A Formal Ontology Reasoning with Individual Optimization: A Realization of the Semantic Web

Pakornpong Pothipruk and Guido Governatori

School of ITEE, University of Queensland, Australia
[pkp,guido]@itee.uq.edu.au

Abstract. Answering a query over a group of RDF data pages is a trivial process. However, in the Semantic Web, there is a need for ontology technology. Consequently, OWL, a family of web ontology languages based on description logic, has been proposed for the Semantic Web. Answering a query over the SemanticWeb is thus not trivial, but a deductive process. However, the reasoning on OWL with data has an efficiency problem. Thus, we introduce optimization techniques for the inference algorithm. This work demonstrates the techniques for instance checking and instance retrieval problems with respect to $\mathcal{ALC}$ description logic which covers certain parts of OWL.

1 Motivation

The *Semantic Web*, originated from an idea of the creator of the Web Tim Berners-lee [3], is an effort to bring back structure to information available on the Web. The structures are semantic annotations that conform to an explicit specification (called ontology) of the intended meaning of a piece of information. Thus the the Semantic Web contains implicit knowledge, and information on the Semantic Web is often incomplete since it assumes open-world semantics. In this perspective query answering on the Semantic Web is a deductive process [13].

A family of web ontology languages (OWL) based on *Description Logic* (DL) has been proposed as the languages to represent and reason with the Semantic Web. DL emphasizes clear unambiguous languages supported by complete denotational semantics and tractable/intractable reasoning algorithms [10]. Nevertheless, DL still faces problems when applied in context of the Web. One of them is the efficiency of query answering.

There are many works about DL reasoning optimization. However, most of them focus on DL-Tbox reasoning. In fact, DL-Abox reasoning, which is the basis for query answering over the Semantic Web, was seriously studied by some researchers recently. At present, there are only two prominent works for DL-Abox reasoning optimization, i.e., Instance Store [9] and RACER [8]. Instance Store uses a DL reasoner to classify Tbox and a database to store Abox. The database is also used to store a complete realization of the Abox. However, this approach only works with roll-free Abox. Thus an Abox reasoning over OWL or $\mathcal{SHOIQ}(\mathcal{D})$ will be reduced to an Abox reasoning over $\mathcal{ALCO}(\mathcal{D})$ without existential and universal quantifications, a very weak DL. Thus, this approach is inappropriate in practice. Another approach, RACER, proposes several optimization techniques for retrieval (an Abox reasoning), e.g., binary instance retrieval

M. Kitsuregawa et al. (eds)
Web Information Systems Engineering. WISE 2005.
LNCS 3806, pp. 119–132, 2005.

and dependency-based instance retrieval techniques. These techniques try to eliminate an individual, to be (SAT) tested, at-a-time.

Nevertheless, it would be better if we have another technique that can eliminate a chunk of individuals at-a-time from retrieval reasoning. Thus, we create optimization techniques that support this idea. In addition, we also introduce a novel optimization technique for instance checking, an Abox reasoning. We address the efficiency issue by a space partitioning and reduction approach.

2 Description Logic

The Semantic Web community implicitly adopted DL as a core technology for the ontology layer. One of the reasons behind this is that this logic has been heavily analyzed in order to understand how constructors interact and combine to affect tractable reasoning, see [4]. Technically, we can view the current Semantic Web, not including rule, proof and trust layers, as a DL knowledge base. Thus, answering a query posed on the Semantic Web (RDF and ontology layers) can be reduced to answering a query posed on a DL knowledge base, not taking into account low-level operations, such as name space resolution.

Description logic itself can be categorized into many different logics, distinguished by the set of constructors they provide. We focus on $\mathcal{ALC}$ description logic since it is the basis of many DL systems.

The language of $\mathcal{ALC}$ consists of an alphabet of distinct concept names **CN**, role names **RN**, and individual names **IN**, together with a set of constructors for building concept and role expressions [14].

