Sound and Complete SHI Instance Retrieval for 1 Billion ABox Assertions

Sebastian Wandelt, Ralf Möller

Institute for Software Systems,
Hamburg University of Technology
wandelt@tuhh.de, moeller@tuhh.de

Abstract. In the last years, reasoning over very large ontologies became more and more important. In our contribution, we propose a reasoning infrastructure, which can perform instance checking and instance retrieval over ontologies with semi-expressive terminological knowledge and large assertional parts.
The key idea is to 1) use modularizations of the assertional part, 2) use some kind of intermediate structure to find similarities between individual modules, and 3) store information efficiently on external memory. For evaluation purposes, experiments on benchmark and real world ontologies were carried out. We show that our reasoning infrastructure can handle up to 1 billion ABox assertions. To the best of our knowledge this is the first system to provide sound and complete reasoning over SHI ontologies with such a huge number of assertions.

1 Introduction

The Semantic Web is intended to bring structure to the meaningful content of web pages and to create an accessible environment for software agents. Ontologies are one way of representing the knowledge of these agents. The idea to represent datasets on the Internet with ontologies was first widely made public in [BLHL01]. Since then the Semantic Web became a widely used buzzword.

There is increased interest in the development of Semantic Web applications, e.g. digital libraries, community management, and health-care systems. As the Semantic Web evolves, the amount of data available in these applications is growing with an incredible speed. Since the size of the Semantic Web is expected to further grow in the coming years, scalability and performance of Semantic Web systems become increasingly important. Usually, such systems deal with information described in description-logic based ontology languages such as OWL [HKP+09], and provide services for storing, querying, and updating large numbers of facts.

Decidability results for many expressive description logics and for query answering over these description logics have been shown. However, early tableau-based description logic reasoning systems, e.g. Racer [HMW04] and Pellet [SPG$^+$07], do not perform well with large ontologies since the implementation of tableau algorithms is built based on efficient main memory data structures.

There exists a lot of research to identify tractable description logics. For example the descriptions logic $\mathcal{EL}$ and extensions up to $\mathcal{EL}^{++}$, introduced in [BBL08], admit reasoning in polynomial time for classification and instance checking. Another lightweight description logic (family) is *DL-LITE* [CDGL$^+$05]. *DL-LITE* allows the use of relational database management systems for query answering. Another tractable fragment is the rule-based language OWL-R, introduced in [HKP$^+$09]. All tractable fragments have in common that the set of constructors in the ontology language is restricted in order to obtain efficient reasoning algorithms for query answering. However, in practical applications, users often need more expressive languages.

The increasing growth of Semantic Web applications also led to the development of a new class of external memory-based retrieval systems, so called triple stores. Originally motivated to store RDF schema information, see [Bec04], a general architecture to store triples was proposed in [BKvH03]. In the recent years, the number of these stores substantially increased, see for instance Franz AllegroGraph [Fra11] or OWLIM [Kir06]. Although the creators of triple stores continuously come up with more impressive performance evaluation results, there are two basic problems with these statistics. First, in general, it is not clear what kind of reasoning takes place inside the triple store during retrieval - it can be anything from pure lookup to complex description logic reasoning. Second, the hardware test configurations used by triple stores creators seem to be a little over the line. For instance, if one uses four computers with 48 GB of main memory each, then it is not a big surprise that the system is able to handle datasets in the order of several GB. This scenario seems to be at odds with the original intention of triple stores - managing data in external memory.

Another approach to overcome the problem of reasoning over large ontologies is to approximate the ontology by a more compact representation or in a weaker description logic. In [PTZ09], the authors propose to reuse the idea of knowledge compilation to approximate ontologies in a weaker ontology language. For the ontology language of their choice, i.e. *DL-LITE*, efficient query answering algorithms with polynomial data complexity exist. Reasoning on the approximated ontology allows to include/reject potential answers with respect to the original ontology. A similar direction was taken in [RPZ10], where the terminology part of an ontology is approximated to the description logic $\mathcal{EL}^{++}$. The results from the approximated ontology are used for more efficient classification over the original ontology. The classification results can then be used for more efficient retrieval as well.

