Distributed Island-based Query Answering for Semi-Expressive Ontologies

Sebastian Wandelt and Ralf Moeller

31.05.2010

Abstract. Scalability of reasoning systems is one of the main criteria which will determine the success of Semantic Web systems in the future. The focus of recent work is either on (a) systems which rely on in-memory structures or (b) not so expressive ontology languages, which can be dealt with by using database technologies.
In this paper we introduce a method to perform query answering for semi-expressive ontologies without the limit of in-memory structures. Our main idea is to compute small and characteristic representations of the assertional part of the input ontology. Query answering is then more efficiently performed over a reduced set of these small representations. We show that query answering can be distributed in a network of description logic reasoning systems to scale for reasoning. Our initial results are encouraging.

1 Introduction

As the Semantic Web evolves, scalability of inference techniques becomes increasingly important. While in recent years the focus was on pure terminological reasoning, the interest shifts now more to reasoning with respect to large assertional parts, e.g. in the order of millions or billions of triples. The first steps were done in [AF06]. The authors propose to extract a condensed summary graph out of the assertional part of an ontology, and then perform reasoning on that summary. [AF06] reports encouraging performance results. However, for avoiding inconsistencies due to merging, the summaries have to be rewritten in expensive *query-dependent* refinement steps. With increasing number of refinement steps necessary, the performance of the aproach degrades [JDS09]. Moreover, the technical criteria for summarization (creating representative nodes by grouping concept sets), seems arbitrary. In [WM08], a method is proposed to identify the relevant *islands*, i.e. set of assertions/information, required to reason about a given individual. The main motivation is to enable in-memory reasoning over ontologies with a large ABox, for traditional tableau-based reasoning systems.

Given the island of an individual, we will make the idea of summarization more formal. In this paper we present an approach to execute efficient instant retrieval tests on database-oriented ontologies. The main insight of our work is that the islands computed in [WM08] can be checked for similarity and instance retrieval can then be performed over equivalence classes of similar islands. The query answering algorithm for instance retrieval over similar islands is implemented in a distributed manner. We report interesting scalability results with respect to our test ontology: increasing the number of nodes in the network by the factor of n almost reduces the query answering time to $\frac{1}{n}$. Moreover, we implemented our algorithm in such a way that the input ontology can be loaded in a offline phase and changed afterwards incrementally online.

For the remaining part of the introduction, we will discuss a general framework to reason about description logics, and show which parts are relevant for our proposal. For description languages as expressive as, for instance, SHIQ, proposals to ensure scalability have been made in the literature as well. The goal of this is to apply approximation techniques without sacrificing the quality of the results [KMWW08]. For scalable query answering over an expressive Tbox, it has been proposed to apply a completeness-preserving approximation step (see Figure 1). A SHIQ-Tbox is transformed into a DL-Lite Tbox which is then used to provide ontology-based query answering. Query answering yields complete results (no result w.r.t. the original Tbox is missing). But there might be some elements in the result set corresponding to the approximated Tbox that would not be return if the query was answered w.r.t. the original Tbox [ZWC95] and [GST90]. Thus, we only have a set of candidates (usually not much larger than the true result set), and a subsequent filtering step is required. For this step, we exploit that the original Abox can be partitioned, i.e., segments can be found that can be processed in isolation [WM08]. The set of candidates is then filtered w.r.t. segments corresponding to the candidates. Different approaches can be used to further reduce the candidates with a scalable process. A compressor (see

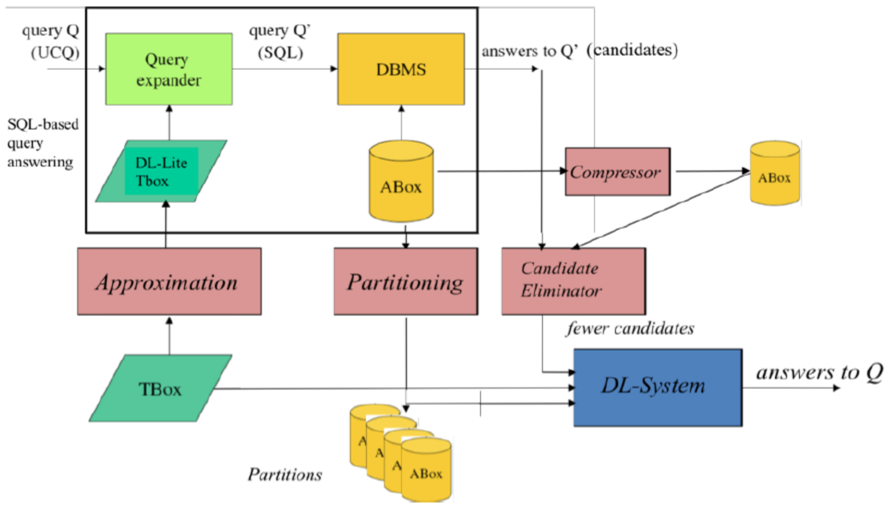

Fig. 1. Efficient query answering for expressive description logics ([KMWW08])

Figure 7) is used to produce a synopsis, which is then used to eliminate obvious elements of the candidate set which must not be in the result set (for details see [WM07]). A new research goal is to test the combination of approximation and segment-wise treatment of data description in a large-scale industrial scenario using the results of [HM08]. Our approach is situated in the modules *Partitioning* and *Candidate Eliminator* (Figure 1). Furthermore, it can be used as a stand-alone reasoning system, under the assumption that the ontology yields many similar islands (in case of database-like assertional data).

