Towards Scalable Instance Retrieval over Ontologies

Sebastian Wandelt, Ralf Möller, 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 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.

In this paper, we propose a new *filter & refine strategy* for expressive ontologies in order to address the *data- and expressivity scalability problem* [2].

The idea is depicted in Figure 1. Initially (left of Figure 1), a set of individuals is present in an ontology - indicated as small circles. Given a retrieval query, we want to determine all the individuals, which are instances of a concept description. We propose to perform instance retrieval in two steps:

[1] This work has been funded by the German Science Foundation with the project PRESINT (DFG MO 801/1-1).

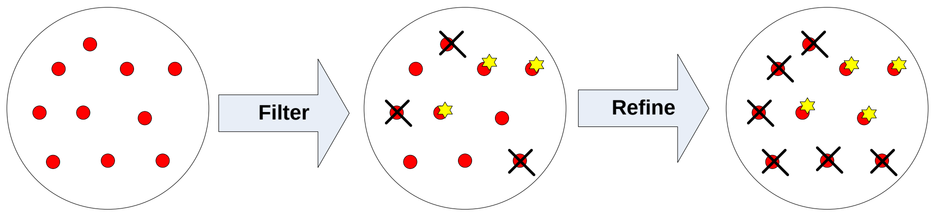

Fig. 1. General idea of Filter and Refine for Retrieval

1. *Filter Step*: The idea of the Filter step is to find some obvious solutions (indicated with a small star) and remove some obvious non-solutions (indicated as crossed out). In Figure 1, there are three obvious solutions to a retrieval query and three obvious non-solutions. For the remaining four individuals it is not yet clear, whether they are solutions or not.
2. *Refine Step*: For the remaining individual (four individuals in Figure 1), we perform an optimized instance check, which is based on using locality in the ABox, i.e. one does not have to load the whole ABox for the instance check.

Recently, query answering in less expressive DLs received great attention. E.g., the QuOnto system [3] is able to perform query answering on secondary memory by taking advantage of (relational) database technology. Based on efficient reasoning over less expressive description logics, we propose a filter step, which approximates a $\mathcal{SHI}$ ontology to a less expressive, but yet sound or complete, ontology. Then for instance QuOnto can be used to pre-filter individual candidates, i.e. obvious solutions and obvious non-solutions. In addition, we propose an alternative main-memory filtering technique based on individual similarity, which turns out to be more efficient in practice and seems like a promising low memory-footprint data structure for retrieval techniques. Our refine step is based on the work in [4] and exploits locality of ABox individual information. The idea is that a considerable amount of role assertions in an ontology can be broken up without changing the semantics for instance checking and instance retrieval.

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 and we propose a second filtering algorithm based on local similarity of individuals. We then apply a 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.

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 briefly recall syntax and semantics of the description logic $\mathcal{SHI}$ (also called $\mathcal{ALCHI}_{\mathcal{R}^+}$). For the details, please refer to [5]. We assume a collection of disjoint sets: a set of *concept names* N_{CN}, a set of *role names* N_{RN} and a set of *individual names* N_I. The *set of roles* N_R is $N_{RN} \cup \{R^- | R \in N_{RN}\}$, where R^- denotes an *inverse role*. A distinguished subset N_T of roles is called a *set of transitive roles*. The set of $\mathcal{SHI}$*-concept descriptions* is given by the following grammar:

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

where $A \in N_{CN}$ and $R \in N_R$. We say that a concept description is *atomic*, if it is a concept name or a negated concept name. With N_C we denote all atomic concepts. For defining the semantics of concept descriptions and roles we consider *interpretations* $\mathcal{I}$ that consist of a non-empty set $\Delta^{\mathcal{I}}$, the domain, and an interpretation function $\cdot^{\mathcal{I}}$, which assigns to every concept name A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every role R a set $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. For complex concept descriptions the interpretation function is extended as shown in [5]. The semantics of description logics is based on the notion of satisfiability. An interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ *satisfies* a concept description C if $C^{\mathcal{I}} \neq \emptyset$. In this case, $\mathcal{I}$ is called a *model* for C. A concept description is in *negation normal form* if negation occurs only in front of concept names.

A *TBox* is a finite set of axioms of the form $C \dot{\sqsubseteq} D$ (so-called *generalized concept inclusions*, GCIs). together with a finite set of axioms $R \dot{\sqsubseteq} S$ (*role inclusions*). An interpretation $\mathcal{I}$ *satisfies* a GCI $C \dot{\sqsubseteq} D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$. An interpretation $\mathcal{I}$ *satisfies* a role inclusion $R \dot{\sqsubseteq} S$ if $R^{\mathcal{I}} \subseteq S^{\mathcal{I}}$. An interpretation is a *model* of a TBox $\mathcal{T}$ if it satisfies all generalized concept inclusions in $\mathcal{T}$ and all role inclusions in $\mathcal{T}$. An *ABox* is a finite set of so-called *concept and role assertions* $C(a)$ and $R(a, b)$, where a and b are elements of N_I. An interpretation $\mathcal{I}$ *satisfies* a concept assertion $C(a)$ (resp. role assertion $R(a, b)$) if $a^{\mathcal{I}} \in C^{\mathcal{I}}$ (resp. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$).

