

Distributed Island-based Query Answering for Expressive Ontologies

Sebastian Wandelt and Ralf Moeller

Hamburg University of Technology, Hamburg, Germany,
wandelt@tuhh.de, r.f.moeller@tuhh.de

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) expressive description logic 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 in order to support scalable 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. Research on ontologies with medium-sized assertional information has already been conducted on less expressive description logics, e.g. in [CDGL⁺05]. Further techniques were investigated in [FKM⁺06]. The authors propose to extract a condensed summary graph out of the assertional part of an ontology, and then perform reasoning on that summary. [FKM⁺06] 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 numbers of refinement steps necessary, the performance of the approach degrades [DFK⁺09]. 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 instance

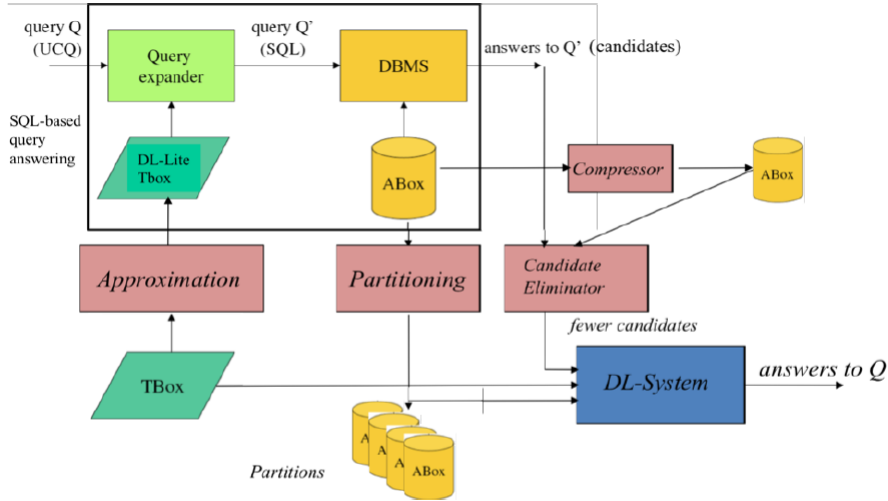


Fig. 1. Efficient query answering for expressive description logics ([KMWW08])

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 an offline phase and changed afterwards incrementally online.

Figure 1 is taken from [KMWW08] and shows the general structure of an optimized query answering system for expressive description logics. Let us assume that a query Q is to be answered w.r.t. a TBox (lower-left corner). The TBox is approximated into a DL-Lite TBox (a complete but unsound approximation is used). Exploiting DL-Lite query answering by transforming queries w.r.t. ontologies (Tboxes) into SQL queries (see the rectangle in the left-upper corner) [CDGL⁺05], candidates are generated for the query w.r.t. the original TBox. Afterwards, candidates can possibly be eliminated in beforehand. The remaining candidates must be investigated using a reasoner for the expressive TBox. In order to be able to do this, huge ABoxes are partitioned (as explained above). The approach described in this paper is situated in the modules *Partitioning* and *Candidate Eliminator*.

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.

There is an extended version of this paper available with proofs and further comments on the implementation and evaluation [WM10].

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 $N_{RN} \cup \{R^- \mid R \in N_{RN}\}$. 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]. An *ontology* \mathcal{O} is a tuple $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, where \mathcal{T} is a 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 in the setting of universities. We evaluate our ideas w.r.t. to “full” LUBM [GPH05] (in fact LUBM without the transitive role *subOrganizationOf*, because we handle \mathcal{ALCHI} 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.

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

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 can 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

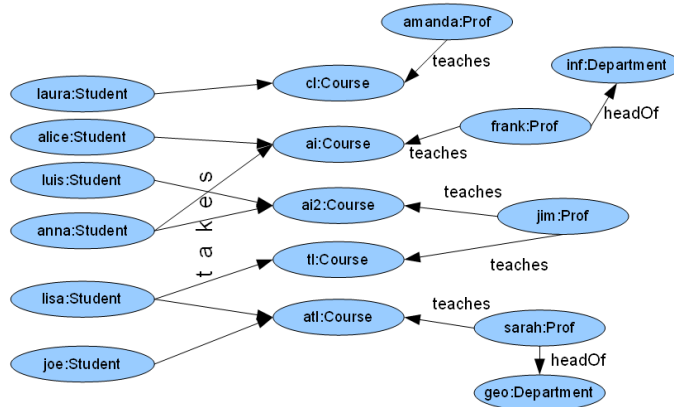


Fig. 2. Example ABox \mathcal{A}_{EX1}

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 technique based on MapReduce [DG04] to compute the closure (set of all implications) over ontologies (in the spirit of Abox realization). 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 expressive logics, i.e., logics allowing for disjunctions or the specification of ontologies which have only infinite models.