The remaining parts of the paper are structured as follows. Section 2 introduces necessary formal notions and gives an overview over Related Work. In Section 3 we adopt the underlying island computation process from [WM08]. The main theoretical contribution of our work is in Section 4, the isomorphism criteria for islands. We show our implementation in Section 5 and provide initial evaluation results in Section 6. The paper is concluded in Section 7.

2 Preliminaries

For details about syntax and semantics of the description logic $\mathcal{ALCHI}$ we refer to [BCM$^+$07]. Some definitions are appropriate to explain our nomenclature, however. 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 $N_{RN} \cup \{R^- | R \in N_{RN}\}$. We say that a concept description is *atomic*, if it is a concept name or its negation. With $\mathcal{S}_{AC}$ we denote all atomic concepts.

Furthermore we assume the notions of TBoxes ($\mathcal{T}$), RBoxes ($\mathcal{R}$) and ABoxes ($\mathcal{A}$) as in [BCM$^+$07]. A *ontology* $\mathcal{O}$ consists of a 3-tuple $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, where $\mathcal{T}$ is a

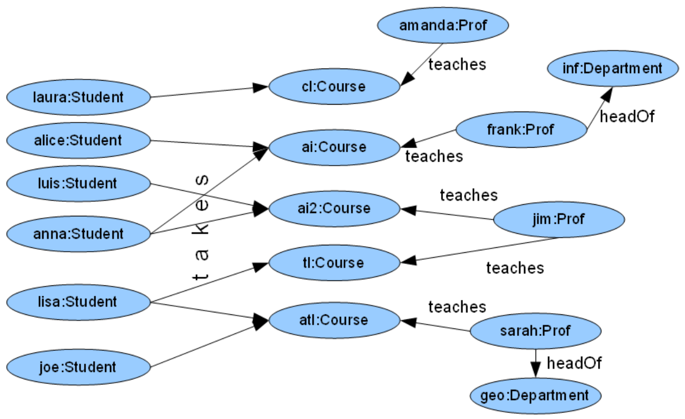

Fig. 2. Example ABox $\mathcal{A}_{EX1}$

TBox, $\mathcal{R}$ is a RBox and $\mathcal{A}$ is a ABox. We restrict the concept assertions in $\mathcal{A}$ to only use atomic concepts. This is a common assumption, e.g. in [GH06], when dealing with large assertional datasets stemming from databases. With $Ind(\mathcal{A})$ we denote the set of individuals occurring in $\mathcal{A}$. Throughout the remaining part of the paper we assume the Unique Name Assumption (UNA), i.e. two distinct individual names denote distinct domain objects.

In Example 1 we define an example ontology, used throughout the remaining part of the paper to explain definitions. The example ontology is setting of universities. We evaluate our ideas w.r.t. to "full" LUBM [GPH05] in Section 6. Although this is a synthetic benchmark, several (if not most) papers on scalability of ontological reasoning consider it as a base reference.

Example 1. Let $\mathcal{O}_{EX1} = \langle \mathcal{T}_{EX1}, \mathcal{R}_{EX1}, \mathcal{A}_{EX1} \rangle$, s.t.

$$\begin{aligned}
\mathcal{T}_{EX1} =& \{Chair \equiv \exists headOf.Department \sqcap Person, Prof \sqsubseteq Person,\\
& GraduateCourseTeacher \equiv Prof \sqcap \exists teaches.GraduateCourse\}\\
\mathcal{R}_{EX1} =& \{headOf \sqsubseteq worksFor\}\\
\mathcal{A}_{EX1} =& \texttt{see Figure 2}
\end{aligned}$$

Next we discuss related work relevant to our contribution. In [SP08], the authors discuss a general approach to partition OWL knowledge bases and distribute reasoning over partitions to different processors/nodes. The idea is that the input for their partitioning algorithm is a fixed number of desired partitions, which will be calculated by different means (weighted graphs, hash-based distribution or domain specific partitions). The partitions are not independent from each other. Moreover, in some cases, the data is just arbitrarily spread over the different nodes in the networks. This leads to a noticeable amount of communication overhead between the nodes, because partial results have to be passed in between the nodes. The authors discuss rather small data sets,

e.g. 1 million triples. These problems can already be solved with state-of-the-art tableau-based reasoning systems. Furthermore, their evaluation only talks about speed-up, without mentioning the actual run-time, or referring to some open/state-of-the art implementation.
The work in [UKOvH09] proposes a MapReduce [DG04]-based technique, to compute the closure (set of all implications) over ontologies. Given the underlying MapReduce framework, their approach could scale in theory. The major difference to our work is that we focus on query answering, instead of brute force (bottom-up) generation of all possible implications of a given knowledge base. Moreover we focus on more expressive description logics and it is at least doubtful, whether their approach will work for non-deterministic logics (e.g. allowing for disjunctions).
The authors of [BS03] discuss an approach to integrate ontologies from different sources in a distributed setting. They introduce so-called bridge-rules to identify, which parts of the ontologies overlap (and thus need to be communicated between the different reasoning nodes). The main focus of their work is rather on the integration of distributed ontologies, but not on scalable reasoning over large ontologies in general.
There is additional work on distributed Datalog implementations (e.g. [ZWC95] and [GST90]) and on non-distributed reasoning optimizations/techniques for description logics, e.g. [GH06].