A *ontology* $\mathcal{O}$ is a pair $\langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a TBox and $\mathcal{A}$ is a ABox. Interpretations are extended to ontologies in the usual way ([5]). We restrict the concept assertions in $\mathcal{A}$ in such a way that each concept description is an atomic concept or a negated atomic concept. This is a common assumption, when dealing with large assertional datasets in ontologies.

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 $\langle \mathcal{T}, \mathcal{A} \rangle \models C(a)$ 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 $\langle \mathcal{T}, \mathcal{A} \rangle \models C(a)$. 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 in 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 $\dot{\sqsubseteq}_{\mathcal{T}} : N_C \times N_C$ we denote the precomputed taxonomy obtained by classification, e.g., we have $\dot{\sqsubseteq}_{\mathcal{T}}(C, D)$ iff $\mathcal{O} \models C \dot{\sqsubseteq} D$ 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 $\dot{\sqsubseteq}_{\mathcal{R}} : N_R \times N_R$ we denote the precomputed role hierarchy, e.g. we have $(R, S) \in \dot{\sqsubseteq}_{\mathcal{R}}$ iff $\mathcal{O} \models R \dot{\sqsubseteq} S$ for roles R and S.

An atomic concept D is a synonym for a concept description C if we have $\mathcal{T} \models C \dot{\sqsubseteq} D$ and $\mathcal{T} \models D \dot{\sqsubseteq} C$. 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.

DL-Lite$_{\mathcal{F}}$ ([6]) is a fragment of OWL DL with the set of concept and role descriptions defined by the following grammar: $B \longrightarrow A \mid \exists R.\top, C \longrightarrow B \mid \neg B, R \longrightarrow P \mid P^{-}, E \longrightarrow R \mid \neg R$
where A is an *atomic concept*, P is an *atomic role*, and P^{-} is the *inverse* of the atomic role P. B denotes a *basic concept*, i.e., a concept that can be either an atomic concept or a concept of the form $\exists R.\top$, and $\top$ is the top concept, R denotes a *basic role*, i.e., a role that is either an atomic role or the inverse of an atomic role. *DL-Lite*$_{\mathcal{F}}$ TBox is a set of axioms of the form $B \dot{\sqsubseteq} C$ (concept inclusions), where only basic concepts may occur on the left-hand side, and a set of global role functionality assertions of the form $funct(R)$. An ABox includes as usually concept and role assertions $C(a)$ and $R(a, b)$.

In the following, we introduce an example ontology, which will be used throughout the remaining part. The example ontology $\mathcal{O}_{Ex1}$ in Example 1 is from university domain and inspired by the Lehigh University Benchmark, introduced in [7].

Example 1 (Running Example Ontology). The example ontology

$$\mathcal{O}_{Ex1} = \langle \mathcal{T}_{Ex1}, \mathcal{A}_{Ex1} \rangle$$

is defined as follows

$$
\begin{aligned}
\mathcal{T}_{Ex1} = \{ & \\
& Chair \equiv \exists headOf.Department, Student \equiv \exists takes.Course, \\
& GraduateStudent \dot{\sqsubseteq} \forall takes.GraduateCourse, \\
& UndergraduateCourse \sqcap Chair \dot{\sqsubseteq} \bot, GraduateCourse \sqcap Chair \dot{\sqsubseteq} \bot, \\
& Student \sqcap Chair \dot{\sqsubseteq} \bot, \top \dot{\sqsubseteq} \forall takes.Course, \\
& headOf \dot{\sqsubseteq} memberOf, teaches \equiv isTaughtBy^{-} \\
& \}
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{A}_{Ex1} = \{ & \\
& Department(cs), Department(ee), \\
& Professor(ann), Professor(eve), Professor(mae), \\
& UndergraduateCourse(c1), UndergraduateCourse(c4), \\
& UndergraduateCourse(c5), \\
& GraduateCourse(c3), GraduateCourse(c4), \\
& Student(ani), Student(ean), Student(eva), Student(noa), \\
& Student(sam), Student(sue), Student(zoe), \\
& headOf(ann, cs), memberOf(eve, cs), headOf(mae, ee), \\
& teaches(ann, c1), teaches(eve, c2), teaches(eve, c3), \\
& teaches(mae, c4), teaches(mae, c5), \\
& takes(ani, c1), takes(ean, c1), takes(ean, c2), takes(eva, c3), \\
& takes(noa, c3), takes(sam, c4), takes(sue, c5), takes(zoe, c5) \\
& \}
\end{aligned}
$$

The relationships among individuals of $\mathcal{A}_{Ex1}$ are depicted in Figure 2. Please note that only role assertions are used to build the graph, since we only want to emphasize the relationship between the ABox individuals.

