mathcal{T}_2$ be an approximation of $\mathcal{T}_1$. Then, the following holds:* $\mathsf{concept\_instances}(D, \mathcal{A}, \mathcal{T}_1) \subseteq \mathsf{concept\_instances}(D, \mathcal{A}, \mathcal{T}_2)$.

[2] We are discussing the case where $\mathcal{DL} = \mathcal{SHI}$, and $\mathcal{DL}' = DL\text{-}Lite$.

How to Compute Approximations Having given these definitions, the question arises, how to actually compute an approximation of $\mathcal{T}$. The idea of the *approximation algorithm* is quite simple. W.l.o.g. we assume that a TBox $\mathcal{T}$ contains only implication axioms (axioms of the form $C \dot{\sqsubseteq} D$; an axiom $C \dot{\equiv} D$ is transformed into two axioms $C \dot{\sqsubseteq} D$, $D \dot{\sqsubseteq} C$). Please note that $\mathcal{SHI}$ admits role inclusion axioms (for roles R, S) $R \dot{\sqsubseteq} S$, which are valid in *DL-Lite* as well. Regarding transitive roles, which are not allowed in *DL-Lite*, the following well-known "trick" from the modal logic realm can be applied:

Definition 3. *Let R be a transitive role in $\mathcal{T}$*[3]*. Let $\mathsf{N_{CN}}(\mathcal{T})$ denote the set of concept names appearing in $\mathcal{T}$, and $\mathsf{N_T}(\mathcal{T})$ the set of transitively closed roles in $\mathcal{T}$. The K_4* closure *of $\mathcal{T}$, $\mathcal{T}^{K_4}$, is defined as follows:*

$$\mathcal{T}^{K_4} =_{def} \mathcal{T} \cup \{ \exists R.\exists R.C \dot{\sqsubseteq} \exists R.C, \quad \forall R.C \dot{\sqsubseteq} \forall R.\forall R.C \mid C \in \mathsf{N_{CN}}(\mathcal{T}), R \in \mathsf{N_T}(\mathcal{T}) \}.$$

Moreover, we assume that R is an ordinary role in $\mathcal{T}^{K_4} =_{def}$ (not a transitively closed one).

Proposition 2. *Let D be an atomic query concept, and $\mathcal{A}$ an ABox in which all concept assertions refer to atomic concepts only. Then,* $\mathsf{concept\_instances}(D, \mathcal{A}, \mathcal{T}) = \mathsf{concept\_instances}(D, \mathcal{A}, \mathcal{T}^{K_4})$.

We assume a corresponding function get_K4_closure which computes the $\mathcal{T}^{K_4}$ for a given $\mathcal{T}$. Please note that this proposition does not hold for arbitrary ABoxes and query concepts D (only for ABox containing atomic concept assertions, and atomic instance retrieval concepts).

Another preprocessing step is applied to remove nested occurrences of (sub) concepts of the form $\exists R.C$ and $\forall R.C$ from the axioms, so they can be better approximated to *DL-Lite* axioms. Thus, for each axiom $C \dot{\sqsubseteq} D$, and for each subconcept E in $\neg C \sqcup D$ with $E = \exists R.F$ or $E = \forall R.F$, and $F \notin N_{CN}$, we replace E with a new atomic concept C_E and add $\{C_E \dot{\sqsubseteq} E, E \dot{\sqsubseteq} C_E\}$ to $\mathcal{T}$. This process continues, until $\mathcal{T}$ no longer contains such axioms (note that E itself might still contain such subconcepts as well). For example, $\{C \dot{\sqsubseteq} D \sqcap \exists R.(E \sqcap F)\}$ is rewritten into $\{C \dot{\sqsubseteq} D \sqcap C_{\exists R.(E \sqcap F)}, C_{\exists R.(E \sqcap F)} \dot{\sqsubseteq} E \sqcap F, E \sqcap F \dot{\sqsubseteq} C_{\exists R.(E \sqcap F)}\}$. Consequently, we assume a function flatten_tbox which applies this transformation to a TBox $\mathcal{T}$. Each model of $\mathsf{flatten\_tbox}(\mathcal{T})$ is trivially also a model of $\mathcal{T}$, and vice versa, each model of $\mathcal{T}$ can uniquely be extended to a model of $\mathsf{flatten\_tbox}(\mathcal{T})$ (only the new atomic concepts must be interpreted correctly so that their axioms become satisfied).