3 Island calculation

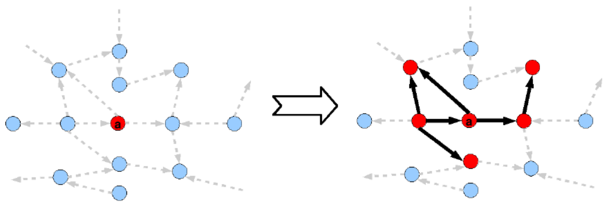

Fig. 3. Example: Island relevant for reasoning about individual a

In [WM08], a method is proposed to identify the relevant information (assertions) to reason about a given individuals. 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. The general idea is visualized in Figure 3. More formally, given an input individual a, the proposal is to compute a set of ABox assertions $\mathcal{A}_{isl}$ (a subset of the source ABox $\mathcal{A}$), such that $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle \vDash C(a)$ iff $\langle \mathcal{T}, \mathcal{R}, \mathcal{A}_{isl} \rangle \vDash C(a)$.

Figure 4 shows the general algorithm to compute the island of an individual, $getIsland(a, \emptyset)$. Please note that we generalized the algorithm from [WM08] by hiding the criteria for checking role (in-)separability vie $\forall$-info structures, since the particular choice of strategy does not have an impact on our simulation algorithm. Thus, we make use of a general function $insep_{\mathcal{O}}$, which determines whether a role assertion $R(A, b)$ is inseparable with respect to ontology $\mathcal{O}$. In the following we will use $ISLAND(a)$ to refer to the island (set of assertions "relevant" for reasoning) computed for individual a.

```
Function $getIsland(a, seen)$
Parameter: Individual $a$, list of visited individuals $seen$
Returns: Set $S$ of relevant ABox assertions
Algorithm:
    1. If $a \in seen$ then Return $\emptyset$
    2. $seen = seen \cup \{a\}$
    3. $S = \{a : X \in \mathcal{A}\}$
    4. For $R(a, b) \in \mathcal{A}$ do
        (a) If $insep_{\mathcal{O}}(R(a, b))$ then
                $S = S \cup \{R(a, b)\} \cup build(b, seen)$
    5. For $R(b, a) \in \mathcal{A}$ do
        (a) If $insep_{\mathcal{O}}(R(b, a))$ then
                $S = S \cup \{R(b, a)\} \cup build(b, seen)$
    6. Return $S$
```

Fig. 4. Build island for individual a

The goal of the approach [WM08] is that it enables to use tableau-based in-memory reasoning systems to be run on ontologies, which are too large to fit into main memory. But if you want to preform more complex reasoning than instance checking on ontologies, e.g. answering conjunctive queries, the optimization is not yet enough. To make this more clear, we look on how traditional description logic reasoning systems answer instance retrieval queries based on instance checks.

Given an instance retrieval task for concept C with respect to ontology $\mathcal{O}$ $=\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, a naive reasoner will iterate all individuals $a \in Ind(\mathcal{A})$ of the input ABox and check, whether $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle \models C(a)$. If yes, then a is included in the result set for the instance retrieval query. The performance of instance retrieval queries in [WM08] depends highly on the number of individuals in the ABox. For 100 universities, we have around 300.000 individuals, i.e. 300.000 islands. When we assume that one instance retrieval check takes around 1 ms, we would need already 1 hour to answer one instance retrieval query on a single machine. If we want to improve query answering times, one could parallelize these islands checks. For example, one would need to have 3600 machines at his disposal, to obtain all the answers within one second. If the average instance checking time is higher, or the number of individuals is larger (= more universities), then these statistics become even worse. Thus, our motivation to further improve instance retrieval time for database-oriented ontologies.

4 Similarity of Islands

In the following, we discuss how islands can be used for optimized instance retrieval tests and answering conjunctive queries. The main insight is that many of the computed islands are similar to each other. Especially in database-oriented scenarios, ontologies contain a lot of individuals, which follow patterns defined by a schema (i.e. the terminology of the ontology). If it is possible to define a formal notion of similarity for islands, and show that it is sufficient to perform reasoning over one representative island, instead of all these similar islands, then query answering can potentially be increased by several orders of magnitude (depending on the number dissimilar island classes). We consider an example to clarify the idea of island isomorphisms more clear.