3 Filter Step 1: Terminological Approximation

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 (TBox Entailment). *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 $\mathcal{T}_1$.*

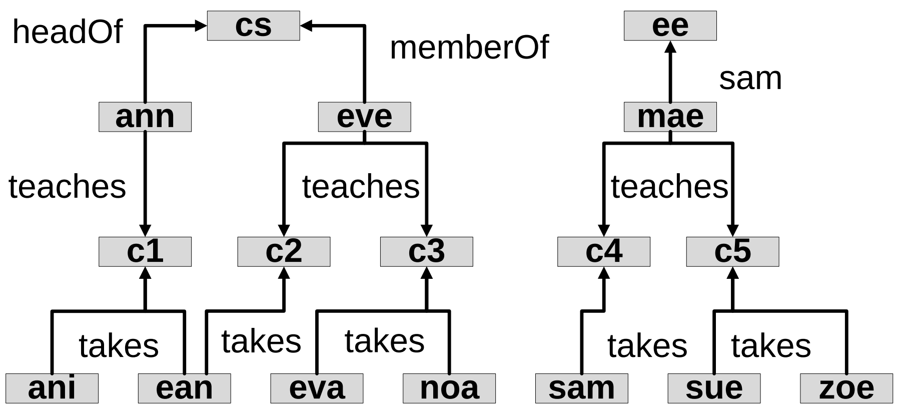

Fig. 2. Individual relationships for Example 1

Definition 2 (Approximation). *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:* $\mathsf{concept\_instances}(D, \mathcal{A}, \mathcal{T}_1) \subseteq \mathsf{concept\_instances}(D, \mathcal{A}, \mathcal{T}_2)$.

Proof. (Sketch) Assume that $i \in \mathsf{concept\_instances}(D, \mathcal{A}, \mathcal{T}_1)$, but $i \notin \mathsf{concept\_instances}(D, \mathcal{A}, \mathcal{T}_2)$. Let Σ_i, $i \in \{1, 2\}$ denote the logical theory of $\mathcal{T}_i$ and $\mathcal{A}$, where $\alpha \in \Sigma_i$ is either an ABox assertion or a TBox axiom in $\mathcal{DL}'$ (not $\mathcal{DL}$). Then, $\Sigma_1 \subseteq \Sigma_2$, since $\mathcal{T}_2$ is an approximation of $\mathcal{T}_1$, so $\mathcal{T}_2 \models \mathcal{T}_1$ by definition. Obviously, $i \in \mathsf{concept\_instances}(D, \mathcal{A}, \mathcal{T}_j)$ for $j \in \{1, 2\}$ iff $i : D \in \Sigma_j$ (note that $D \in \mathcal{DL}'$). But then, $i \in \mathsf{concept\_instances}(D, \mathcal{A}, \mathcal{T}_1)$ implies $i \in \mathsf{concept\_instances}(D, \mathcal{A}, \mathcal{T}_2)$, contradicting the assumption. □

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

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

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 (Transitive Role Replacement). *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 we have that*

$$\mathsf{concept\_instances}(D, \mathcal{A}, \mathcal{T}) = \mathsf{concept\_instances}(D, \mathcal{A}, \mathcal{T}^{K_4}).$$

We assume a corresponding function $\mathsf{get\_K4\_closure}$ 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 $\mathsf{flatten\_tbox}$ 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 approximate($\mathcal{T}$)
Parameter: $\mathcal{SHI}$ TBox $\mathcal{T}$
    $\mathcal{T}$ := flatten_tbox(get_K4_closure($\mathcal{T}$))
```

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