Formally, a description logic knowledge base is a pair $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$ where $\mathcal{T}$ is a Tbox, and $\mathcal{A}$ is an Abox. The Tbox contains a finite set of axiom assertions. Axiom assertions are of the form

$$C \sqsubseteq D \mid C \doteq D,$$

where C and D are concept expressions. Concept expressions are of the form

$$A \mid \top \mid \ \bot \ \mid \neg C \mid C \sqcap D \mid C \sqcup D \mid \exists R.C \mid \forall R.C,$$

where A is an atomic concept or concept name in **CN**, R is a role name in **RN**, $\top$ (top or full domain) is the most general concept, and $\bot$ (bottom or empty set) is the least general concepts. The Abox contains a finite set of assertions about individuals of the form $a : C$ or $C(a)$ (concept membership assertion) and $(a,b) : R$ or $R(a,b)$ (role membership assertion), where a, b are names in **IN**.

The semantics of description logic is defined in terms of an interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \bullet^{\mathcal{I}})$, consisting of a nonempty domain $\Delta^{\mathcal{I}}$ and an interpretation function $\bullet^{\mathcal{I}}$. The interpretation function maps concept names into subsets of the domain ($A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$), role names into subsets of the Cartesian product of the domain ($R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$), and individual names into elements of the domain. The only restriction on the interpretations is the so called unique name assumption (UNA), which imposes that different individual names must be mapped into distinct elements of the domain. Given a concept name A (or role name R), the set denoted by $A^{\mathcal{I}}$ (or $R^{\mathcal{I}}$) is called the *interpretation*, or *extension*, of A (or R) with respect to $\mathcal{I}$.

The interpretation is extended to cover concepts built from negation ($\neg$), conjunction ($\sqcap$), disjunction ($\sqcup$), existential quantification ($\exists R.C$) and universal quantification ($\forall R.C$) as follows:

$$\begin{aligned}
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}}\\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}}\\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}}\\
(\exists R.C)^{\mathcal{I}} &= \left\{x \in \Delta^{\mathcal{I}} | \exists y.\langle x,y\rangle \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}}\right\}\\
(\forall R.C)^{\mathcal{I}} &= \left\{x \in \Delta^{\mathcal{I}} | \forall y.\langle x,y\rangle \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}}\right\}
\end{aligned}$$

An interpretation $\mathcal{I}$ satisfies (entails) an inclusion axiom $C \sqsubseteq D$ (written $\mathcal{I} \models C \sqsubseteq D$) if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it satisfies an equality $C \doteq D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$. It satisfies a Tbox $\mathcal{T}$ if it satisfies each assertion in $\mathcal{T}$. The interpretation $\mathcal{I}$ satisfies a concept membership assertion $C(a)$ if $a^{\mathcal{I}} \in C^{\mathcal{I}}$, and satisfies a role membership assertion $R(a,b)$ if $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$. $\mathcal{I}$ satisfies an Abox $\mathcal{A}$ (written $\mathcal{I} \models \mathcal{A}$) if it satisfies each assertion in $\mathcal{A}$. If $\mathcal{I}$ satisfies an axiom (or a set of axioms), then we say that it is a *model* of the axiom (or the set of axioms). Two axioms (or two sets of axioms) are *equivalent* if they have the same models. Given a knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A}\rangle$ we will say that the knowledge bases entails an assertion α (written $\mathcal{K} \models \alpha$) iff for every interpretation $\mathcal{I}$, if $\mathcal{I} \models \mathcal{A}$ and $\mathcal{I} \models \mathcal{T}$, then $\mathcal{I} \models \alpha$.

3 The Efficiency Issue

In DL, there are two standard types of queries allowed, i.e., boolean query and non-boolean query, which are in turn instance checking (or instantiation test) and retrieval Abox reasoning services respectively.

A boolean query $\mathcal{Q}_b$ refers to a formula of the form

$$\mathcal{Q}_b \leftarrow QExp,$$

where *QExp* is an assertion about an individual, e.g.,

$$\mathcal{Q}_b \leftarrow Tom : (Parent \sqcap \exists hasChild.Employee)$$

The query will return one of the member of the boolean set $\{True, False\}$. $\mathcal{Q}_b$ will return $True$ if and only if every interpretation that satisfies the knowledge base $\mathcal{K}$ also satisfies *QExp*, and return $False$ otherwise.