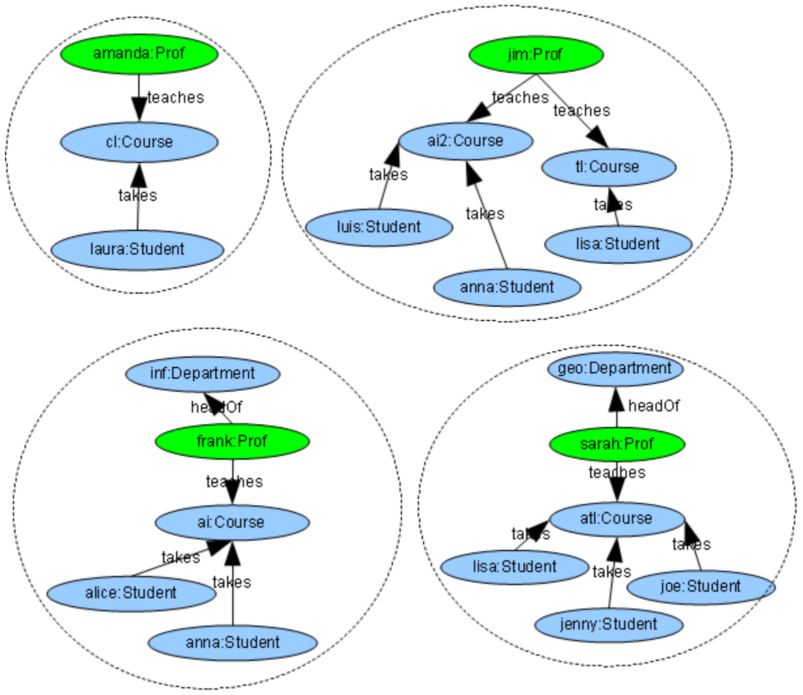

Fig. 5. Example: Islands of the four Professors in $\mathcal{O}_{EX1}$

In Figure 5 we show the extracted islands of all professors in our example ontology $\mathcal{O}_{EX1}$. While all four graphs are different, they have some similarities in common, which can be used to optimize reasoning over these islands. To define similarities over islands, we introduce formally the notion of an island and define a similarity criterion.

Definition 1. *A* individual-island-graph *IIG is a tuple $\langle N, \phi_n, \phi_e, root\rangle$, such that*

- *N is a set of nodes,*
- *$\phi_n : N \rightarrow 2^{\mathcal{S}_{AC}}$ is a node-labeling function ($\mathcal{S}_{AC}$ is the set of atomic concepts),*
- *$\phi_e : N \times N \rightarrow 2^{L_e}$ is a edge-labeling function*
- *$root \in N$ is a distinguished root node.*

If we have $\phi_e(a, b) = \rho$ and $\rho \neq \emptyset$, then we write $a \xrightarrow{\rho}_{IIG} b$. The definition of individual-island-graphs is quite straight-forward. In the following we define a similarity relation over two individual-island-graphs, based on graph bisimulations. Although the term *bisimulation* is usually used in process algebra to define similar processes, we use it here in the context of graphs.

Definition 2. *A* bisimulation *over $IIG_1 = \langle N_{IIG_1}, \phi_{nIIG_1}, \phi_{eIIG_1}, root_{IIG_1}\rangle$ and $IIG_2 = \langle N_{IIG_2}, \phi_{nIIG_2}, \phi_{eIIG_2}, root_{IIG_2}\rangle$ is a binary relation $R_{IIG_1,IIG_2} \subseteq N \times N$, such that*

- *$R_{IIG_1,IIG_2}(root_{IIG_1}, root_{IIG_2})$*
- *if $R_{IIG_1,IIG_2}(a, b)$ then $\phi_{nIIG_1}(a) = \phi_{nIIG_2}(b)$*
- *if $R_{IIG_1,IIG_2}(a, b)$ and $a \xrightarrow{\rho}_{IIG_1} a$‘ then there exists a b‘ $\in N_{IIG_2}$ with $b \xrightarrow{\rho}_{IIG_2} b$‘ and $R_{IIG_1,IIG_2}(a‘, b‘)$*
- *if $R_{IIG_1,IIG_2}(a, b)$ and $b \rightarrow^{\rho}_{IIG_2} b$‘ then there exists a a‘ $\in N_{IIG_1}$ with $a \xrightarrow{\rho}_{IIG_2} a$‘ and $R_{IIG_1,IIG_2}(a‘, b‘)$*

Definition 3. *Two individual-island-graphs IIG_1 and IIG_2 are called* bisimilar*, if there exists a bisimilation R for them.*

Example 2. To make these definitions more clear, we show individual-island-graphs for $amanda$, jim and $frank$, plus a possible bisimulation between $amanda$ and jim:

- $IIG_{amanda} = \langle N_{amanda}, \phi_{n_{amanda}}, \phi_{e_{amanda}}, root_{amanda}\rangle$, s.t.