```
    $\mathcal{T}' := \{\, C \dot{\sqsubseteq} D \mid \mathcal{T} \models C \dot{\sqsubseteq} D, C, D \in N_{CN} \,\}$
    while $\mathcal{T} \neq \emptyset$
        $axiom :=$ select_axiom$(\mathcal{T})$
        $\mathcal{T}' := \mathcal{T}' \cup$ approximate_axiom$(axiom, \mathcal{T}')$
        $\mathcal{T} := \mathcal{T} \setminus \{axiom\}$
    end while
return $\mathcal{T}'$
```

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*; $C \to_G C'$ means that C is generalized to C':

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

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: $(\exists R.C \sqcap E) \sqcup (\exists R.C \sqcap F) \to_G \exists R.\top \sqcup F$, since $(\exists R.C \sqcap E) \to_G \exists R.\top$ and $(\exists R.C \sqcap F) \to_G F$. 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 $\forall R.C \to_G \top$ 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 does not 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, $\mathsf{specialize}$, exploit a similar function $\mathsf{syns\_or\_children}$ 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, $\mathsf{specialize}$ 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 $\mathsf{approximate\_axiom}$ 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 $\mathcal{T}$.*

Proof. (Sketch) This can be shown by induction on $\mathcal{T}$. It is easy to check that $\mathsf{approximate\_axiom}(C \dot{\sqsubseteq} D, \mathcal{T}') \cup \mathcal{T} \models \{C \dot{\sqsubseteq} D\} \cup \mathcal{T}$ for every $\mathcal{T}'$ by monotonicity and by definition of $\mathsf{approximate\_axiom}$.

An Implementation of the Approximation Algorithm We have eliminated the non-determinism in the $\mathsf{approximate}$ algorithm by implementing it in a depth-first (backtracking) search algorithm. Thus, for a given *axiom*, $\mathsf{approximate\_axiom}(\mathit{axiom})$ 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.

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 Filter Step 2: One Step Nodes

In the previous section we have introduced one way to pre-filter individuals for instance retrieval queries - based on TBox approximation to a less expressive ontology language. However, this step is usually quite time-consuming and can only be done offline. We propose another filter step, based on reduction of the size of the input, while remaining the level of expressivity. The idea is to extract small subsets of the ABox, which allow for sound (and possibly complete) reasoning over an individual. In addition, we will show that many of these extracted structures are *similar* to each other and can be handled in one reasoning step.

The basic idea is to define a notion of so-called pseudo node successors, which represent the directly asserted successors of a named individual in an ABox. Then, for each individual in the ABox the information about all pseudo node successors plus the information about the original individual will be combined to obtain so-called one step nodes. These one step nodes can be grouped and used to answer instance checking and instance retrieval queries directly as shown below.

Definition 4 (Pseudo Node Successor). *Given an ABox $\mathcal{A}$, a* Pseudo Node Successor *of an individual $a \in Ind(\mathcal{A})$ is a pair $pns^{a,\mathcal{A}} = \langle \mathbf{rs}, \mathbf{cs} \rangle$, s.t. $\exists a_2 \in Ind(\mathcal{A})$ with*

1. $\forall R \in \mathbf{rs}.(R(a, a_2) \in \mathcal{A} \vee R^{-}(a_2, a) \in \mathcal{A})$,
2. $\forall C \in \mathbf{cs}.C(a) \in \mathcal{A}$ *and*
3. $\mathbf{rs}$ *and* $\mathbf{cs}$ *are maximal.*

The third criteria (maximality) is important to ensure that for each pair of named individuals $\langle a_1, a_2 \rangle \in Ind(\mathcal{A}) \times Ind(\mathcal{A})$, we have that a_2 can be represented as *exactly* one kind of pseudo node successor for a_1.

Next, we combine all pseudo node successors of an individual a in an ABox $\mathcal{A}$ and add the directly asserted concepts of a, to create a summarization representative, called *one step node*.

Definition 5 (One Step Node). *Given an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{A} \rangle$ and an individual $a \in Ind(\mathcal{A})$, the* One Step Node of a for $\mathcal{A}$, *denoted $osn^{a,\mathcal{A}}$, is a pair $osn^{a,\mathcal{A}} = \langle \mathbf{rootconset}, \mathbf{pnsset} \rangle$, s.t.*

- $\mathbf{rootconset} = \{C | C(a) \in \mathcal{A}\}$ *and*
- $\mathbf{ansset}$ *is the set of all pseudo node successors of individual a.*

Example 2 (Example for One Step Nodes). Given ontology $\mathcal{O}_{Ex1}$ from Example 1, one step nodes are for instance:

$$\begin{aligned} osn^{mae,\mathcal{A}_{Ex1}} =& \langle \{Professor\}, \{\langle \{headOf\}, \{Department\} \rangle, \\ & \langle \{teaches\}, \{UndergraduateCourse\} \rangle \} \rangle \\ osn^{c5,\mathcal{A}_{Ex1}} =& \langle \{Student\}, \\ & \{\langle \{takes^{-}\}, \{Student\} \rangle, \langle \{teaches^{-}\}, \{Professor\} \rangle \} \rangle \\ osn^{ann,\mathcal{A}_{Ex1}} =& \langle \{Professor\}, \{\langle \{headOf\}, \{Department\} \rangle, \\ & \langle \{teaches\}, \{UndergraduateCourse\} \rangle \} \rangle \end{aligned}$$