Another approach focusing on reasoning over instances in large ontologies is presented in [TRKH08]. The algorithms in [TRKH08] are based on KAON2 [Mot08] algorithms, which transform the terminological part of an ontology into

Datalog [MW88]. Depending on the transformation strategy, the obtained Datalog program can be used for sound or complete reasoning over instances in the source ontology. The preceding approximation approaches rely on expressivity reduction of the ontology language.

A different approach is proposed in [DFK$^+$07], based on summarization and refinement. First, a summarization of the assertional part is created by aggregating individuals. This is part of a setup step that can be performed offline, i.e. before query answering takes place. Queries are then executed over the summarization. During the summarization process, one has to take care of inconsistencies. If the summarization leads to inconsistencies, previously merged individuals have to be broken up again.

In our work, we propose a system which can handle (i.e. perform sound and complete instance retrieval) more than 1 billion ABox assertions. For the syntax and semantics of the description logic $\mathcal{SHI}$ please refer to [Baa99]

2 Ingredients for Efficient Instance Retrieval

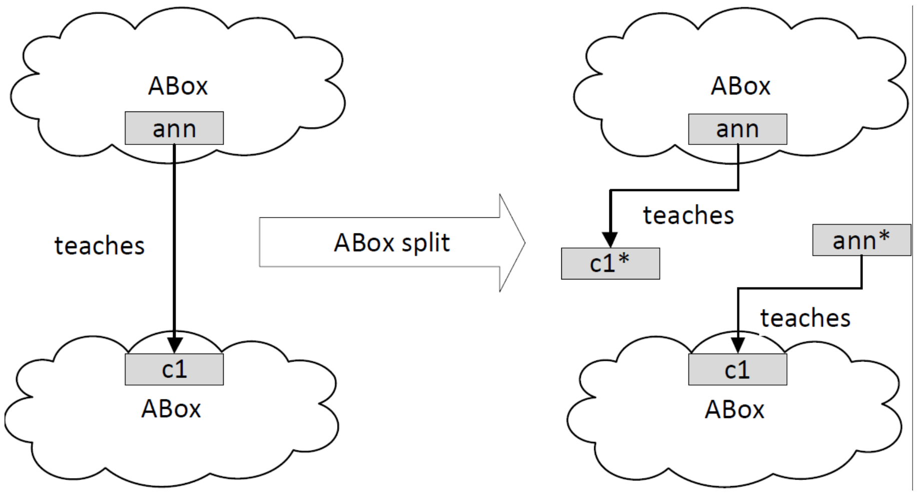
2.1 ABox Modularization

In [WM08], a method is proposed to identify the relevant information (assertions) to reason about an individual. The main motivation is to enable in-memory reasoning over large ontologies, i.e. ontologies with a large ABox, for traditional tableau-based reasoning systems.

Inspired by graph partitioning approaches, we developed techniques to break down an ABox into smaller chunks (modules), such that decision problems can be solved by considering these smaller parts only.

Naive modularization techniques (based on connectedness of individuals) are usually not sufficient for ABox modularizations, since most individuals are somehow connected to each other. We extend the naive modularization technique by introducing so-called *ABox splits*. Informally speaking, an ABox split breaks up a role assertion in an ABox, while preserving the semantics (this is formalized below). The idea is depicted in Figure 1. The clouds in Figure 1 indicate a set of ABox assertions. We split up the role assertion $teaches(ann, c1)$, create two new individuals (ann^* and $c1^*$), and keep the concept assertions for each fresh individual copy. After applying all possible ABox splits to an ABox of an ontology, a graph-based ABox modularization becomes more fine-grained, i.e. one obtains more (and smaller) modules.

Fig. 1 Intuition of an ABox split



Definition 1 **($\mathcal{SHI}$-splittability of Role Assertions).** *Given a $\mathcal{SHI}$-ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ and a role assertion $R(a_1, a_2)$, we say that $R(a_1, a_2)$ is $\mathcal{SHI}$-*splittable with respect to $\mathcal{O}$ *if*