For an $\mathcal{SHI}$ TBox $\mathcal{T}$ we we can now compute an approximated $\mathcal{T}'$ by approximating each axiom. So, $C \dot{\sqsubseteq} D \in \mathcal{T}$ is replaced by a logically stronger axiom $C' \dot{\sqsubseteq} D'$, $\{C' \dot{\sqsubseteq} D'\} \models \{C \dot{\sqsubseteq} D\}$, which is a *DL-Lite* axiom. The algorithm is best understood as a non-deterministic algorithm which works as follows (the actual deterministic implementation is described briefly below):

Function $\mathsf{approximate}(\mathcal{T})$
Parameter: $\mathcal{SHI}$ TBox $\mathcal{T}$
$\mathcal{T} := \mathsf{flatten\_tbox}(\mathsf{get\_K4\_closure}(\mathcal{T}))$
$\mathcal{T}' := \{ C \dot{\sqsubseteq} D \mid \mathcal{T} \models C \dot{\sqsubseteq} D, C, D \in N_{CN} \}$

[3] *DL-Lite* does not offer transitive roles.

```
    while T ≠ ∅
        axiom := select_axiom(T)
        T' := T' ∪ approximate_axiom(axiom, T')
        T := T \ {axiom}
    end while
return T'
```

The algorithm first syntactically transforms the input TBox $\mathcal{T}$ as explained. Although flatten_tbox introduces new atomic concepts, no additional "K4" axioms need to be introduced for them by get_K4_closure. Then, the taxonomy of $\mathcal{T}$ is made explicit by adding corresponding axioms to $\mathcal{T}'$; these axioms are *DL-Lite* axioms. The reason for this addition to $\mathcal{T}'$ is that the taxonomy of $\mathcal{T}$ shall be available for approximate_axiom (see below). Both select_axiom and approximate_axiom are non-deterministic as well. Given an axiom $C \dot{\sqsubseteq} D$, the basic idea of approximate_axiom is to *generalize* the left-hand side C to C', and to *specialize* the right-hand side D to D'. This ensures that the approximated axiom is stronger than the original axiom, since $C' \dot{\sqsubseteq} D' \models C \dot{\sqsubseteq} D$ iff $\neg C' \sqcup D' \models \neg C \sqcup D$ iff $(\neg C' \sqcup D') \sqcap \neg(\neg C \sqcup D)$ is unsatisfiable iff $(\neg C' \sqcup D') \sqcap C \sqcap \neg D$ is unsatisfiable iff both $\neg C' \sqcap C \sqcap \neg D$ and $D' \sqcap C \sqcap \neg D$ are unsatisfiable. Then, either $C \dot{\sqsubseteq} D$ (so this is a tautology, and thus the trivial case), or $C \dot{\sqsubseteq} C'$ (then $C \sqcap \neg C'$ is unsatisfiable) and $D' \dot{\sqsubseteq} D$, (so $D' \sqcap \neg D$ is unsatisfiable). In principle, it is of course also sufficient to find equivalent C', D' in *DL-Lite*. The concepts C' and D' are called *possible rewritings* of C resp. D, and $C' \dot{\sqsubseteq} D'$ is called a *possible rewriting* of $C \dot{\sqsubseteq} D$ in the following, or also a *candidate rewriting*.

For example, the axiom $C \dot{\sqsubseteq} D \sqcup E$ can be rewritten to $C \dot{\sqsubseteq} D$, or to $C \dot{\sqsubseteq} E$ (assuming that $C, D, T \in N_{CN}$). Moreover, $C \dot{\sqsubseteq} D \sqcup E$ can also be written as $\neg D \dot{\sqsubseteq} \neg C \sqcup E$, $\neg E \dot{\sqsubseteq} \neg C \sqcup D$, or even $\neg D \sqcap C \dot{\sqsubseteq} E$, and so on, yielding additional rewriting possibilities. Thus, re-arranging the left-hand sides of the axioms maximizes the number of rewriting possibilities. Even though these axioms are still equivalent to the original one, after rewriting into *DL-Lite* they no longer are. Perhaps for some reordering, no better approximations than $\top \dot{\sqsubseteq} \bot$ can be found. It is thus even more important to maximize the number of possible approximations in order to avoid bad approximations which are *too strong* (rendering the whole TBox unsatisfiable).