It can already be seen from Example 2, that individuals often have similar one step nodes. During reasoning, these individuals can be handled together, while still remaining soundness (and even possibly completeness). This is formalized next.

Definition 6 (One Step Node Similarity). *Two individuals a_1 and a_2 are called* One Step Node Similar *for an ABox $\mathcal{A}$, if we have $osn^{a_1,\mathcal{A}} = osn^{a_2,\mathcal{A}}$.*

To formally define entailment for one step nodes, we need to have some kind of serialization of one step nodes into an ontology. This serialization is formally defined in Definition 7 for pseudo node successors and in Definition 8 for one step nodes.

Definition 7 (Pseudo Node Successor ABox Realization). *Given one pseudo node successor $pns^{a,\mathcal{A}} = \langle \mathbf{rs}, \mathbf{cs} \rangle$ and an individual a_2, an* ABox realization of $pns^{a,\mathcal{A}}$ with respect to a_2, *denoted $ABox^{a_2}(ans^{a,\mathcal{A}})$, is*

$$ABox^{a_2}(ans^{a,\mathcal{A}}) = \bigcup_{C \in \mathbf{cs}} \{C(a_2)\} \cup \bigcup_{R \in \mathbf{rs}} \{R(a, a_2)\}$$

Definition 8 (One Step Node ABox Realization). *Given a one step node $osn^{a,\mathcal{A}} = \langle \mathbf{rootconset}, \mathbf{pnsset} \rangle$, let $a_1, ..., a_n$ be individuals distinct with a, such that $n = |\mathbf{pnsset}|$. Furthermore let $\mathbf{f}$ be a bijective function from $\mathbf{pnsset}$ to $\{a_1, ..., a_n\}$. An* ABox realization of $osn^{a,\mathcal{A}}$, *denoted $ABox(osn^{a,\mathcal{A}})$, is*

$$ABox(osn^{a,\mathcal{A}}) = \{C(a) \mid C \in \mathbf{rootconset}\} \cup \bigcup_{pns^{a,\mathcal{A}} \in \mathbf{pnsset}} ABox^{\mathbf{f}(pns^{a,\mathcal{A}})}(ans^{a,\mathcal{A}})$$

Definition 9 (One Step Node Entailment). *Given a TBox $\mathcal{T}$ and a one step node $osn^{a,\mathcal{A}} = \langle \mathbf{rootconset}, \mathbf{pnsset} \rangle$ for an individual $a \in Ind(\mathcal{A})$, we say that $osn^{a,\mathcal{A}}$* entails an atomic concept assertion *$C(a)$, denoted $osn^{a,\mathcal{A}} \models_{\mathcal{T}} C(a)$, if we have $\langle \mathcal{T}, ABox(osn^{a,\mathcal{A}}) \rangle \models C(a)$. If $\mathcal{T}$ is clear from the context, then it is omitted, e.g. then we denote concept assertion entailment with $osn^{a,\mathcal{A}} \models C(a)$.*

Lemma 1 (One Step Node Entailment is Sound for Instance Checking). *Given an ontology $\langle \mathcal{T}, \mathcal{A} \rangle$, an individual $a \in Ind(\mathcal{A})$ and a one step node $osn^{a,\mathcal{A}} = \langle \mathbf{rootconset}, \mathbf{pnsset} \rangle$ for a, we have that $osn^{a,\mathcal{A}}$ is sound for instance checking in ontology $\langle \mathcal{T}, \mathcal{A} \rangle$.*

Proof (Proof of Lemma 1). For the proof we have to show that $osn^{a,\mathcal{A}} \models C(a) \implies \langle \mathcal{T}, \mathcal{A} \rangle \models C(a)$. By contraposition, we obtain $\langle \mathcal{T}, \mathcal{A} \rangle \not\models C(a) \implies osn^{a,\mathcal{A}} \not\models C(a)$. We assume $\langle \mathcal{T}, \mathcal{A} \rangle \not\models C(a)$. That means, there exists an interpretation $\mathcal{I}$, such that $\mathcal{I} \models \langle \mathcal{T}, \mathcal{A} \rangle$, but $\mathcal{I} \not\models C(a)$. By Definition 4, Definition 5 and Definition 8, there must exist a homomorphism π from $ABox(osn^{a,\mathcal{A}})$ to $\mathcal{A}$. Let $\mathcal{I}_\pi$ be the interpretation obtained from $\mathcal{I}$, by using π on the individuals in $ABox(osn^{a,\mathcal{A}})$. We can conclude that $\mathcal{I}_\pi \models osn^{a,\mathcal{A}}$. By $\mathcal{I}_\pi \not\models C(a)$ (which follows from $\mathcal{I} \not\models C(a)$ and the unchanged concept labels of the domain elements of $\mathcal{I}$ in $\mathcal{I}_\pi$), we obtain $osn^{a,\mathcal{A}} \not\models C(a)$.

To summarize, one step nodes enable us to pre-filter individuals for an instance retrieval query in a sound way. It will be shown below, how one step nodes can also be used for complete pre-selection. Given TBox approximation and one step nodes as filter techniques, we have a means to determine subsets and supersets of solutions for instance retrieval queries. Next, we introduce the refine step, which is used to determine the status of possible solutions, which are obtained by complete filtering, but are not included in any sound filtering.

5 The Refine Step: Island-Based Instance Retrieval

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 [4] for the DL $\mathcal{ALCHI}$. This section is only intended as a overview of the refine step. Detailed explanations and proofs are omitted here.

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 $\langle \mathcal{T}, \mathcal{A} \rangle \vDash^? C(a)$ can be reduced to checking whether $\langle \mathcal{T}, \mathcal{A} \cup \{\neg C(a)\} \rangle$ is unsatisfiable via a tableau algorithm).