1. *there exists no transitive role R_2 with respect to $\mathcal{R}$, such that $\mathcal{R} \models R \sqsubseteq R_2$,*
2. *for each C that can be propagated over the role description R*
   - *$C = \bot$ or*
   - *there exists a concept description C_2, such that $C_2(b) \in \mathcal{A}$ and $\mathcal{T} \models C_2 \sqsubseteq C$ or*
   - *there exists a concept description C_2, such that $C_2(b) \in \mathcal{A}$ and $\mathcal{T} \models C \sqcap C_2 \sqsubseteq \bot$*

   and
3. *for each $C \in extinfo^{\forall}_{\mathcal{T},\mathcal{R}}(R^-)$*
   - *$C = \bot$ or*
   - *there exists a concept description C_2, such that $C_2(a) \in \mathcal{A}$ and $\mathcal{T} \models C_2 \sqsubseteq C$ or*
   - *there exists a concept description C_2, such that $C_2(a) \in \mathcal{A}$ and $\mathcal{T} \models C \sqcap C_2 \sqsubseteq \bot$.*

It can be shown that ABox splits are consistency preserving, if the role assertion is $\mathcal{SHI}$-splittable. The idea is that each tableau proof which makes use of propagated concept descriptions (via a $\forall$-tableau rule application) can be rewritten into a tableau proof without using the $\forall$-tableau rule application.

An individual island can be computed following the approach in [WM08]: given a start (root) individual, one can perform a graph search following all $\mathcal{SHI}$-unsplittable role assertions. This individual island is then sound and complete for instance checking w.r.t. the root individual.

2.2 One-Step nodes

We introduce a specialization of individual islands next. The basic idea is to define a notion of so-called pseudo node neighbors, 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 is combined, to obtain so-called one-step nodes. In addition to similarity detection, these one-step nodes can be used to answer instance checking and instance retrieval queries directly (always sound, and possible in a complete manner).

First, in Definition 2, we formally define a pseudo node successor for an individual with respect to an ABox.

Definition 2 (Pseudo Node Successor). *Given an ABox $\mathcal{A}$, a* pseudo node successor *of a named individual a is a pair $pns^{a,\mathcal{A}} = \langle \mathbf{rs}, \mathbf{cs} \rangle$, such that $\exists a_2 \in Ind(\mathcal{A})$ with*

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

Next, we combine all pseudo node successors of a named individual a in an ABox $\mathcal{A}$, the reflexive role assertions for a, and the directly asserted concepts of a, in order to create a summarization representative, called one-step node.

Definition 3 (One-Step Node). *Given an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$ and an individual $a \in NInd(\mathcal{A})$, the* one-step node of a for $\mathcal{A}$, *denoted $osn^{a,\mathcal{A}}$, is a tuple $\langle \mathbf{rootconset}, \mathbf{reflset}, \mathbf{pnsset} \rangle$, such that $\mathbf{rootconset} = \{C | C(a) \in \mathcal{A}\}$, $\mathbf{reflset} = \{R | R(a, a) \in \mathcal{A} \vee R^-(a, a) \in \mathcal{A}\}$, and $\mathbf{pnsset}$ is the set of all pseudo node successors of individual a. The set of all possible one-step nodes is denoted* $\mathbf{OSN}$.

Definition 4 (One-Step Node Similarity). *Two individuals a_1 and a_2 are called* one-step node similar *for an ABox $\mathcal{A}$ if $osn^{a_1,\mathcal{A}} = osn^{a_2,\mathcal{A}}$.*

Every one-step node can be used for sound instance checking, since it represents a subset of the ABox assertions from the input ABox. It is clear that not every one-step node is complete for instance checking. However, in case the one-step node coincides with the individual island, then we can show that instance checking over the one-step node is even complete. For this, we define so-called splittable one-step nodes, for which each role assertion to a direct neighbor is $\mathcal{SHI}$-splittable.

Definition 5 (Splittable One-Step Node). *Given an ontology* $\mathcal{O} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$*, an named individual* a*, and a one-step node* $osn^{a,\mathcal{A}} = \langle \mathbf{rootconset}, \mathbf{reflset}, \mathbf{pnsset} \rangle$*, we say that* $osn^{a,\mathcal{A}}$ *is* splittable *if for each* $\langle \mathbf{rs}, \mathbf{cs} \rangle \in \mathbf{pnsset}$*, a fresh individual* $a_2 \notin Ind(\mathcal{A})$*, and for each* $R \in \mathbf{rs}$*, the role assertion axiom* $R(a, a_2)$ *is* $\mathcal{SHI}$*-splittable with respect to ontology* $\mathcal{O}_2 = \langle \mathcal{T}, \mathcal{R}, \mathcal{A}_2 \rangle$ *with*

$$\mathcal{A}_2 = \{C(a) \mid C \in \mathbf{rootconset}\} \cup \{C(a_2) \mid C \in \mathbf{cs}\} \cup \{R(a, a_2)\}.$$

It is easy to see, that splittable one-step nodes can be directly used for sound and complete instance checking. Furthermore, two similar one-step nodes only need to be checked one time during instance retrieval. An example is shown below.