The approximate_axiom function considers the input axiom $C \dot{\sqsubseteq} D$ as a disjunction $\neg C \sqcup D$ which, in a first step, is brought into *disjunctive normal form (DNF)*. A concept is in DNF if it is in *negation normal form (NNF)*, and does not contain any (sub)concepts of the form $D \sqcap (E \sqcup F)$. Using simple boolean algebra, each concept can be brought into DNF. Note that the concepts are even simpler at this step in the processing chain, because complex qualification concepts have been removed in advance. In the following, we use the set notation for disjuncts of a concept in DNF: $\mathsf{DNF}(C \sqcap (E \sqcup F)) = (C \sqcap E) \sqcup (C \sqcap F) = \{C \sqcap E, C \sqcap F\}$. The function approximate_axiom non-deterministically chooses a subset of $\mathsf{DNF}(\neg C \sqcup D)$ as a possible left-hand side of the axiom, and uses the remaining disjuncts as right-hand side. Then, approx_axiom calls the non-deterministic functions generalize and specialize:

Function approximate_axiom($axiom, \mathcal{T}'$)

Parameter: $\mathcal{SHI}$ axiom $axiom = C \dot{\sqsubseteq} D$ and partial approximation $\mathcal{T}'$
if $\mathcal{T}' \models axiom$ then return $\mathcal{T}'$
else if $axiom$ is a *DL-Lite* axiom then return $\{axiom\} \cup \mathcal{T}'$
else
$concept := \mathsf{DNF}(\neg C \sqcup D)$
$left\_side := \mathsf{some\_subset\_of}(concept)$
$right\_side := concept \setminus left\_side$
$left\_side' := \mathsf{generalize}(\neg left\_side, \mathcal{T}')$
$right\_side' := \mathsf{specialize}(right\_side, \mathcal{T}')$
if $left\_side' \neq \emptyset$ and $right\_side' \neq \emptyset$ then
$axiom' := left\_side' \dot{\sqsubseteq} right\_side'$
if $\mathcal{T}' \not\models axiom'$ then return $\{axiom'\} \cup \mathcal{T}'$
return $\mathcal{T}'$

Both specialize and generalize first bring their argument concepts in DNF, and then specialize or generalize using a set of *non-deterministic rewriting rules* which are guided by the structure of the concept. The rules are applied exhaustively to the concept C until no more rule is applicable.

The rules make use of the helper function syns_or_parents which returns a non-empty result for *non-atomic concepts only:*

$$\mathsf{syns\_or\_parents}(C, \mathcal{T}') =_{def} \begin{cases} \mathsf{synonyms}(C, \mathcal{T}') & \text{if } \mathsf{synonyms}(C, \mathcal{T}') \neq \emptyset, C \notin N_{CN} \\ \mathsf{parents}(C, \mathcal{T}') & \text{if } \mathsf{synonyms}(C, \mathcal{T}') = \emptyset, C \notin N_{CN} \\ \emptyset & \text{otherwise} \end{cases}$$

Note that $\mathcal{T}'$ is only partially available, but already contains the taxonomy axioms derived from $\mathcal{T}$ (see approximate). Note that $C \in \mathsf{synonyms}(C, \mathcal{T}, \mathcal{T}')$ for all $C \in N_{CN}$.

The function generalize uses the following non-deterministic *generalization rules*; $C \rightarrow_G C'$ means that C is generalized to C':

- $C \rightarrow_G C'$, if C is a valid left-hand side of a *DL-Lite* axiom
- $\exists R.C \rightarrow_G C'$, $C' \in \{\exists R.\top\} \cup \mathsf{syns\_or\_parents}(\exists R.C, \mathcal{T}, \mathcal{T}')$ (note: $C \in N_{CN}$)
- $C \sqcap D \rightarrow_G C'$, $C' \in \{C, D\} \cup \mathsf{syns\_or\_parents}(C \sqcap D, \mathcal{T}, \mathcal{T}')$
- $C \sqcup D \rightarrow_G C'$, where $C' = C_1 \sqcup D_1$, with $C \rightarrow_G C_1$, $D \rightarrow_G D_1$, or $C' \in \mathsf{syns\_or\_parents}(C \sqcup D, \mathcal{T}, \mathcal{T}')$
- for all other concepts C: $C \rightarrow_G C'$, $C' \in \mathsf{syns\_or\_parents}(C, \mathcal{T}, \mathcal{T}')$

To give an example, consider generalize is applied to $\exists R.C \sqcup (E \sqcap F)$. First, the DNF is computed: $(\exists R.C \sqcap E) \sqcup (\exists R.C \sqcap F)$. Then, a possible rewriting is: $(\exists R.C \sqcap E) \sqcup (\exists R.C \sqcap F) \rightarrow_G \exists R.\top \sqcup F$, since $(\exists R.C \sqcap E) \rightarrow_G \exists R.\top$ and $(\exists R.C \sqcap F) \rightarrow_G F$. There are many other different rewritings.