First, we transform the ontology into some kind of normal form, called *shallow normal form*. For the details of the transformation please refer to [4]. 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" via role assertions.

Definition 10. *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 $INC(\langle \mathcal{T}, \mathcal{A}_2 \rbrace \rangle)$, where*

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

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 $\mathcal{O}$-inseparable role assertion in the original ABox, until no more $\mathcal{O}$-separable role assertions are left, i.e. in a breath-first-search style. For the details of this algorithm please refer to [4]. 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.

To summarize, we have a locality-based extraction algorithm, which allows us to perform instance checking for a given individual. This is important, since we can avoid loading the whole ABox for instance checking. We show a detailed example for instance retrieval in the following section.

6 Running Example

The extended $\forall$-info structure, i.e. the core information structure for deciding $\mathcal{O}$-separability, for $\mathcal{T}_{Ex1}$ is:

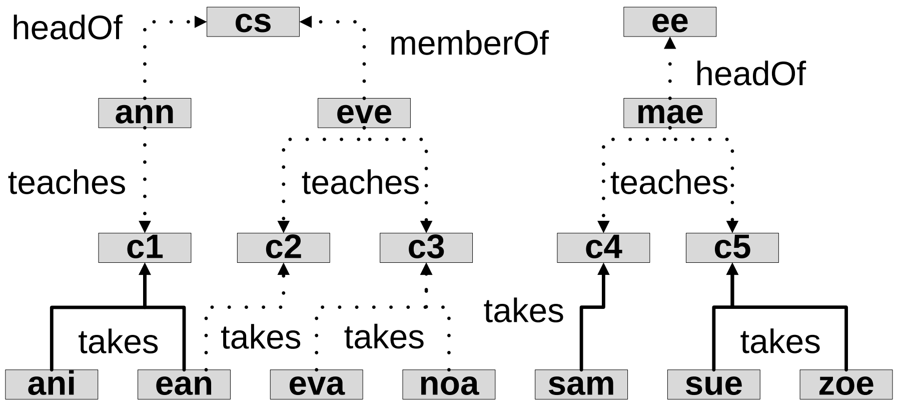

Fig. 3. Individual relationships and separability for Example 1

$$extinfo(R) = \begin{cases} \{\neg Department\}, \text{if } R = headOf \\ \{\neg Course, Course, GraduateCourse\}, \text{if } R = takes \\ \emptyset, \text{else} \end{cases}$$

The $\mathcal{O}$-separability among individuals of $\mathcal{A}_{Ex1}$ are depicted in Figure 3. Separable role assertions are indicated with a dashed line. For instance, the role assertion $takes(ani, c1)$ is not separable, since the concept description $GraduateCourse$ can be propagated via role description $takes$. Please note that all these role assertions would be separable, if we had a disjointness axiom for $GraduateCourse$ and $UndergraduateCourse$. However, to show the behavior of reasoning in case of inseparability, we omitted the disjointness axiom here.

6.1 Instance Checking

First, we discuss instance checking, since it is a the basic problem underlying instance retrieval. For instance checking, we are given an ontology $\mathcal{O}$, an atomic concept description C and an individual $a \in Ind(\mathcal{A})$, and we want to find out, whether $\mathcal{O} \models C(a)$. The process of instance checking is done in two steps. First, we take the one step node $osn^{a,\mathcal{A}}$ of individual a and see, whether $osn^{a,\mathcal{A}} \models C(a)$. If yes, then we are done, since we know that one step nodes are sound for instance checking with respect to the input ontology $\mathcal{O}$. If $osn^{a,\mathcal{A}} \not\models C(a)$, then we distinguish two cases. First, if we have $osn^{a,\mathcal{A}} \models \neg C(a)$, then then we can conclude by soundness that $\mathcal{O} \models \neg C(a)$ and $\mathcal{O} \not\models C(a)$, if the whole ontology is assumed to be consistent. If we do not have $osn^{a,\mathcal{A}} \not\models \neg C(a)$ either, then we need to load the individual island ISL_a for individual a and perform instance checking over ISL_a.

As an example for instance checking, we want want to check, whether the individual *ann* is an instance of concept description *Chair* with respect to the ontology $\mathcal{O}_{Ex1}$. The one step node $osn^{ann,\mathcal{A}_{Ex1}}$ is defined as follows:

$$\begin{aligned} osn^{ann,\mathcal{A}_{Ex1}} =& \langle\{Professor\}, \{\langle\{headOf\}, \{Department\}\rangle, \\ & \langle\{teaches\}, \{UndergraduateCourse\}\rangle\}\rangle \end{aligned}$$

One possible one step node realization of $osn^{ann,\mathcal{A}_{Ex1}}$ is

$$ABox(osn^{ann,\mathcal{A}_{Ex1}}) = \{Professor(ann), headOf(ann, a_1), teaches(ann, a_2)\}.$$

It is easy to see that we have $\langle\mathcal{T}, ABox(osn^{ann,\mathcal{A}_{Ex1}})\rangle \models Chair(ann)$, and thus we have $osn^{ann,\mathcal{A}_{Ex1}} \models_{\mathcal{T}_{Ex1}} Chair(ann)$ and by soundness of one step nodes $\mathcal{O}_{Ex1} \models Chair(ann)$.

As a second example for instance checking, we want want to check, whether the individual $c1$ is an instance of concept description *Chair* with respect to the ontology $\mathcal{O}_{Ex1}$. The one step node $osn^{c1,\mathcal{A}_{Ex1}}$ is defined as follows:

$$\begin{aligned} osn^{c1,\mathcal{A}_{Ex1}} =& \langle\{UndergraduateCourse\}, \{\langle\{teaches^-\}, \{Professor\}\rangle, \\ & \langle\{takes^-\}, \{Student\}\rangle\}\rangle \end{aligned}$$

One possible one step node realization of $osn^{c1,\mathcal{A}_{Ex1}}$ is

$$\begin{aligned} ABox(osn^{c1,\mathcal{A}_{Ex1}}) = \{& UndergraduateCourse(c1), teaches(a_1, c1,), \\ & takes(a_2, c1)\}. \end{aligned}$$