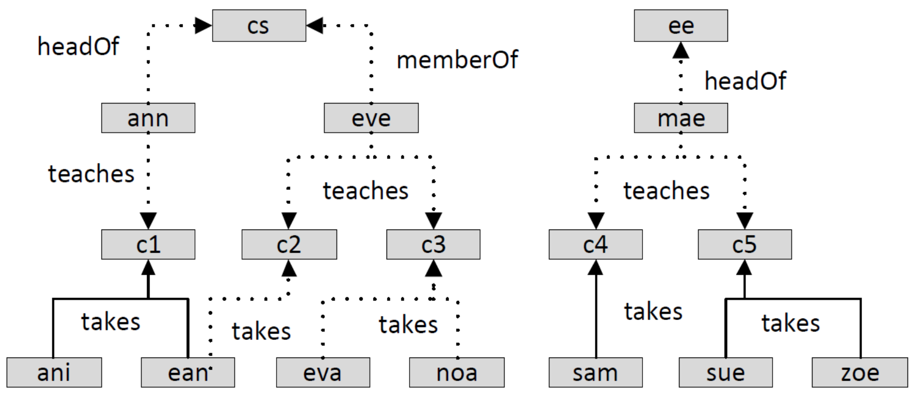
3 Example

In the following, we look at an example to discuss the optimization of instance checking and instance retrieval by the techniques introduced above.

Example 1 (Example Ontology for Island Reasoning). The example ontology $\mathcal{O}_{Ex1} = \langle \mathcal{T}_{Ex1}, \mathcal{R}_{Ex1}, \mathcal{A}_{Ex1} \rangle$ is defined as follows

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

Fig. 2 Individual relationships and splittability for Example 1



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

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. $\mathcal{SHI}$-splittable role assertions are indicated with a dashed line. For instance, the role assertion $takes(ani, c1)$ is not $\mathcal{SHI}$-splittable because the concept description $GraduateCourse$ can be propagated via role description $takes$. Please note that all these role assertions would be $\mathcal{SHI}$-splittable if we had a disjointness axiom for $GraduateCourse$ and $UndergraduateCourse$. However, to show the behavior of reasoning in case of $\mathcal{SHI}$-unsplittability, we omitted the disjointness axiom here.

3.1 Instance Checking

For instance checking, we are given an ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, an atomic concept description C, and an individual $a \in NInd(\mathcal{A})$, and we would like 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 check, 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 $osn^{a,\mathcal{A}}$ is splittable, then we know that we have $\mathcal{O} \not\models C(a)$. Otherwise, if $osn^{a,\mathcal{A}}$ is not splittable, then we load the individual island ISL_a for individual a and perform instance checking over ISL_a.

As an example for instance checking, we would like 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:

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

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 $\langle \mathcal{T}, \mathcal{R}, ABox(osn^{ann,\mathcal{A}_{Ex1}}) \rangle \models Chair(ann)$, and thus we have $ABox(osn^{ann,\mathcal{A}_{Ex1}}) \models_{\mathcal{T}_{Ex1}, \mathcal{R}_{Ex1}} Chair(ann)$ and by soundness of one-step node reasoning $\mathcal{O}_{Ex1} \models Chair(ann)$.

As a second example for instance checking, we would like 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 as follows:

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

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

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

It is easy to see that we have $\langle \mathcal{T}, \mathcal{R}, ABox(osn^{c1,\mathcal{A}_{Ex1}}) \rangle \not\models Chair(c1)$. In this case, the one-step node does not indicate entailment, and since $osn^{c1,\mathcal{A}_{Ex1}}$ is not a splittable one-step node, we should refer to the individual island of individual $c1$. However, another simple instance check can help us to avoid using the individual island here. It is easy to see that we have $\langle \mathcal{T}, \mathcal{R}, ABox(osn^{c1,\mathcal{A}_{Ex1}}) \rangle \models \neg Chair(c1)$, by disjointness of $UndergraduateCourse$ and $Chair$. And this means that we have $\mathcal{O}_{Ex1} \vDash \neg Chair(c1)$. 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, and the one-step node is unsplittable, then we really have to use sound and complete individual islands.