A non-boolean query $\mathcal{Q}_{nb}$ refers to a formula of the form

$$\mathcal{Q}_{nb} \leftarrow QExp,$$

where *QExp* is a concept expression, e.g.

$$\mathcal{Q}_{nb} \leftarrow Parent \sqcap \exists hasChild.Employee$$

In this case the query will return one of the member of the set $\{\perp, \mathcal{M}\}$, where $\perp$ refers to the empty set, and $\mathcal{M}$ represents a set of models $\{\mathcal{M}_1, \ldots, \mathcal{M}_m\}$, where each of them satisfies *QExp* with respect to the knowledge base $\mathcal{K}$. The query will return $\mathcal{M}$ if and only if there exists at least one such model, otherwise return $\perp$.

A non-boolean query (retrieval) can be trivially transformed into a set of boolean queries for all candidate tuples, i.e., retrieving sets of tuples can be achieved by repeated application of boolean queries with different tuples of individuals substituted for variables. However, answering a boolean query is in fact an entailment problem. For example, answering the boolean query:

$$\mathcal{Q}_b \leftarrow Tom : (Parent \sqcap \exists hasChild.Employee),$$

is the problem of checking whether

$$\mathcal{K} \models Tom : (Parent \sqcap \exists hasChild.Employee).$$

In a DL (supporting full negation, e.g., $\mathcal{ALC}$), boolean query or instance checking can be reduced to knowledge base satisfiability test: $\mathcal{K} \models C(a)$ iff $\mathcal{K} \cup \{\neg C(a)\}$ is unsatisfiable.

[5] gave a tableaux algorithm for solving $\mathcal{ALC}$ satisfiability problem with respect to a Tbox. They proved that their algorithm has EXPTIME-complete worst-case complexity. To the best of our knowledge, this is the latest known result of complexity proof for the $\mathcal{ALC}$ satisfiability problem with respect to a Tbox. Nevertheless, the ontology language OWL, in particular OWL-DL, of the Semantic web technology is based on $\mathcal{SHOIQ}(\mathcal{D})$ which is even more expressive than $\mathcal{ALC}$. Since the query answering is in fact an instance checking (or a retrieval reasoning service) which can be reduced to a satisfiability problem. It is easy to verify that the existing DL reasoning services are still not enough to be used solely with the Semantic web technology. One way to mitigate the problem is to optimize the algorithm even more. We propose an optimization technique for answering a query over a description logic knowledge base. This technique is coherent with nature of the Web in that it supports multiple-Abox environment, which corresponds to multiple data source environment in the Web.

4 Space Partitioning and Reduction Approach

This work focuses on finding an answer to the question: "How can we (efficiently) answer a query in a description logic, in particular $\mathcal{ALC}$, based-Semantic Web system, given single ontology $\mathcal{T}$, and multiple data sources $\mathcal{A}$s, examining the minimum number of data sources?". We refer to an Abox as a data source. Due to the space limitation, we present the result for boolean query only.

The idea of this section is based on the observation that $2^m > 2^n + 2^p$, where $m = n + p$ for $n, p > 1$. This means that if we can split the search space into independent parts, query the parts independently from each other, and combine the answers, then we have an improvement of the performance of the query system. This idea agrees with the understanding of the Semantic Web as a collection of sometime "unrelated" data sources. In addition we propose to attach to each data source a data source description (or source description), a compact representation of the content of the data page. This idea is similar to the intuition behind indexes in databases. In the same way that type of indexes is more or less appropriate for particular queries, source descriptions depend on the type of queries. On the other hand, as we will see in the rest of this section, the relationships among data sources are not influenced by queries. They are determined by the structure of the data itself.

4.1 The Knowledge Base

The intuition here is to associate to every Abox $\mathcal{A}$ a source description $SD(\mathcal{A})$, and to supplement the inference engine with information about the mutual dependencies of the Aboxes in the system, in order to determine which Aboxes are relevant and must be queried.

The first step is to associate to every Abox its domain.