$$\begin{aligned}
N_{amanda} =& \{x_{amanda}, x_{cl}, x_{laura}\}\\
\phi_{n_{amanda}} =& \{x_{amanda} \rightarrow \{Prof\}, x_{cl} \rightarrow \{Course\}, x_{laura} \rightarrow \{Student\}\}\\
\phi_{e_{amanda}} =& \{(x_{amanda}, x_{cl}) \rightarrow \{teaches\}, (x_{laura}, x_{cl}) \rightarrow \{takes\}\}\\
root_{amanda} =& \{x_{amanda}\}
\end{aligned}$$

- $IIG_{jim} = \langle N_{jim}, \phi_{n_{jim}}, \phi_{e_{jim}}, root_{jim}\rangle$, s.t.

$$\begin{aligned}
N_{jim} =& \{y_{jim}, y_{ai2}, y_{tl}, y_{luis}, y_{anna}, y_{lisa}\}\\
\phi_{n_{jim}} =& \{y_{jim} \rightarrow \{Prof\}, y_{ai2} \rightarrow \{Course\}, y_{tl} \rightarrow \{Course\}, y_{luis} \rightarrow \{Student\}, ...\}\\
\phi_{e_{jim}} =& \{(y_{jim}, y_{ai2}) \rightarrow \{teaches\}, (y_{jim}, y_{tl}) \rightarrow \{teaches\}, (y_{luis}, x_{ai2}) \rightarrow \{takes\}, ...\}\\
root_{jim} =& \{y_{jim}\}
\end{aligned}$$

– $IIG_{frank} = \langle N_{frank}, \phi_{n\,frank}, \phi_{e\,frank}, root_{frank}\rangle$, s.t.

$$\begin{aligned}N_{frank} =& \{z_{frank}, z_{ai}, z_{inf}, z_{alice}, z_{anna}\}\\ \phi_{n\,frank} =& \{z_{frank} \to \{Prof\}, z_{ai} \to \{Course\}, z_{inf} \to \{Department\},\\ & z_{alice} \to \{Student\}, z_{anna} \to \{Student\}\}\\ \phi_{e\,frank} =& \{(z_{frank}, z_{ai}) \to \{teaches\}, (z_{frank}, z_{inf}) \to \{headOf\},\\ & (z_{alice}, z_{ai}) \to \{takes\}, (z_{anna}, z_{ai}) \to \{takes\}\}\\ root_{jim} =& \{z_{frank}\}\end{aligned}$$

– $R_{jim,amanda} =$

$$\{(x_{amanda}, y_{jim}), (x_{cl}, y_{ai2}), (x_{cl}, y_{tl}), (x_{laura}, y_{luis}),\\ (x_{laura}, y_{luis}), (x_{anna}, y_{lisa})\}$$

It is easy to see, that $R_{jim,amanda}$ is a bisimulation for the islands (graphs) of the individuals jim and $amanda$. Furthermore, it is easy to see that there cannot be a bisimulation, for instance, between jim and $frank$.

The important insight is that bisimilar islands entail the same concept sets for their root individual, if the underlying description logic is restricted to $ALCHI$. This is shown in the following theorem.

Theorem 1. *Given two individuals a and b, we have $\langle \mathcal{T}, \mathcal{R}, ISLAND(a)\rangle \vDash C(a) \iff \langle \mathcal{T}, \mathcal{R}, ISLAND(b)\rangle \vDash C(b)$, if we can find a bisimulation $R_{a,b}$, for $ISLAND(a)$ and $ISLAND(b)$.*

Proof. We assume that we have a bisimulation $R_{a,b}$ for the islands $ISLAND(a)$ and $ISLAND(b)$. Now we have to prove two directions:

1. $\langle \mathcal{T}, \mathcal{R}, ISLAND(a)\rangle \vDash C(a) \implies \langle \mathcal{T}, \mathcal{R}, ISLAND(b)\rangle \vDash C(b)$:
   Sketch (proof by contradiction):
   Assume that we have $\langle \mathcal{T}, \mathcal{R}, ISLAND(a)\rangle \vDash C(a)$, and therefore $\mathcal{O}_a = \langle \mathcal{T}, \mathcal{R}, ISLAND(a) \cup \{\neg C(a)\}\rangle$ can be shown to be inconsistent by a tableau-proof. That means that all tableau-runs are closed (lead to a clash). By $\langle \mathcal{T}, \mathcal{R}, ISLAND(b)\rangle \nvDash C(b)$, we know that $\mathcal{O}_2 = \langle \mathcal{T}, \mathcal{R}, ISLAND(b) \cup \{\neg C(b)\}\rangle$ can be shown to be consistent by a tableau-proof, i.e. there exists an open tableau run TR_b on $\mathcal{O}_b$. Now it is easy to show that R_T can be transformed into a open tableau-run TR_a on $\mathcal{O}_a$, by applying tableau rules always to the corresponding nodes of the bisimulation $R_{a,b}$. The only slightly difficult part is the $\exists$-rule for the $\mathcal{ALCHI}$-tableau algorithm, because one has to update the bisimulation relation on the newly created successor nodes. Since we can find an open tableau-run TR_a, the initial assumption (all tableau-runs are closed) is shown to be contradicting.
2. $\langle \mathcal{T}, \mathcal{R}, ISLAND(b)\rangle \vDash C(b) \implies \langle \mathcal{T}, \mathcal{R}, ISLAND(a)\rangle \vDash C(a)$ Analogous to the first case.