3.2 Instance Retrieval

In the following, we discuss instance retrieval optimization over ontologies. This is a direct extension of instance checking, by using one-step node similarity in addition. The first 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 would like to perform instance retrieval for the concept description $Chair$ with respect to ontology $\mathcal{O}_{Ex1}$. First, we retrieve the one-step node for each named individual in $\mathcal{A}_{Ex1}$. The resulting one-step nodes are shown below:

$$\begin{aligned}
osn^{ani,\mathcal{A}_{Ex1}} &= osn^{sam,\mathcal{A}_{Ex1}} = osn^{sue,\mathcal{A}_{Ex1}} = osn^{zoe,\mathcal{A}_{Ex1}} = \langle\{Student\}, \emptyset, \\
&\quad \{\langle\{takes\}, \{UndergraduateCourse\}\rangle\}\rangle \\
osn^{ean,\mathcal{A}_{Ex1}} &= \langle\{Student\}, \emptyset, \\
&\quad \{\langle\{takes\}, \{UndergraduateCourse\}\rangle, \langle\{takes\}, \{GraduateCourse\}\rangle\}\rangle \\
osn^{eva,\mathcal{A}_{Ex1}} &= osn^{noa,\mathcal{A}_{Ex1}} = \langle\{Student\}, \emptyset, \\
&\quad \{\langle\{takes\}, \{GraduateCourse\}\rangle\}\rangle \\
osn^{c1,\mathcal{A}_{Ex1}} &= osn^{c4,\mathcal{A}_{Ex1}} = osn^{c5,\mathcal{A}_{Ex1}} = \langle\{UndergraduateCourse\}, \emptyset, \\
&\quad \{\langle\{teaches^{-}\}, \{Professor\}\rangle, \langle\{takes^{-}\}, \{Student\}\rangle\}\rangle \\
osn^{c2,\mathcal{A}_{Ex1}} &= osn^{c3,\mathcal{A}_{Ex1}} = \langle\{GraduateCourse\}, \emptyset, \\
&\quad \{\langle\{teaches^{-}\}, \{Professor\}\rangle, \langle\{takes^{-}\}, \{Student\}\rangle\}\rangle \\
osn^{ann,\mathcal{A}_{Ex1}} &= osn^{mae,\mathcal{A}_{Ex1}} = \langle\{Professor\}, \emptyset, \\
&\quad \{\langle\{headOf\}, \{Department\}\rangle, \langle\{teaches^{-}\}, \{UndergraduateCourse\}\rangle\}\rangle
\end{aligned}$$

$$\begin{aligned}
osn^{eve,\mathcal{A}_{Ex1}} &= \langle \{Professor\}, \emptyset, \\
&\quad \{\langle \{memberOf\}, \{Department\}\rangle, \langle \{teaches^{-}\}, \{GraduateCourse\}\rangle\}\rangle \\
osn^{cs,\mathcal{A}_{Ex1}} &= \langle \{Department\}, \emptyset, \\
&\quad \{\langle \{headOf^{-}\}, \{Professor\}\rangle, \langle \{memberOf^{-}\}, \{Professor\}\rangle\}\rangle \\
osn^{ee,\mathcal{A}_{Ex1}} &= \langle \{Department\}, \emptyset, \\
&\quad \{\langle \{headOf^{-}\}, \{Professor\}\rangle\}\rangle.
\end{aligned}$$

Instead of 17 instance checks for 17 named individuals, we are left with 9 instance checks over 9 similar one-step nodes. For ontologies with a larger assertional part, similarity of one-step nodes reduces 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 $Chair$ if the input ontology is consistent. After one-step node retrieval, we are left to check three individuals for being an instance of concept description $Chair$, or not: cs, ee and eve. Usually, one would have to perform instance checks over the three individual islands. However, since the corresponding one-step nodes for these three individuals are splittable, we do not need to do any further checks, since the one-step nodes are already sound and complete for reasoning in $\mathcal{O}_{Ex1}$.