Definition 1. *Given an Abox $\mathcal{A}$, let $H_{\mathcal{A}}$ be the Herbrand universe of $\mathcal{A}$ (i.e., the set of all the individual occurring in expression in $\mathcal{A}$). For any interpretation $\mathcal{I}$, $\Delta^{\mathcal{I}}_{\mathcal{A}}$, the domain of $\mathcal{A}$ is defined as follows:*

$$\Delta^{\mathcal{I}}_{\mathcal{A}} = \left\{ d \in \Delta^{\mathcal{I}} | a \in H_{\mathcal{A}} \wedge a^{\mathcal{I}} = d \right\}.$$

Definition 2 (Multiple Assertional Knowledge Base). *Given a set $\bar{A}$ of Aboxes $\mathcal{A}_1, \ldots, \mathcal{A}_k$, i.e., $\bar{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_k\}$ and a Tbox $\mathcal{T}$, the* multiple assertional knowledge base *is the knowledge base $\mathcal{K} = \langle \mathcal{T}, \mathcal{A} \rangle$, where $\mathcal{A}$ is the Abox obtained from the union of all the Aboxes in $\bar{A}$, i.e., $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \ldots \cup \mathcal{A}_k$.*

A consequence of the above definition is that the interpretation domain of $\mathcal{A}$ is equivalent to the union of interpretation domains of the $\mathcal{A}_j$s ($\Delta^{\mathcal{I}}_{\mathcal{A}} = \bigcup_{1 \le j \le k} \Delta^{\mathcal{I}}_{\mathcal{A}_j}$). Since $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ by definition, thus, for arbitrary C, $C^{\mathcal{I}}_{\mathcal{A}} = \bigcup C^{\mathcal{I}}_{\mathcal{A}_j}$, for $j \in \{1, \ldots, k\}$ and $C_{\mathcal{A}_j}$ is the concept C that occurs in a concept membership assertion in $\mathcal{A}_j$.

4.2 The Algorithm

We approach the problem in 5 steps:

1. Determine dependencies among data sources, and group data sources which are dependent on each other together.
2. Associate each data source (or group of data sources) with a source description.
3. When one queries the knowledge base, exploit a procedure to find irrelevant data sources (or groups of data sources) with respect to the query, taking into account source descriptions and the query. Eliminate the irrelevant data sources (or groups of data sources) from query answering process, yielding a set of possible relevant data sources (or groups of data sources) to be queried.
4. For each remaining data source (or group of data sources) from the previous step, apply the existing query answering procedure to each of them, yielding answer from each of them.
5. Simply combine answers from the queried data sources (or groups of data sources) together, since each data source (or group of data sources) is independent with the other.

Since a reasoning procedure for simple query answering in the fourth step exists [14], we will focus on other steps, which are in fact the steps of the data source space partitioning and reduction using source description.

The approach can be implemented by the following algorithm.

Algorithm 1 *partitioned_QA*$(query, \mathcal{A})$:

```
depset = {}
answer = False
for all $\mathcal{A}_m \in \bar{\mathcal{A}}$ do
    $AG(\mathcal{A}_m)$ = create_abox_graph($\mathcal{A}_m$)
end for
for all 2-combinations $\{\mathcal{A}_i, \mathcal{A}_j\}$ of $\mathcal{A}$ do
    if find_abox_dependency($AG(\mathcal{A}_i)$,$AG(\mathcal{A}_j)$) = True then
        add dep($\mathcal{A}_i, \mathcal{A}_j$) to depset
    end if
end for
$\mathcal{A}^g$ = combine_dependent_abox($\mathcal{A}$,depset)
create_source_description($\mathcal{A}^g$)
for all $\mathcal{A}_h \in \mathcal{A}^g$ do
    if query_relevancy($SD(\mathcal{A}_h)$,$query$) = True then
        answer = answer $\vee$ instance_checking($\mathcal{A}_h$,$query$)
    end if
end for
```

First, since an Abox $\mathcal{A}_i$ can overlap with another Abox $\mathcal{A}_j$, we must consider multiple Aboxes at the same time. However, we will not treat all of the Aboxes as a single Abox, because, in this case, the associated reasoning is computational expensive. Consequently, we need some additional procedure to determine dependencies among Aboxes since we need to know which Aboxes should be considered together. In other word, we need to group dependent Aboxes together and treat them as a new single Abox consisting of multiple dependent Aboxes. To make this clear, we need to formally define the dependency between Aboxes in the context of Abox reasoning.