The above theorem can be easily lifted to the case of more than two individuals, i.e. if we have n individuals, and for all of their islands one can find a bisimilarity relation, it is sufficient to perform instance checking on one island. In practice, especially in database-oriented ontologies, this can dramatically speed up the time for instance retrieval. To show this, we need to further introduce some mathematical notions.

Definition 4. *A* individual-island-equivalence $\sim_{ISL}$ *is an equivalence relation over individual islands, such that we have* $\sim_{ISL}$ (ISL, ISL)*, if we can find a bisimulation* $R_{ISL,ISL}$ *between the two islands* ISL *and* ISL*. With* $[\sim_{ISL}]$ *we denote the set of equivalence classes of* $\sim_{ISL}$*.*

The main theoretical result of our work is summarized in the following theorem.

Theorem 2. *Given an ontology* $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$*, one can perform grounded instance retrieval for the atomic concept* C *over* $[\sim_{ISL}]$*, instead of all islands.*

Proof. Follows immediately from the above Theorem and the fact from [WM08], that instance retrieval can be performed over all islands.

Please note that our approach does not work directly for more expressive description logics, e.g. $\mathcal{SHOIQ}$. In the presence of cardinality restrictions we will need more sophisticated bisimulation criteria to identify similar nodes, since the number of related similar individuals matters. Nominals further complicate the bisimulation criteria, since individuals can be forced by the terminological axioms to refer to the same domain object, i.e. one might need to calculate all TBox-implications in the worst calse.

5 Distributed Implementation

We have implemented our proposal for Island Simulations in Java. For ontology access we use the OWLAPI 2.2.0[BVL03].

The general structure of our implementation is shown in Figure 6. In the following we give a short description of each component and especially provide performance optimization insights we gained during the implementation of the prototype.

– (Server) OWL-Converter: Our tests with OWLAPI showed that, if we use OWLAPI directly to load ontologies, there are some memory leaks, which will degrade performance over time. Thus we decided to convert ontologies from OWL to an internal representation in a preprocessing step.
– (Server) Update Handler: This module is responsible update the internal structures (representation of islands) in case of an ontology update. Depending on the kind of update - terminological or assertional - the module determine which islands, i.e. which individuals, have to be recomputed.
– (Server) Island Computor: This module computes the island for a given an individual. It uses the current state of the TBox and ABox to obtain the currently valid island.

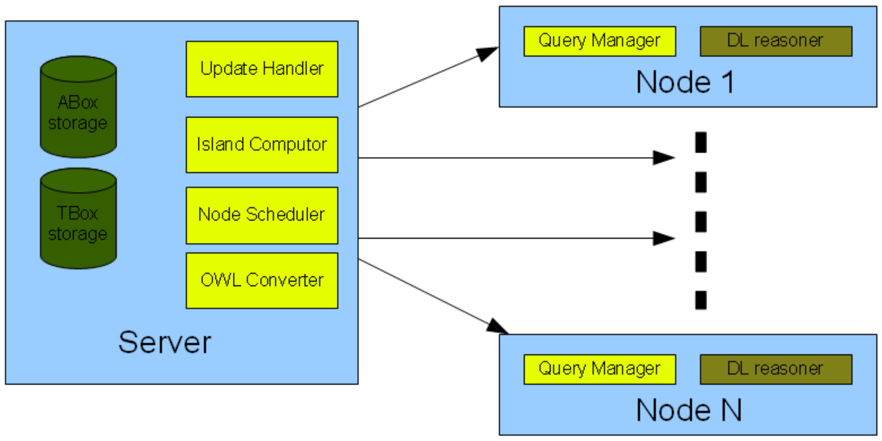

Fig. 6. Structure of our prototypical implementation

- *(Server) Node Scheduler*: Whenever a new island is found, the task of the Node Scheduler is to determine the responsible node for the island. At first, we implemented a simple Round-Robin-Strategy, i.e. the number of islands on every node would be (roughly) the same. But since we ran some of our tests in a heterogeneous network, it turned out, that the server always had to wait for the slowest node. Thus, we revised the Node scheduling as follows:
  1. Once a node connects to the server, we perform some basic inference tests with a standardized ontology and measure the time needed to report the results to the server.
  2. During node population phase, i.e. for loading the actual ontology, we distribute the islands corresponding to the time taken for the initial tests (faster nodes obtain more islands).

  This measure greatly improved the overall performance of query answering in the (heterogeneous) network, since almost all nodes finish their complete task at roughly the same time now.
- (Server) TBox/ABox Storage: This part of the server is responsible for storing the current state of the terminological/assertional part of the ontology.
- (Client) Query Manager: Given an incoming query, this module determines all active islands and uses the DL Reasoner module to find out which islands match the input query.
- (Client) DL Reasoner: A module implementing an interface to a general description logic reasoner (in our case we used Racer [HM01]). The main task is to load an island and perform instance checking or individual classification respectively.