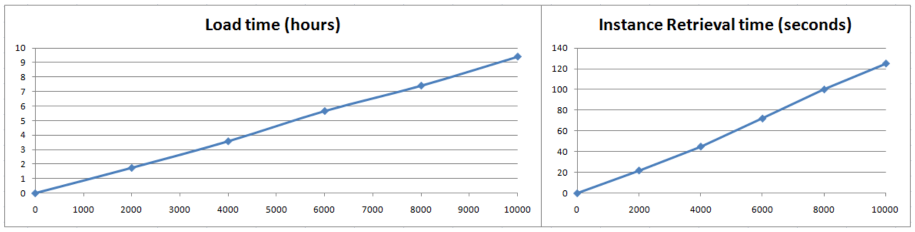
4 Implementation and Evaluation

In the following, we provide a general overview over the prototype. We have implemented the algorithms for reasoning over $\mathcal{SHI}$-ontologies using the programming language Java. We used the description logic reasoner Racer [HMW04] for our evaluation.

While TBox and RBox are kept in main memory, the ABox is serialized to a database. In our prototypical implementation, we used the relational database management system MySQL, see [WA02]. Apart from the assertional data, we also serialize the identifiers of one-step nodes for each individual and information about splittability of role assertions. For each serialized data structure, we have implemented caching algorithms, in order to avoid working on external memory directly for each update. During our experiments, a segmented least recently used cache, see for instance [KLW94], turned out to be most efficient.

We have used two benchmark ontologies for evaluation of our modularization techniques: one synthetic benchmark introduced in [GPH05] and a real world

Fig. 3 Load time and IR time for LUBM (up to 10000 universities)



multimedia annotation ontology used in the CASAM project and introduced in [GMN+09]. The results for both ontologies are outlined below.

4.1 LUBM

The Lehigh University Benchmark, short LUBM, is a synthetic ontology developed to benchmark knowledge base systems with respect to large OWL applications. The ontology is situated in the university domain. The background knowledge, i.e. the terminology, is described in a schema called Univ-Bench, see [GPH05] for an overview over the history, different versions and the predecessor Univ 1.0.

While the terminological part of LUBM is static, the assertional part is dynamic in size and can be generated as big as necessary/desired. The dataset we have used for our experiments was generated by the Univ-Bench artificial data generator.

We determined the number of ABox modules for different LUBM datasets. It turned out that most of the role assertions in LUBM can be broken up and the average module size (number of root individuals in the module) is 1.01.

Our next evaluation measure is load time, shown in Figure 3 on the left. The load time covers loading data from external memory (here: OWL files), applying the update algorithms and serializing the data to a database representation. We process the terminological part first and afterwards the assertional part is loaded. Please note that for 10000 universities the system has to deal with 1.380.000.000 ABox assertions.

In Figure 3, on the right-hand side, we show the instance retrieval time for the concept description *Chair* and different numbers of universities. It can be seen that the instance retrieval time is almost linear - even for more than 170 million individuals in LUBM(10000). Furthermore, we would like to emphasize that most of the instance retrieval time is spent by the database system to lookup the solution names on different pages in the data file. The actual description logic reasoning in our system is roughly constant for the number of universities. We conjecture that, if one finds a more sophisticated way to store the mapping between one-step nodes and individuals, instance retrieval times can be further reduced.

The space needed to store the data for 10000 universities is around 120 GByte (including all index structures).

4.2 CASAM Multimedia Content Ontology

We performed additional test with a real world ontology from the CASAM project. The project is focused on computer-aided semantic annotation of multimedia content. The novelty is the aggregation of human and machine knowledge. For a detailed discussion of the research objectives, see [GMN$^+$10], [PTP10], and [CLHB10]. Within the CASAM project, there is a need to define an expressive annotation language which allows for typical-case reasoning systems. The proposed annotation language is defined by the so-called Multimedia Content Ontology, short MCO, introduced in [GMN$^+$09]. Inspired by the MPEG-7 standard, see [IF02], strictly necessary elements describing the structure of multimedia documents are extracted. The intention is to exploit quantitative and qualitative time information in order to relate co-occurring observations about events in videos. Co-occurrences are detected either within the same or between different modalities, i.e. text, audio and speech, regarding the video shots.

Our tests show that all role assertions in the CASAM test ontology can be split up and therefore reasoning can be reduced to one-step nodes only. We do not provide diagrams for this ontology, since it is too small (only few thousand ABox assertions) and the time for reasoning can hardly be measured correctly.