Firstly, we will introduce graph notation for an Abox.

Definition 3 (Abox Graph). *An Abox graph for an Abox $\mathcal{A}$, $AG(\mathcal{A})$, consists of a set N of nodes (vertexes), a set E of edges (arcs), and a function f from E to $\{(a,b) \mid a,b \in N\}$. Each edge, label ed R_i, represents exactly a role name of a role membership assertion $R_i(a,b) \in \mathcal{A}$. Hence, each node represents exactly one individual name. An Abox graph is a directed multigraph.*

The create_abox_graph function will produce an Abox graph $AG(\mathcal{A}_m)$ for each Abox $\mathcal{A}_m$. We will say that an Abox $\mathcal{A}$ is *connected* if its Abox graph $AG(\mathcal{A})$ is weakly connected (see its definition in any discrete mathematics textbook).

Definition 4 (Abox Dependency). *Given two connected Aboxes $\mathcal{A}_1$ and $\mathcal{A}_2$, where $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$; $\mathcal{A}_1$ and $\mathcal{A}_2$ depend on each other if the graph of Abox $\mathcal{A}$ is (weakly) connected, and independent otherwise.*

Proposition 1. *Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be two independent Aboxes in multiple assertional knowledge base. Let $\Delta^{\mathcal{I}}_{\mathcal{A}_1}$ and $\Delta^{\mathcal{I}}_{\mathcal{A}_2}$ be the domains of $\mathcal{A}_1$ and $\mathcal{A}_2$, then:*

- $\Delta^{\mathcal{I}}_{\mathcal{A}_1} \cap \Delta^{\mathcal{I}}_{\mathcal{A}_2} = \emptyset$;
- *for any concept C, $C^{\mathcal{I}}_{\mathcal{A}_1} \cap C^{\mathcal{I}}_{\mathcal{A}_2} = \emptyset$, where $C^{\mathcal{I}}_{\mathcal{A}_i}$ is the extension of C in $\Delta^{\mathcal{I}}_{\mathcal{A}_i}$.*

If $\mathcal{A}$ is unconnected, i.e., $\mathcal{A}_1$ and $\mathcal{A}_2$ are independent on each other, then it means that $\mathcal{A}_1$ and $\mathcal{A}_2$ do not share any common node (individual) because Aboxes $\mathcal{A}_1$ and $\mathcal{A}_2$ are already connected by themselves. Thus, we can use Abox graphs to determine Abox dependency.

For any unordered pair of Aboxes $\{\mathcal{A}_i, \mathcal{A}_j\}$, we determine the Abox dependency between the two Aboxes ($\mathcal{A}_i$ and $\mathcal{A}_j$). According to the definition, Abox dependency can be detected using the connectivity of the Abox graph of $\mathcal{A}$, i.e., $AG(\mathcal{A})$, where $\mathcal{A} = \mathcal{A}_i \cup \mathcal{A}_j$. Thus, we can exploit any UCONN (undirected graph connectivity problem) algorithm for this purpose. The function find_abox_dependency($AG(\mathcal{A}_i)$,$AG(\mathcal{A}_j)$) returns True if two Aboxes $\mathcal{A}_i$ and $\mathcal{A}_j$ depend on each other, and False otherwise. If the function returns True, then we add dep($\mathcal{A}_i, \mathcal{A}_j$) to the set "depset", i.e., the set that stores dependency value of each pair of Aboxes. Then we virtually combine dependent Aboxes together as a group by the function combine_dependent_abox($\mathcal{A}$,depset). The Abox $\bar{\mathcal{A}}$ will become $\mathcal{A}^g$, i.e., the set of already-grouped Aboxes and ungrouped Aboxes. Each Abox in $\mathcal{A}^g$ is independent of each other.

Next, we need to show two things:

1. if two Aboxes depend on each other, then a DL reasoning service, in particular instance checking and retrieval, needs to take into account the two Aboxes together;
2. if two Aboxes are independent of each other, then a DL reasoning over the two Aboxes can be done separately over each of them.