Please note that *DL-Lite* does not permit negation or conjunctions on the left-hand sides of axioms; thus, it is impossible to generalize conjunctions by generalizing the arguments analog to the $\sqcup$-case. Note that, from this definition, in most cases $\forall R.C \rightarrow_G \top$ unless syns_or_parents finds some parent for $\forall R.C$ in $\mathcal{T}'$. In principle, it is also possible to generalize a disjunction $C \sqcup D$ to something like $C \sqcup D \sqcup E$, for some $E \in N_{CN}$ (although this will result in a huge search space in the implementation). The rules are designed in such a way to avoid *over-generalization* in order to keep the number of unsound query answers small. That

means, more specific rewriting alternatives shall be favored over less specific ones. For example, although $C \sqcap D \rightarrow_G C \sqcup D$ is conceivable, it doesn't make much sense under this premise, since both $C \sqcap D \rightarrow_G C$ as well as $C \sqcap D \rightarrow_G D$ are more specific.

The rules for concept specialization, specialize, exploit a similar function syns_or_children and follow the principle to avoid *over-specialization*, i.e., more general rewriting alternatives are preferred over more specific ones. In these rules, there is the possibility to rewrite a concept C to $\emptyset$. In case $C \rightarrow_S \emptyset$ for a conjunct C in $C \sqcap D$, then $\emptyset$ is considered as $\top$. However, in case C is a disjunct, then $\emptyset$ is considered as $\bot$. So, $\emptyset$ serves as the neutral element w.r.t. the surrounding operation:

- $C \rightarrow_S C'$, if C is a valid right-hand side for a *DL-Lite* axiom
- $\neg C \rightarrow_S \neg C'$, where $C \rightarrow_G C'$ (i.e., C is generalized),
  or $C \in \mathsf{syns\_or\_children}(\exists R.C, \mathcal{T}, \mathcal{T}')$.
- $\exists R.C \rightarrow_S C'$, $C' = \exists R_C.\top$ with $\mathcal{T}' := \mathcal{T}' \cup \{R_C \dot{\sqsubseteq} R, \exists R_C^-.\top \dot{\sqsubseteq} C\}$,
  or $C' = \exists R.\top$ with $\mathcal{T}' := \mathcal{T}' \cup \{\exists R^-.\top \dot{\sqsubseteq} C\}$,
  or $C \in \mathsf{syns\_or\_children}(\exists R.C, \mathcal{T}, \mathcal{T}')$ (note: $C \in N_{CN}$)
- $\forall R.C \rightarrow_S \emptyset$, $\mathcal{T}' := \mathcal{T}' \cup \{\exists R^-.\top \dot{\sqsubseteq} C'\}$, where $C \rightarrow_S C'$
- $C \sqcup D \rightarrow_S C'$, $C' \in \{C, D\} \cup \mathsf{syns\_or\_children}(C \sqcup D, \mathcal{T}, \mathcal{T}')$
- $C \sqcap D \rightarrow_G C'$, where $C' = C_1 \sqcap D_1$, with $C \rightarrow_S C_1$, $D \rightarrow_S D_1$,
  or $C' \in \mathsf{syns\_or\_children}(C \sqcap D, \mathcal{T}, \mathcal{T}')$
- for all other concepts C: $C \rightarrow_S C'$, $C' \in \mathsf{syns\_or\_children}(C, \mathcal{T}, \mathcal{T}')$

In principle, it is possible to use $C \sqcap D \rightarrow_S C \sqcap D \sqcap E$, for some $E \in N_{CN}$, but the same comments as given above (for $C \sqcup D$) apply. Please note that *DL-Lite* does not permit disjunctions on the right-hand sides of axioms. Moreover, specialize has a side-effect on $\mathcal{T}'$, since it may introduce additional axioms. For example, the $\exists R.C$-rule introduces a new range restriction on R by adding $\exists R^-.\top \dot{\sqsubseteq} C$ to $\mathcal{T}'$. So, $\exists R.C$ is in fact *generalized* to $\exists R.\top$; however, due to the introduced range restriction $\exists R^-.\top \dot{\sqsubseteq} C$ we get $\exists R.\top \models \exists R.C$. In combination this is a specialization of $\exists R.C$, as required. Another possibility would be to introduce a subrole R_C, $R_C \dot{\sqsubseteq} R$ with range C, and rewrite $\exists R.C$ to $\exists R_C.\top$, but this would require a modification of the ABox during instance retrieval.