5 Conclusions

The main goal of our work was to address the problem of instance retrieval over large ABoxes. We focused on the semi-expressive description logic $\mathcal{SHI}$, which can be seen as a first step towards more expressive description logics. We solve the given problem by applying ABox modularization techniques and using a compact representation of individual islands (modules). These compact representations can be used to group similar individuals together and handle them in one step during instance retrieval.

Our evaluation showed that we can handle more than one billion ABox assertions and perform sound and complete instance retrieval for $\mathcal{SHI}$-ontologies.

In the following, we would like to discuss interesting directions for future work.

An extension from the semi-expressive description logic $\mathcal{SHI}$ to $\mathcal{SHIQ}$ should be possible. We think that a syntactical analysis of the TBox and RBox can be used to identify a set of $\mathcal{SHIQ}$-unsplittable role assertions. Our homomorphism-based similarity criteria for individuals cannot be directly applied in the presence of cardinality restrictions. Further extensions, for instance to $\mathcal{SHOIQ}$, might be possible, but will surely require a lot of work and sophisticated analysis techniques.

Another direction for future work is the focus on more expressive query languages. While we focus on instance checking and instance retrieval, the next natural step is conjunctive query answering [GHLS07]. We think that query answering with respect to the class of grounded conjunctive queries, is straightforward. One would have to combine the results from sound (and complete reasoning) in order to identify possible variable bindings. The extension to standard conjunctive queries is without doubt much harder.

Since rules over ontologies have become more important recently, it would be interesting to implement a rule-based query answering engine on top of our ABox modularizations. We already performed first tests. By syntactical analysis of rule bodies we decided which individual islands have to be extended/merged. The first results are quite encouraging.

Finally, more comprehensive experimental studies are required. Recently published work [SCH10] on new data generation algorithms for synthetic test ontologies might be a good place to start from. In general, we believe that our results carry over to other ontologies. However there exist scenarios, especially extensive use of transitive roles, which make it much harder to find fine-grained ABox modularizations.

References

[Baa99] Franz Baader. Logic-Based Knowledge Representation. In *Artificial Intelligence Today*, pages 13–41. Springer-Verlag, 1999.

[BBL08] Franz Baader, Sebastian Brandt, and Carsten Lutz. Pushing the $\mathcal{EL}$ envelope further. In Kendall Clark and Peter F. Patel-Schneider, editors, *In Proceedings of the OWLED 2008 DC Workshop on OWL: Experiences and Directions*, 2008.

[Bec04] Dave Beckett. RDF/XML Syntax Specification (Revised). www.w3.org/TR/REC-rdf-syntax/, 2004.

[BKvH03] Jeen Broekstra, Arjohn Kampman, and Frank van Harmelen. Sesame: An Architecture for Storing and Querying RDF Data and Schema Information. In Dieter Fensel, James A. Hendler, Henry Lieberman, and Wolfgang Wahlster, editors, *Spinning the Semantic Web*, pages 197–222. MIT Press, 2003.

[BLHL01] Tim Berners-Lee, James Hendler, and Ora Lassila. The Semantic Web. *Scientific American*, 284(5):34–43, 2001.

[CDGL$^+$05] Diego Calvanese, Giuseppe De Giacomo, Domenico Lembo, Maurizio Lenzerini, and Riccardo Rosati. DL-Lite: Tractable description logics for ontologies. In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 2005)*, pages 602–607, 2005.

[CLHB10] Chris Creed, Peter Lonsdale, Robert Hendley, and Russell Beale. Synergistic annotation of multimedia content. In *Proceedings of the 2010 Third International Conference on Advances in Computer-Human Interactions*, ACHI '10, pages 205–208, Washington, DC, USA, 2010. IEEE Computer Society.

[DFK$^+$07] Julian Dolby, Achille Fokoue, Aditya Kalyanpur, Aaron Kershenbaum, Edith Schonberg, Kavitha Srinivas, and Li Ma. Scalable semantic retrieval through summarization and refinement. In *AAAI'07: Proceedings of the 22nd national conference on Artificial intelligence*, pages 299–304. AAAI Press, 2007.

[Fra11] Franz Inc. Allegrograph. http://www.franz.com/agraph/, 2011.