It is easy to see that we have $\langle\mathcal{T}, ABox(osn^{c1,\mathcal{A}_{Ex1}})\rangle \not\models Chair(c1)$. In this case, the one step node does not indicate entailment. However, another simple instance check can help us to avoid using the individual island here. It is easy to see that we have $\langle\mathcal{T}, ABox(osn^{c1,\mathcal{A}_{Ex1}})\rangle \models \neg Chair(c1)$. And this means that we have $\langle\mathcal{T}_{Ex1}, \mathcal{A}_{Ex1}\rangle \models \neg Chair(c1)$. Thus, in some cases, the negated instance check for one step nodes can also help us to avoid performing reasoning on (more complex) individual islands. However, if the negated instance check fails, then we really have to fall back to the use of sound and complete individual islands.

6.2 Instance Retrieval

In the following, we discuss instance retrieval optimization over ontologies. We propose to use both filter techniques in parallel and then combine the results. In our experiments so far, the one step node-based technique always outperforms the TBox approximation. This is due to the high complexity of approximation mentioned above. Thus, in the following we will focus on the explanation of one step node-based instance retrieval.

The naive approach would be to apply instance checking techniques to each named individual in the ABox. For ontology $\mathcal{O}_{Ex1}$, we would have to perform

17 instance checks in that case. However, we have introduced the notion of one step node similarity. The idea is that similar one step nodes entail the same set of concept descriptions for the named root individual. Given the set of all one step nodes for an input ontology, we can reduce the number of instance checks.

For example, assume that we want to perform instance retrieval for the concept description *Chair* with respect to ontology $\mathcal{O}_{Ex1}$. First, we compute the one step node for each individual in $\mathcal{A}_{Ex1}$. The resulting one step nodes are shown in Figure 4:

$$osn^{ani,\mathcal{A}_{Ex1}} = osn^{sam,\mathcal{A}_{Ex1}} = osn^{sue,\mathcal{A}_{Ex1}} = osn^{zoe,\mathcal{A}_{Ex1}} =$$
$$\langle\{Student\}, \{\langle\{takes\}, \{UndergraduateCourse\}\rangle\}\rangle$$
$$osn^{ean,\mathcal{A}_{Ex1}} =$$
$$\langle\{Student\}, \{\langle\{takes\}, \{UndergraduateCourse\}\rangle,$$
$$\langle\{takes\}, \{GraduateCourse\}\rangle\}\rangle$$
$$osn^{eva,\mathcal{A}_{Ex1}} = osn^{noa,\mathcal{A}_{Ex1}} =$$
$$\langle\{Student\}, \{\langle\{takes\}, \{GraduateCourse\}\rangle\}\rangle$$
$$osn^{c1,\mathcal{A}_{Ex1}} = osn^{c4,\mathcal{A}_{Ex1}} = osn^{c5,\mathcal{A}_{Ex1}} =$$
$$\langle\{UndergraduateCourse\}, \{\langle\{teaches^-\}, \{Professor\}\rangle, \langle\{takes^-\}, \{Student\}\rangle\}\rangle$$
$$osn^{c2,\mathcal{A}_{Ex1}} = osn^{c3,\mathcal{A}_{Ex1}} =$$
$$\langle\{GraduateCourse\}, \{\langle\{teaches^-\}, \{Professor\}\rangle, \langle\{takes^-\}, \{Student\}\rangle\}\rangle$$
$$osn^{ann,\mathcal{A}_{Ex1}} = osn^{mae,\mathcal{A}_{Ex1}} =$$
$$\langle\{Professor\}, \{\langle\{headOf\}, \{Department\}\rangle, \langle\{teaches^-\}, \{UndergraduateCourse\}\rangle\}\rangle$$
$$osn^{eve,\mathcal{A}_{Ex1}} =$$
$$\langle\{Professor\}, \{\langle\{memberOf\}, \{Department\}\rangle, \langle\{teaches^-\}, \{GraduateCourse\}\rangle\}\rangle$$
$$osn^{cs,\mathcal{A}_{Ex1}} =$$
$$\langle\{Department\}, \{\langle\{headOf^-\}, \{Professor\}\rangle, \langle\{memberOf^-\}, \{Professor\}\rangle\}\rangle$$
$$osn^{ee,\mathcal{A}_{Ex1}} =$$
$$\langle\{Department\}, \{\langle\{headOf^-\}, \{Professor\}\rangle\}\rangle$$

Fig. 4. One step nodes for individuals

Instead of 17 instance checks for 17 named individuals, we are left with 9 instance checks over 9 one step nodes. For ontologies with a larger assertional part, similarity of one step nodes will reduce the number of instance checks usually by orders of magnitudes.

By performing instance checks for concept description $Chair$ over the 9 one step nodes, we can conclude that individual *ann* and individual *mae* are instances of $Chair$. Additional instance checks for concept description $\neg Chair$ yields that $c1$, $c2$, $c3$, $c4$, $c5$, *ani*, *ean*, *eva*, *noa*, *sam*, *sue* and *zoe* are instances of concept description $\neg Chair$, and therefore are not instances of concept description $Chair$. After the one step node filter process, we are left to check only *three* individuals for being an instance of concept description $Chair$: *cs*, *ee* and *eve*. For these three individuals we load the individual island and perform instance checking.

Thus, our simple filter techniques allows to cut off 14 out of 17 individuals immediately. And the remaining three individuals are checked by our optimized refine step.

In the following section we evaluate our proposal on a real world multimedia ontology.

7 Preliminary Evaluation and Related Work

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 $\mathcal{ALCH}f$. 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 ≈ 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).

The first filter algorithm based on TBox approximation is quite time consuming. Thus, we have evaluated our one step node-based filtering technique with respect to the AEO ontology. The results are quite encouraging, since the instance retrieval over AEO was possible in only 0.8 seconds.