The rule $\forall R.C \rightarrow_S \emptyset$ deserves an explanation. The idea here is to completely ignore this (sub)concept on the right-hand side, and instead put a new axiom into $\mathcal{T}'$ (which is modified per side-effect): $\mathcal{T}' := \mathcal{T}' \cup \{\exists R^-.\top \dot{\sqsubseteq} C'\}$. For example, consider the TBox $\{C \dot{\sqsubseteq} (\forall R.D) \sqcap E\}$. Since the left-hand side is already acceptable, only the right-hand side is rewritten: $(\forall R.D) \sqcap E \rightarrow_S \top \sqcap E$, since $\forall R.D \rightarrow_S \emptyset$ and $E \rightarrow_S E$. However, also $\exists R^-.\top \dot{\sqsubseteq} D$ has been added to $\mathcal{T}'$, thus the approximation is $\mathcal{T}' = \{C \dot{\sqsubseteq} E, \exists R^-.\top \dot{\sqsubseteq} D\}$. It is easy to see that $\mathcal{T}' \models \mathcal{T}$ holds. In case the input TBox is $\{C \dot{\sqsubseteq} (\forall R.D) \sqcup E\}$, then the following rewriting is possible: $(\forall R.D) \sqcup E \rightarrow_S \forall R.D \rightarrow_S \emptyset$. Since approximate_axiom will reject axioms with $right\_side' = \emptyset$, the approximation is simply $\mathcal{T}' = \{\exists R^-.\top \dot{\sqsubseteq} D\}$. Another possibility is of course $\mathcal{T}' = \{C \dot{\sqsubseteq} E\}$, according to the $\sqcup$-rule.

Proposition 3. *Let* $\mathcal{T}' = \mathsf{approximate}(\mathcal{T})$ *for a* $\mathcal{SHI}$ *TBox* $\mathcal{T}$*. Then,* $\mathcal{T}'$ *is a DL-Lite approximation of* $\mathcal{T}$*.*

An Example Approximation If approximate is applied to the example TBox $\mathcal{T}$, then after the preprocessing only the following non-*DL-Lite* axioms remain which thus have to be approximated:

$\{\ \top \dot{\sqsubseteq} \forall publicationAuthor^{-}.(book \sqcup conferencePaper), \top \dot{\sqsubseteq} \forall teacherOf.course,$
$person \sqcap \exists takesCourse.course \dot{\sqsubseteq} student, student \dot{\sqsubseteq} \exists takesCourse.course,$
$person \sqcap \exists headOf.department \dot{\sqsubseteq} chair, chair \dot{\sqsubseteq} \exists headOf.department\ \}$

One possible *DL-Lite* approximation $\mathcal{T}'$ is:

$\{\ chair \dot{\sqsubseteq} \exists headOf.\top, \exists headOf.\top \dot{\sqsubseteq} chair, chair \dot{\sqsubseteq} person,$
$professor \dot{\sqsubseteq} faculty, book \dot{\sqsubseteq} publication, faculty \dot{\sqsubseteq} person,$
$graduateStudent \dot{\sqsubseteq} student, student \dot{\sqsubseteq} \exists takesCourse.\top, student \dot{\sqsubseteq} person,$
$\exists teacherOf^{-}.\top \dot{\sqsubseteq} course, \exists takesCourse.\top \dot{\sqsubseteq} student, \exists teacherOf.\top \dot{\sqsubseteq} faculty,$
$\exists publicationAuthor.\top \dot{\sqsubseteq} book,$
$\exists headOf^{-}.\top \dot{\sqsubseteq} department, \exists takesCourse^{-}.\top \dot{\sqsubseteq} course\ \}$

If this $\mathcal{T}'$ is used for retrieval on the example ABox, then no unsound answers to atomic instance retrieval queries are delivered. Thus, the approximation is *perfect* for every atomic concept of the ABox. Of course, this depends on the ABox. Note that $p2$ is a *chair* instance, as required, since $\exists headOf.\top \dot{\sqsubseteq} chair \in \mathcal{T}'$.