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 query answering over large ontologies. There is additional work on distributed Datalog implementations (see, e.g., [ZWC95] and [GST90]) and on non-distributed reasoning optimization techniques for description logics [GH06].

3 Island calculation

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. 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 for all concept descriptions C , we have $\langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle \models C(a)$ iff $\langle \mathcal{T}, \mathcal{R}, \mathcal{A}_{isl} \rangle \models C(a)$. We call these sets of assertions *islands*. Despite the fact that query answering is, in order to support more complex reasoning tasks, e.g., answering conjunctive queries, island computation as described in [WM08] is not enough. Given an instance retrieval task for concept C with respect to ontology $\mathcal{O} = \langle \mathcal{T}, \mathcal{R}, \mathcal{A} \rangle$, a naive approach will iterate over all individuals $a \in Ind(\mathcal{A})$ of the input ABox in order to determine 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. If we assumed 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 one intended to improve query answering times using a MapReduce approach, one could parallelize the island checks. For example, in the best case, one would need to have 3600 machines at ones disposal to obtain all answers within one second. If the average instance checking time is higher, or the number of individuals is larger (= more universities), then the situation becomes even worse. Thus, our motivation is to further improve instance retrieval time, while still supporting expressive logics.

4 Similarity of Islands

In the following, we discuss how islands can be used for optimized instance retrieval tests. 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 following patterns defined by a schema (the terminology of the ontology). If it is possible to define a formal notion of similarity for islands, and to show that it is sufficient to perform reasoning over one representative island instead, then query answering can potentially be increased by several orders of magnitude (depending on the number of dissimilar island classes). We consider an example to demonstrate the idea of island similarities.

In Figure 3 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, and this can be exploited to optimize reasoning over these islands. To define similarities over islands, we formally introduce the notion of an island and define the 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),

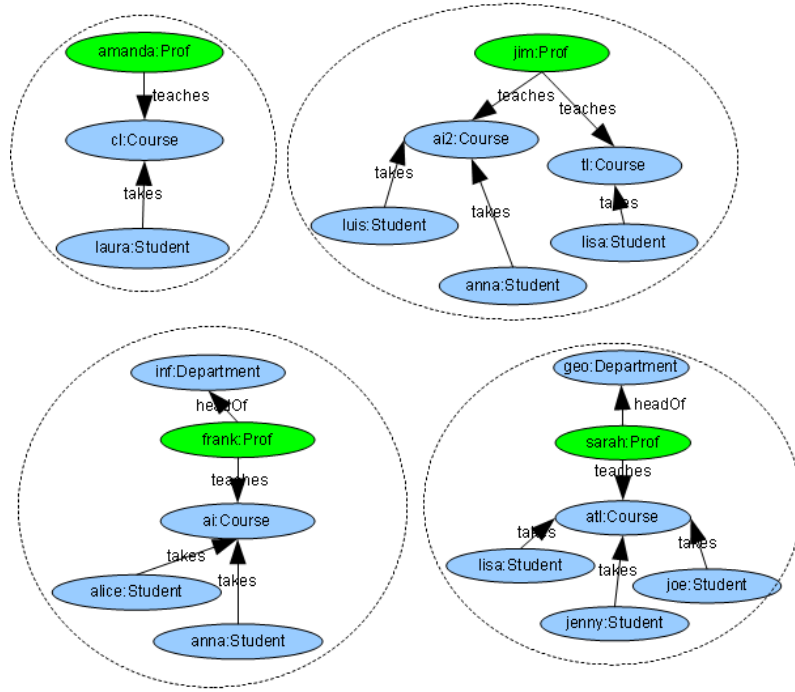


Fig. 3. Example: Islands of the four Professors in \mathcal{O}_{EX1}

- $\phi_e : N \times N \rightarrow 2^{L_e}$ 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 $IIG_1 = \langle N_{IIG_1}, \phi_{n_{IIG_1}}, \phi_{e_{IIG_1}}, root_{IIG_1} \rangle$ and $IIG_2 = \langle N_{IIG_2}, \phi_{n_{IIG_2}}, \phi_{e_{IIG_2}}, root_{IIG_2} \rangle$ is a binary relation $R_{IIG_1, IIG_2} \subseteq N_{IIG_1} \times N_{IIG_2}$, such that