One more remark on our implementation: while loading an ontology we built a dependency tree, which stores which impact updates on particular islands have, e.g. we store that, adding a *teaches*-relation to a particular individual (island), then we obtain another fixed island. This kind of lookup-table greatly improved the performance in our tests, because we do not have to recompute complete islands in case we had "'similar"' updates before.

6 Evaluation

Our tests were run with respect to the synthetic benchmark ontology LUBM [GPH05]. Although some people claim that LUBM does not fully represent all the capabilities provided by the complete OWL specification, we think that it fits our constraint of database-oriented ontologies: rather small and simple TBox, but a bulk of assertional information with a realistic distribution with respect to numbers of professors, students, departments, etc. In our evaluation we compare three different measures, to determine the performance of our implementation:

- *Load time*: In Figure 7 we show the size of the assertional part in triples and compare the load time with different number of nodes in our network (1, 2 and 4 nodes). The load time only represents the time spent to traverse the input ontology one time and compute that bisimilarity relation over islands of all individuals. It can be seen that the load time increases linearly with the number of triples in the assertional part. Please note that our loading algorithm is designed and implemented as an incremental algorithm. Thus, if we add a new assertion, we do not have to recompute all the internal structures, but only update the relevant structures.
  Furthermore note that the load time is independent from the number of nodes. This is because we focused on improving query answering times, not load times, and thus major computations for loading are performed on the server (island computation and bisimulation tests). It is possible to distribute the loading process over nodes as well, but then one has to deal with the problem of a distributed ABox. In our tests the communication overhead for a distributed ABox implementation was usually too high. Thus we leave the further improvement of load times for future work.
- *Preparation time*: This measure indicates an initial preparation time after the ontology is fully loaded. The time is spent to prepare the internal structures of the DL reasoner for incoming queries. Please note that this preprocessing step is query independent and only performed one time after the ontology was updated. The idea is that we can perform incremental bulk loading (measured in *load time*), without updating the (expensive) internal structures of the DL reasoner all the time.
  In the left part of Figure 8, we show the query preparation time for different numbers of universities and different numbers of nodes in the network. The number of nodes indeed affects the query preparation time. If we use 8 nodes, the preparation time is almost $\frac{1}{8}$ of the time needed for one node. Thus, the distribution of computational power works for query preparation.
  In the right part of Figure 8 we compare the necessary number of islands to perform instance retrieval with the original work in [WM08]. It can be

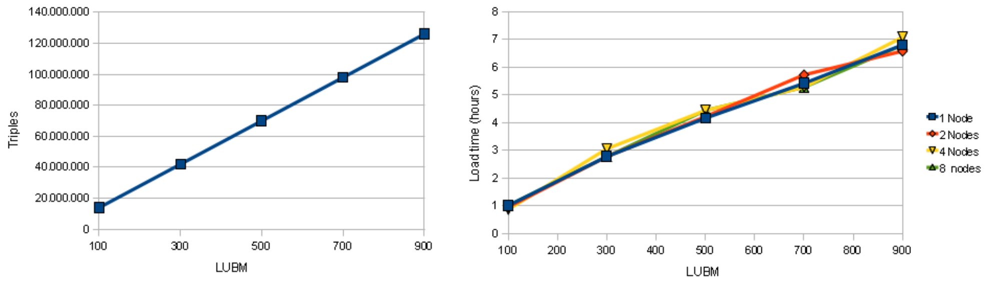

Fig. 7. Input size and load time

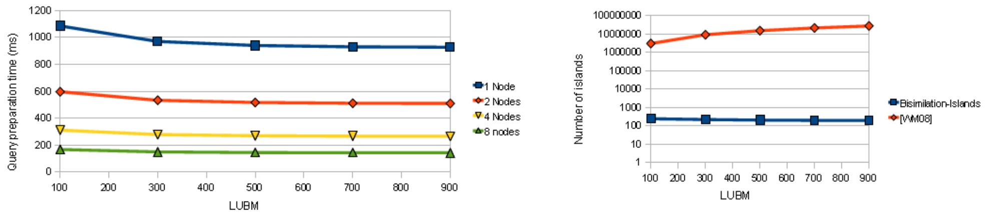

Fig. 8. Query preparation time and island count

seen, that the number of islands increases linearly with the size of the input ontology for [WM08] (please note the logarithmic scale). Using bisimulation, the number of islands is almost constant for all input ontologies, since most of the newly introduced individual-islands are bisimilar to each other, e.g. professors who teach particular students in particular kinds of courses.

– *Query answering time*: The third measure indicates how long the actual query answering takes. In Figure 9, the query answering time (for instance retrieval) for the concepts *Chair* (small number of answers, linearly growing with the number of universities) are shown. Please note that query answering times are rather independent from the chosen concept description for instance retrieval. We only focus on *Chair*, since it is also commonly used in the literature to perform benchmarks on LUBM.