However, there also exist imperfect approximations. On the one hand, there are many $\mathcal{T}'$'s containing incoherent concepts, or even unsatisfiable $\mathcal{T}'$'s. In the latter case, the ABox is definitely inconsistent (unsatisfiable), and in the former case, inconsistency of the ABox is likely (if the ABox contains instances of incoherent concepts). From a logical perspective, an instance retrieval query performed on an inconsistent ABox returns the set of *all* ABox individuals (since, from an inconsistent theory, everything follows). So, such a query answer is still complete.[4]

On the other hand, an example for an unsound approximation is given by $\mathcal{T}'$, if $\top \dot{\sqsubseteq} \forall publicationAuthor^{-}.(book \sqcup conferencePaper)$ is approximated to $\exists publicationAuthor.\top \dot{\sqsubseteq} conferencePaper$ instead of the concept chosen before, $\exists publicationAuthor.\top \dot{\sqsubseteq} book$. Then, $b1$ will be a false answer to the instance retrieval query for $conferencePaper$.

Even for the simple example TBox we get 757 coherent approximations (there are a few hundred thousand consistent approximations containing incoherent concepts); w.r.t. retrieval, there is one perfect approximation (see above), and the worst coherent approximation has an average failure of 2.5 individuals which means that an atomic instance retrieval query, in the average, returns 2.5 false query answers.

An Implementation of the Approximation Algorithm We have eliminated the non-determinism in the approximate algorithm by implementing it in a depth-first (backtracking) search algorithm. Thus, for a given *axiom*, approximate_axiom(*axiom*) returns a set of *candidate axioms*, representing possible approximations of *axiom*. Each axiom thus represents a state in the search space, whose branching factor is given by the number of its candidate approximation axioms.

[4] Of course, a DL reasoner typically does not permit ABox retrieval on inconsistent ABoxes.

In principle, the number of possible approximations is truly astronomic for larger TBoxes. Consider a TBox with 500 axioms to approximate, in which each axiom can be approximated in three different ways – the number of atoms in the universe is $10^{80} \approx 3^{167.6722}$ and thus tiny compared to the 3^{500} nodes in this search space. Thus, clever heuristics are needed to guide the search. Since, in principle, one is only interested in coherent approximations (containing only one incoherent concept name, $\bot$), it is a good idea to prune a path in the search tree as soon as more than one incoherent concept is discovered in the partial $\mathcal{T}'$. Of course, this requires a TBox coherence check by the DL reasoner (RacerPro) at each step. This would be a good use case for *incremental* reasoning. In order to filter out candidate axioms which are too specific, RacerPro is used as well.

It may not be possible to compute a coherent approximation at all. In this case, an incoherent TBox is wanted which at least leaves the ABox satisfiable (but even this may be impossible), or contains only a minimal number of incoherent concepts.

Sometimes it is possible to compute more than one approximation. Even if each computed approximation is unsound for retrieval on an actual ABox, it is good to have a multitude of approximations available, since their retrieval results can be intersected. Even if no – w.r.t. an actual ABox – prefect approximation is among the computed approximations, this intersected answer set may by perfect for some concept C on this ABox.

4 Island-Based Instance Retrieval – The Refine Step

In the following section we discuss how to post-filter individuals, which were obtained by the Filter Step before. The original algorithm was proposed in [8] for the DL $\mathcal{ALCHI}$. This section is only intended as a overview of the refine step. Detailed explanations and proofs are omited in this paper.

The idea for the refine step is that only a subset of role and concept assertions is necessary/used to perform instance checking for a particular given individual a and a given concept C. The approach undertaken here is to identify role assertions which can be used during the application of a tableau algorithm for instance checking (note that $\langle \mathcal{T}, \mathcal{A} \rangle \models^? a : C$ can be reduced to checking whether $\langle \mathcal{T}, \mathcal{A} \cup \{a : \neg C\} \rangle$ is unsatisfiable via a tableau algorithm).

First, we transform the ontolofy into some kind of normal form, called *shallow normal form*. For the details of the transformation please refer to [8]. We only provide an example for $\mathcal{T}_{EX}$ from Example 1 in Shallow Normal Form. The TBox $\mathcal{T}_{EX}$ in SNF is as follows: $Shallow(\mathcal{T}_{EX}) =$

$$
\begin{aligned}
\{\ & \neg Chair \sqcup \exists headOf.Department, \neg Chair \sqcup Person, \\
& \forall headOf.\neg Department \sqcup \neg Person \sqcup Chair, \\
& \neg Professor \sqcup Faculty, \neg Book \sqcup Publication, \\
& \neg GraduateStudent \sqcup Student, \neg Student \sqcup Person, \\
& \neg Student \sqcup \exists takesCourse.Course, \\
& \neg Person \sqcup \forall takesCourse.\neg Course \sqcup Student, \\
& \forall teacherOf.Course, \forall teacherOf.\bot \sqcup Faculty, \neg Faculty \sqcup Person, \\
& \forall publicationAuthor^{-}.(Book \sqcup ConferencePaper)\ \}
\end{aligned}
$$