- $R_{IIG_1, IIG_2}(root_{IIG_1}, root_{IIG_2})$
- if $R_{IIG_1, IIG_2}(a, b)$ then $\phi_{n_{IIG_1}}(a) = \phi_{n_{IIG_2}}(b)$
- 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 $b \xrightarrow{\rho}_{IIG_2} b'$ then there exists a $a' \in N_{IIG_1}$ with $a \xrightarrow{\rho}_{IIG_1} a'$ 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 bisimulation R for them.

Example 2. To illustrate these definitions we show individual-island-graphs for *amanda*, *jim*, and *frank*, together with a possible bisimulation between *amanda* and *jim*:

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

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

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

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

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

$$\begin{aligned} N_{frank} &= \{z_{frank}, z_{ai}, z_{inf}, z_{alice}, z_{anna}\} \\ \phi_{n_{frank}} &= \{z_{frank} \rightarrow \{Prof\}, z_{ai} \rightarrow \{Course\}, z_{inf} \rightarrow \{Department\}, \\ &\quad z_{alice} \rightarrow \{Student\}, z_{anna} \rightarrow \{Student\}\} \\ \phi_{e_{frank}} &= \{(z_{frank}, z_{ai}) \rightarrow \{teaches\}, (z_{frank}, z_{inf}) \rightarrow \{headOf\}, \\ &\quad (z_{alice}, z_{ai}) \rightarrow \{takes\}, (z_{anna}, z_{ai}) \rightarrow \{takes\}\} \\ root_{frank} &= \{z_{frank}\} \end{aligned}$$

$$- R_{jim, amanda} =$$

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

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 and any concept description C , it holds that $\langle \mathcal{T}, \mathcal{R}, ISLAND(a) \rangle \models C(a) \iff \langle \mathcal{T}, \mathcal{R}, ISLAND(b) \rangle \models C(b)$ if there exists a bisimulation $R_{a,b}$, for $ISLAND(a)$ and $ISLAND(b)$.

For the proof see [WM10]. 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. *An individual-island-equivalence \sim_{ISL} is an equivalence relation over individual islands, such that we have $\sim_{ISL}(ISL_1, ISL_2)$ if we can find a bisimulation R_{ISL_1, ISL_2} between the two islands ISL_1 and ISL_2 . 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 $[\sim_{ISL}]$.*

For details see [WM10]. Please note that our approach does not work directly for more expressive description logics, e.g. *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 can 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 compute all TBox consequences in the worst case.

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, a description of each component, and additional performance optimization insights can be found in [WM10]. Here we only give a short overview on the modules.

- (Server) OWL-Converter: converts OWL data to an internal representation
- (Server) Update Handler: determines changed islands in case of ontology updates
- (Server) Island Computer: computes the island for a given individual and performs similarity computation
- (Server) Node Scheduler: determines the responsible node for the island: Round-Robin / capability-based
- (Server) TBox/ABox Storage: terminological/assertional part of the ontology.
- (Client) Query Manager: determines all active islands and uses the DL Reasoner module to find out which islands match the input query.
- (Client) DL Reasoner: implements an interface to a general description logic reasoner (in our case we used RacerPro [HM03]).

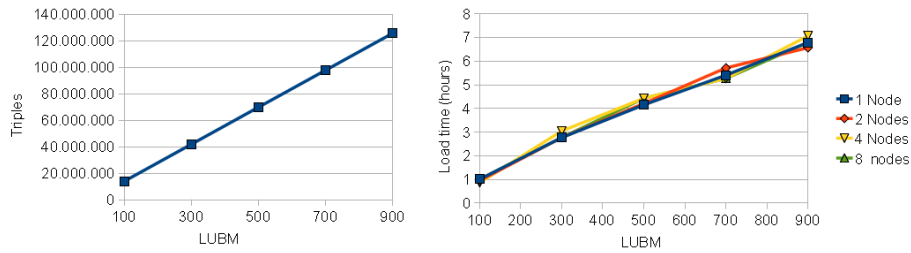


Fig. 4. Input size and load time (number of universities on the x-axis)

One more remark on our implementation should be pointed out here: While loading an ontology we built a dependency tree for storing which impact updates on particular islands have, e.g., we store that if we add a *teaches*-relation to a particular individual (island), 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 there were similar updates before. As usual, in order to obtain optimal performances, many details have to be appropriately handled.