[GHLS07] Birte Glimm, Ian Horrocks, Carsten Lutz, and Uli Sattler. Conjunctive Query Answering in the Description Logic SHIQ. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI 2007)*, 2007.

[GMN$^+$09] O. Gries, R. Möller, A. Nafissi, K. Sokolski, and M. Rosenfeld. CASAM Domain Ontology. Technical report, Hamburg University of Technology, 2009.

[GMN$^+$10] Oliver Gries, Ralf Möller, Anahita Nafissi, Maurice Rosenfeld, Kamil Sokolski, and Michael Wessel. A Probabilistic Abduction Engine for Media Interpretation Based on Ontologies. In Pascal Hitzler and Thomas Lukasiewicz, editors, *RR*, volume 6333 of *Lecture Notes in Computer Science*, pages 182–194. Springer, 2010.

[GPH05] Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin. LUBM: A benchmark for OWL knowledge base systems. *J. Web Sem.*, 3(2-3):158–182, 2005.

[HKP$^+$09] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph. OWL 2 Web Ontology Language Primer. W3C Recommendation, World Wide Web Consortium, October 2009.

[HMW04] V. Haarslev, R. Möller, and M. Wessel. Querying the Semantic Web with Racer + nRQL. In *Proceedings of the KI-2004 International Workshop on Applications of Description Logics (ADL'04), Ulm, Germany, September 24*, 2004.

[IF02] ISO/IEC15938-5FCD. Multimedia Content Description Interface (MPEG-7). http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm, 2002.

[Kir06] Atanas Kiryakov. OWLIM: Balancing between scalable repository and light-weight reasoner. In *Proc. of WWW2006*, Edinburgh, Scotland, 2006.

[KLW94] Ramakrishna Karedla, J. Spencer Love, and Bradley G. Wherry. Caching strategies to improve disk system performance. *Computer*, 27(3):38–46, 1994.

[Mot08] Boris Motik. KAON2 - Scalable Reasoning over Ontologies with Large Data Sets. *ERCIM News*, 2008(72), 2008.

[MW88] David Maier and David Scott Warren. *Computing with Logic: Logic Programming with Prolog.* Benjamin/Cummings, 1988.

[PTP10] Katerina Papantoniou, George Tsatsaronis, and Georgios Paliouras. KDTA: Automated Knowledge-Driven Text Annotation. In José L. Balcázar, Francesco Bonchi, Aristides Gionis, and Michèle Sebag, editors, *ECML/PKDD (3)*, volume 6323 of *Lecture Notes in Computer Science*, pages 611–614. Springer, 2010.

[PTZ09] Jeff Z. Pan, Edward Thomas, and Yuting Zhao. Completeness Guaranteed Approximations for OWL-DL Query Answering. In Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, and Ulrike Sattler, editors, *Description Logics*, volume 477 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2009.

[RPZ10] Yuan Ren, Jeff Z. Pan, and Yuting Zhao. Soundness Preserving Approximation for TBox Reasoning. In Maria Fox and David Poole, editors, *AAAI*. AAAI Press, 2010.

[SCH10] Giorgos Stoilos, Bernardo Cuenca Grau, and Ian Horrocks. How Incomplete is your Semantic Web Reasoner? In *Proc. of the 20th Nat. Conf. on Artificial Intelligence (AAAI 10)*, pages 1431–1436. AAAI Publications, 2010.

[SPG$^+$07] Evren Sirin, Bijan Parsia, Bernardo Cuenca Grau, Aditya Kalyanpur, and Yarden Katz. Pellet: A practical OWL-DL reasoner. *J. Web Sem.*, 5(2):51–53, 2007.

[TRKH08] Tuvshintur Tserendorj, Sebastian Rudolph, Markus Krötzsch, and Pascal Hitzler. Approximate OWL-reasoning with Screech. In Diego Calvanese and Georg Lausen, editors, *RR*, volume 5341 of *Lecture Notes in Computer Science*, pages 165–180. Springer, 2008.

[WA02] Michael Widenius and Davis Axmark. *MySQL Reference Manual*. O'Reilly & Associates, Inc., Sebastopol, CA, USA, 2002.

[WM08] Sebastian Wandelt and Ralf Möller. Island reasoning for ALCHI ontologies. In *Proceedings of the 2008 conference on Formal Ontology in Information Systems*, pages 164–177, Amsterdam, The Netherlands, 2008. IOS Press.