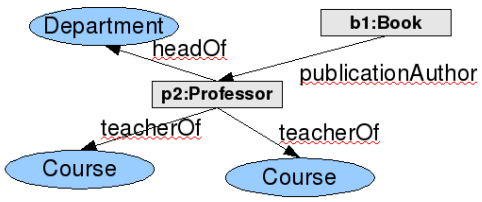

Fig. 2. Example island for individual $b1$ in $\mathcal{O}_{EX}$

Given the shallow normal form, we use a so-called $\forall$-info structure for an ontology $\mathcal{O}$ to determine which concepts are (worst-case) propagated over role assertions in an ABox. This helps us to define a notion of separability. The following definition of $\mathcal{O}$-separability is used to determine the importance of role assertions in a given ABox. Informally speaking, the idea is that $\mathcal{O}$-separable assertions will never be used to propagate "complex and new information" (see below) via role assertions.

Definition 4. *Given an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$, a role assertion $R(a, b)$ is called* $\mathcal{O}$-separable*, if we have $INC(\mathcal{O})$ iff $INC(\langle \mathcal{T}, \mathcal{A}_2 \} \rangle)$, where*

$$\mathcal{A}_2 = \mathcal{A} \setminus \{R(a,b)\} \cup \{R(a,i_1), R(i_2,b)\} \cup \{i_1 : C | b : C \in \mathcal{A}\} \cup \{i_2 : C | a : C \in \mathcal{A}\},$$

s.t. i_1 and i_2 are fresh individual names.

The extraction of islands for instance checking in ontology $\mathcal{O}$, given an individual a, is now straightforward. From an individual a one just follows each non-$\mathcal{O}$-separable role assertion in the original ABox, until at most $\mathcal{O}$-separable role assertions are left. For the details of this algorithm please refer to [8]. In Figure 2 we show the island computed for individual $b1$. Please recall that $b1$ potentially was a false answer to the $conferencePaper$ instance retrieval query. It is easy to see that the computed island does not entail $b1 : conferencePaper$ and thus $b1$ can be eliminated from the set of candidates. In Figure 2 it is also easy to see that $p2$ can be verified to be an instance of $Chair$, by only taking into account the corresponding island, since the $headOf$-edge was preserved during the computation of the island.

Extension from DL $\mathcal{ALCHI}$ to DL $\mathcal{SHI}$ Transitive roles can be easily read off from the TBox by additionally taking into account the role hierarchy. Then, whenever we want to compute the island for an individual w.r.t. DL $\mathcal{SHI}$, then we have to additionally "follow" all transitive role assertions. This proposal for the extension to DL $\mathcal{SHI}$ is quite straight-forward and we do not prove it here.

5 Preliminary Evaluation

We have performed an initial evaluation of our algorithms on a version of the AEO ontology of the BOEMIE project. Using RacerPro, we have transformed this OWL ontology into a DL ontology (= TBox, ABox). The utilized TBox DL is $\mathcal{ALCHf}$. It contains 1061 axioms which are already in *DL-Lite*, and 499 axioms which have to be approximated to *DL-Lite*. AEO also contains some

so-called number restrictions, which we simply approximate to functional roles in *DL-Lite*$_{\mathcal{F}}$ (since only $\leq_1 R$ concepts appear).

The ABox of the AEO version we used is rather small – it only contains 138 individuals (266 concept assertions plus 70 role assertions = 336 assertions). We have chosen this ABox since some interesting reasoning is required in order to retrieve the instances of the concept $HighJump$ (similar to the *chair* example, but over 2 role fillers).

As illustrated previously, it is very demanding to approximate a TBox with 499 axioms. Unfortunately, we were not successful to compute a coherent approximation of this AEO ontology in reasonable time. Better heuristics are needed here. The reason for this is a massive number of *disjointness axioms*; e.g., axioms of the form $A \dot{\sqsubseteq} \neg B$, $A \dot{\sqsubseteq} \neg C$, ... Additionally, get_K4_closure introduces another 1110 additional axioms.

We have thus simplified AEO substantially by removing all disjointness axioms and ignoring transitivity (so get_K4_closure adds no axioms). With this version, a coherent approximation could be computed within 5 minutes. These simplifications do not affect retrieval. In the average, it returns 0,984 false instances for a concept (w.r.t. to the original AEO).