In Figure 9 (upper part) we show the time needed to identify the islands which entail the concept *Chair*. This is the actual "'description-logic"'-hard task. To return the answers one has to further lookup the names of all individuals for the matching islands, e.g. in a hashmap. The actual query answering time (including lookup) for one node is shown in the lower part of Figure 9. It can be seen that our bisimulation-based approach can improve query answering times in the order of several magnitudes in the case of instance retrieval.

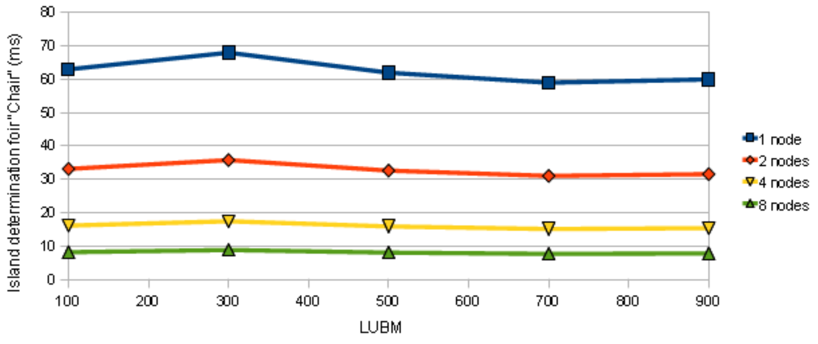

Fig. 9. Query answering time

Please note that the compact representation of solutions for instance retrieval queries (sets of solution islands), can have a high potential to improve answering tiems for conjunctive queries, since the system does not have to deal with large sets of solution individuals. Further investigations about this potential are part of future work.

7 Conclusions

We have proposed a method for instance retrieval over ontologies in a distributed system of DL reasoners. To the best of our knowledge, we are the first to propose instance retrieval reasoning based on similarity of individual-islands. The results are encouraging so far. We emphasize that our approach especially works for ontologies with a rather simple or average size terminological part. For future work, it will be important to investigate more ontologies and check the performance of our proposal. Furthermore, we want to extend our proposal to more expressive description logics, e.g. SHIQ or even SHOIQ.

References

[AF06] Li Ma E. Schonberg K. Srinivas A. Fokoue, A. Kershenbaum. The Summary ABox: Cutting ontologies down to size. In *SSWS 2006*, pages 61–74, Athens, GA, USA, November 2006.

[BCM$^+$07] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. *The Description Logic Handbook*. Cambridge University Press, New York, NY, USA, 2007.

[BS03] Alex Borgida and Luciano Serafini. Distributed description logics: Assimilating information from peer sources. *J. of Data Semantics*, 2003.

[BVL03] S. Bechhofer, R. Volz, and P. Lord. Cooking the Semantic Web with the OWL API, 2003.

[DG04] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI 2004*, pages 137–150, 2004.

[GH06] Yuanbo Guo and Jeff Heflin. A Scalable Approach for Partitioning OWL Knowledge Bases. In *SSWS 2006*, Athens, GA, USA, November 2006.

[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.

[GST90] Sumit Ganguly, Avi Silberschatz, and Shalom Tsur. A framework for the parallel processing of datalog queries. *SIGMOD Rec.*, 19(2):143–152, 1990.

[HM01] V. Haarslev and R. Möller. Description of the RACER System and its Applications. In *Proceedings International Workshop on Description Logics (DL-2001), Stanford, USA, 1.-3. August*, pages 131–141, 2001.

[HM08] V. Haarslev and R. Möller. On the scalability of description logic instance retrieval. *Journal of Automated Reasoning*, 41(2):99–142, 2008.

[JDS09] Aditya Kalyanpur Edith Schonberg Julian Dolby, Achille Fokoue and Kavitha Srinivas. Efficient reasoning on large SHIN Aboxes in relational databases. In *SSWS 2009*, 2009.

[KMWW08] Alissa Kaplunova, Ralf Möller, Sebastian Wandelt, and Michael Wessel. Approximation and ABox Segmentation. Technical report, Institute for Software Systems (STS), Hamburg University of Technology, Germany, 2008. See http://www.sts.tu-harburg.de/tech-reports/papers.html.

[SP08] R. Soma and V.K. Prasanna. Parallel inferencing for OWL knowledge bases. In *Parallel Processing, 2008. ICPP '08. 37th International Conference on*, pages 75–82, Sept. 2008.

[UKOvH09] Jacopo Urbani, Spyros Kotoulas, Eyal Oren, and Frank van Harmelen. Scalable distributed reasoning using MapReduce. In *8th International Semantic Web Conference (ISWC2009)*, October 2009.

[WM07] S. Wandelt and R. Möller. Scalability of OWL Reasoning: Role condensates. In *Proc. International Workshop on Scalable Semantic Web Systems*, 2007.

[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, The Netherlands, 2008. IOS Press.

[ZWC95] Weining Zhang, Ke Wang, and Siu-Cheung Chau. Data partition and parallel evaluation of datalog programs. *IEEE Transactions on Knowledge and Data Engineering*, 7(1):163–176, 1995.