In addition, we have evaluated our filter and refine paradigm for the benchmark ontology [7]. The results for the TBox approximation filter are rather discouraging again. Only after removing several disjointness axioms from the TBox, we can compute a consistent approximation. However, our evaluation for the one step nodes is promising. We have ran tests for different numbers of universities, and the number of one step nodes is constant. For example, for the LUBM dataset with 100 universities, we have 6.645.928 individuals, but only 276 distinct one step nodes. Out of these one step nodes, 255 can be even used for complete instance checking! Thus, we have a compact data-structure, which enables for sound and complete reasoning for around 93 percent of the individuals. This is a big step towards scalable instance retrieval over ontologies with a large ABox.

In [8], the authors propose an algorithm to convert a TBox into Datalog. Although the transformation is quite efficient using the algorithms from KAON2, the resulting ontologies introduce many wrong solution, depending on the instantiation of their algorithms. Furthermore, the authors focus more on computability of TBox approximization, than on the actual retrieval. In addition, we introduce similarity notions, to further dramatically reduce the time needed for reasoning. In [9], a notion of proxy (or summary) ABox is introduced. The idea is to merge individuals in an ABox - based on their concept assertion sets, and then perform reasoning on the result. Once the summary ABox is inconsistent, the system has to perform a refinement step, where individual mergings are undone and new (more coarse) summarizations are evaluated. While we summarize individuals in one step nodes with the aim of remaining soundness of reasoning, the basic idea of [9] is to remain completeness. Thus, our approach can be seen as contrary to [9]. In [10], instance retrieval optimization techniques are introduced. Our one step node similarity can also be used as a specific instantiation of their binary instance retrieval strategy, as an alternative to island-based instance retrieval. In [11], first partitioning algorithms for the assertional parts of ontologies were proposed. However, as shown in [12], the given criteria are often too soft to partition even simple TBoxes. Furthermore, in case of ontology updates, the partitioning has to be recomputed from the scratch. Our one step nodes can be updates easily.

8 Conclusions and Future Work

Summing up, the evaluation in the previous section has shown that the results for both filter steps are twofold. The results for our TBox approximation are rather discouraging for both ontologies, AEO, as well as LUBM. However, we think that this is basically because of the lack of heuristics. With the correct heuristics, we think it is possible to compute consistent approximations more efficiently. These heuristics are to be developed in future work.

The results for our second filter step, one step nodes, are very encouraging. Especially the results for LUBM show, that our similarity measure for individuals can be used to greatly improve instance retrieval times. We are not aware of any other in-memory data structure, which allows for such efficient candidate elimination for LUBM as one step nodes.

This evaluation should be understood as a first preliminary proof of concept of the ideas conveyed in this paper. In the future, we plan to extend our instance retrieval algorithm to more expressive description logics, and evaluate both retrieval techniques further by using more ontologies.

References

1. Haarslev, V., Möller, R., Wessel, M.: RacerPro User's Guide and Reference Manual Version 1.9.1 (May 2007)
2. 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)
3. 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)
4. 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. 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. Tserendorj, T., Rudolph, S., Krötzsch, M., Hitzler, P.: Approximate owl-reasoning with screech. In Calvanese, D., Lausen, G., eds.: RR. Volume 5341 of Lecture Notes in Computer Science., Springer (2008) 165–180
9. Dolby, J., Fokoue, A., Kalyanpur, A., Kershenbaum, A., Schonberg, E., Srinivas, K., Ma, L.: Scalable semantic retrieval through summarization and refinement. In: AAAI'07: Proceedings of the 22nd national conference on Artificial intelligence, AAAI Press (2007) 299–304

10. Haarslev, V., Moeller, R.: Optimization techniques for retrieving resources described in owl/rdf documents: First results. In: Ninth International Conference on the Principles of Knowledge Representation and Reasoning, KR 2004. (2004) 2–5
11. Guo, Y., Heflin, J.: A scalable approach for partitioning owl knowledge bases. In: Workshop on Scalable Semantic Web Knowledge Base Systems (SSWS2006, Springer (2006) 636–641
12. Wandelt, S.: Partitioning owl knowledge bases - revisited and revised. In Baader, F., Lutz, C., Motik, B., eds.: Description Logics. Volume 353 of CEUR Workshop Proceedings., CEUR-WS.org (2008)