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 4 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 once in order to compute the bisimilarity relation over all 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 internal structures, but only update relevant structures.
- *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 independent of the query and only performed once after the ontology

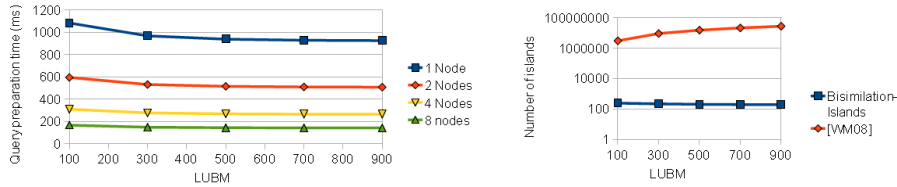


Fig. 5. Query preparation time and island count

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 5, 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 5 we indicate the necessary number of islands to perform instance retrieval with the original work in [WM08]. 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 process takes. In Figure 6, 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 because instances of *Chair* are not syntactically identifiable. In Figure 6 we show the time needed to identify the islands which entail instances of the concept *Chair*. This is the actual description-logic-hard task. In addition one needs to lookup all individuals for the given islands, which is a database-dominated task and usually takes linear time.

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

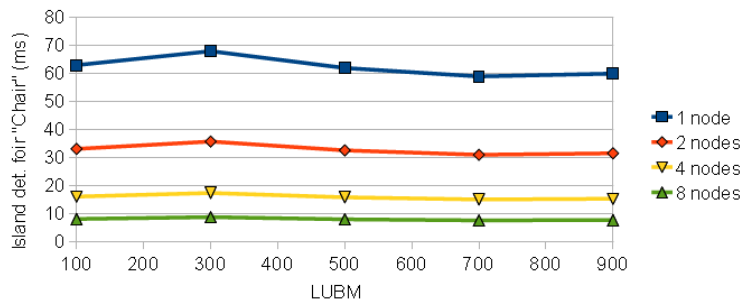


Fig. 6. Query answering time

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

- [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*, 1:153–184, 2003.
- [BVL03] S. Bechhofer, R. Volz, and P. Lord. Cooking the Semantic Web with the OWL API, 2003.
- [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.
- [DFK⁺09] Julian Dolby, Achille Fokoue, Aditya Kalyanpur, Edith Schonberg, and Kavitha Srinivas. Scalable highly expressive reasoner (SHER). *Web Semant.*, 7(4):357–361, 2009.
- [DG04] Jeffrey Dean and Sanjay Ghemawat. MapReduce: Simplified data processing on large clusters. In *OSDI 2004*, pages 137–150, 2004.
- [FKM⁺06] Achille Fokoue, Aaron Kershenbaum, Li Ma, Edith Schonberg, and Kavitha Srinivas. The summary abox: Cutting ontologies down to size. *The Semantic Web - ISWC 2006*, pages 343–356, 2006.
- [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.
- [HM03] V. Haarslev and R. Möller. Racer: A core inference engine for the semantic web. In *Proceedings of the 2nd International Workshop on Evaluation of Ontology-based Tools (EON2003), located at the 2nd International Semantic Web Conference ISWC 2003, Sanibel Island, Florida, USA, October 20*, pages 27–36, 2003.
- [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] Ramakrishna Soma and V. K. Prasanna. Parallel inferencing for OWL knowledge bases. In *ICPP '08: Proceedings of the 2008 37th International Conference on Parallel Processing*, pages 75–82, Washington, DC, USA, 2008. IEEE Computer Society.
- [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.
- [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.
- [WM10] Sebastian Wandelt and Ralf Möller. Distributed island-based query answering for expressive ontologies. Technical report, Institute for Software Systems (STS), Hamburg University of Technology, Germany, 2010. See <http://www.sts.tu-harburg.de/tech-reports/papers.html>.
- [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.