The original AEO contains one instance of $HighJump$, and no instances of $SprintCompetition$. The approximated version is perfect for $HighJump$, but delivers 7 wrong $SprintCompetition$s. The $HighJump$ instance is in fact also a (false) $SprintCompetition$ here. Thus, 7 islands were computed by the partitioning method, ranging in size from 7 to 45 assertions. The average island contains 33.75 assertions. So, in the average, only one tenth of the assertions from the original ABox have to be loaded in order to verify or falsify the candidates for $HighJump$ and $SprintCompetition$. Each instance test requires $\approx$ 180 msecs per candidate, thus, after $\approx 1,440$ seconds the candidate individuals have been refined. Some additional time is needed to compute the islands. Computation of an islands needs milliseconds only (for such small islands).

Since this ABox was rather small, we expanded the ABox artificially by a factor of 500. We thus created an ABox which contained 500 copies of the original ABox, simply by prefixing the individuals with numbers (0 to 499). We then connected these 500 separated ABox parts using some new artificial role assertions, resulting in a connected ABox, containing 415330 assertions. The ABox still fits into main memory, because otherwise we could not have performed this experiment (the secondary memory-access is not yet realized). The ABox consistency check already needs 3,5 minutes on this ABox now; retrieval requires $\approx$ 34 seconds (for each concepts). As expected, from the approximated version of the TBox, we got 500 $HighJump$ instances, and 3500 $SprintCompetition$s. As expected, the newly introduced artificial role assertions connecting the 500 copies have no influence on the size of the islands. Thus, the average islands size is still 33.75; that this is only 0.0008126068 % of the whole ABox. However, now 4000 candidate tests have to be performed, which will require $\approx$ 14 minutes of RacerPro reasoning time. Additional time for accessing and loading from sec-

ondary memory etc. (since also the island partitioning has to work on secondary memory in the future) is taken[5].

6 Conclusions, Related and Future Work

Summing up, the evaluation in the previous section has shown that retrieval will require $\approx$ 20 minutes with our framework. This is not too bad compared with the $\approx$ 5 minutes for retrieval on the original AEO (note that the ABox consistency check needs to be performed only once). For a factor of ≈ 4, we have removed the main memory burden.So, this evaluation should be understood as a first preliminary proof of concept of the ideas conveyed in this paper.

Our framework rests on two central assumptions: (1) it must be possible to compute a coherent approximation of the original ontology in *DL-Lite* (or *DL-Lite*$_{\mathcal{F}}$), or another less expressive DL, which allows for secondary memory retrieval. As our preliminary evaluation with a real-world multimedia ontology has shown, this may be very hard. Several problems regarding the efficient handling of disjoint axioms and transitive roles still need to be solved, (2) the original ontology must allow for effective partitioning, so that the individual partition do not exceed main memory size. This may not be the case for all ontologies.

References

1. Haarslev, V., Möller, R., Wessel, M.: RacerPro User's Guide and Reference Manual Version 1.9.1 (May 2007)
2. Acciarri, A., Calvanese, D., Giacomo, G.D., Lembo, D., Lenzerini, M., Palmieri, M., Rosati, R.: QuOnto: Querying ontologies. In: Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005). (2005)
3. Möller, R., Haarslev, V., Wessel, M.: On the Scalability of Description Logic Instance Retrieval. In Freksa, C., Kohlhase, M., eds.: 29. Deutsche Jahrestagung für Künstliche Intelligenz. Lecture Notes in Artificial Intelligence, Springer Verlag (2006)
4. Kaplunova, A., Möller, R., Wandelt, S., Wessel, M.: Approximation and ABox Segmentation - Technical Report. Technical report, STS (2010) See http://www.sts.tu-harburg.de/tech-reports/2008/KMWW08.pdf.
5. Baader, F., Calvanese, D., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F.: The Description Logic Handbook. Cambridge University Press, New York, NY, USA (2007)
6. Calvanese, D., De Giacomo, G., Lembo, D., Lenzerini, M., Rosati, R.: Tractable Reasoning and Efficient Query Answering in Description Logics: The DL-Lite Family. J. of Automated Reasoning **39**(3) (2007) 385–429
7. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for knowledge base systems. J. Web Sem. **3**(2-3) (2005) 158–182
8. Wandelt, S., Moeller, R.: Island Reasoning for ALCHI Ontologies. In: Proceedings of the 5th International Conference on Formal Ontology in Information Systems (FOIS-04), IOS Press (2008)

[5] The test files and results can be downloaded from `http://www.sts.tu-harburg.de/~mi.wessel/download/boemie-experiment.zip`.