The following theorem supports the last step in our approach. It provides the reason why we can simply combine the answer from each $\mathcal{A}_i \in \mathcal{A}^g$ together. In other words it states that the the instance checking (a query answering) problem over $\mathcal{A}^g$ can be reduced to separate instance checking problems over each $\mathcal{A}_i$.

Theorem 1 (Independent Abox and Instance Checking). *Given two connected Aboxes $\mathcal{A}_1$ and $\mathcal{A}_2$, where $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, If $\mathcal{A}_1$ and $\mathcal{A}_2$ are independent on each other, then for any boolean query $\mathcal{Q}$ and Tbox $\mathcal{T}$, $\langle \mathcal{T}, \mathcal{A} \rangle \models \mathcal{Q}$ if and only if $\langle \mathcal{T}, \mathcal{A}_1 \rangle \models \mathcal{Q}$ or $\langle \mathcal{T}, \mathcal{A}_2 \rangle \models \mathcal{Q}$.*

Proof. First, we prove the only if direction, and we will assume that both $\mathcal{A}_1$ and $\mathcal{A}_2$ are consistent with $\mathcal{K}$, since if one of them is not then the theorem trivially holds.

Since $\mathcal{A}_1$ and $\mathcal{A}_2$ are *independent* on each other, by Proposition 1, we have $\Delta^{\mathcal{I}}_1 \cap \Delta^{\mathcal{I}}_2 = \emptyset$, where $\Delta^{\mathcal{I}}_1$ and $\Delta^{\mathcal{I}}_2$ are the domains of $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively.

Suppose $\langle \mathcal{T}, \mathcal{A}_1 \rangle \not\models \mathcal{Q}$ and $\langle \mathcal{T}, \mathcal{A}_2 \rangle \not\models \mathcal{Q}$. These mean $\exists \mathcal{I}_1$ such that $\mathcal{I}_1 \models \mathcal{A}_1$, $\mathcal{I}_1 \models \mathcal{T}$, $\mathcal{I}_1 \models \neg \mathcal{Q}$, and $\exists \mathcal{I}_2$ such that $\mathcal{I}_2 \models \mathcal{A}_2$, $\mathcal{I}_2 \models \mathcal{T}$ and $\mathcal{I}_2 \models \neg \mathcal{Q}$. Note that $\mathcal{I}_1$ and $\mathcal{I}_2$ are arbitrary interpretations of $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively with the only constraint of being interpretations of $\mathcal{T}$.

Since $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ and $\Delta^{\mathcal{I}}_1 \cap \Delta^{\mathcal{I}}_2 = \emptyset$, we can create an interpretation $\mathcal{I}$ of $\mathcal{A}$ such that $\mathcal{I}$ is the union of the interpretation $\mathcal{I}_1$ of $\mathcal{A}_1$ and the interpretation $\mathcal{I}_2$ of $\mathcal{A}_2$ ($\mathcal{I} = \mathcal{I}_1 \cup \mathcal{I}_2$). More precisely, $\mathcal{I} = \langle \Delta^{\mathcal{I}}, \bullet^{\mathcal{I}} \rangle$ is defined as follows:

(i) $\Delta^{\mathcal{I}} = \Delta_1^{\mathcal{I}} \cup \Delta_2^{\mathcal{I}}$ because $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$, where $\Delta^{\mathcal{I}}$ is the domains of $\mathcal{A}$

(ii) For any constant a,

$$a^{\mathcal{I}} = \begin{cases} a^{\mathcal{I}_1} & \text{if } a \text{ occurs in } \mathcal{A}_1 \\ a^{\mathcal{I}_2} & \text{if } a \text{ occurs in } \mathcal{A}_2 \end{cases}$$

(iii) For any concept C, $C^{\mathcal{I}} = C^{\mathcal{I}_1} \cup C^{\mathcal{I}_2}$

(iv) For any role R, $R^{\mathcal{I}} = R^{\mathcal{I}_1} \cup R^{\mathcal{I}_2}$

Since $\Delta_1^{\mathcal{I}} \cap \Delta_2^{\mathcal{I}} = \emptyset$, then it is immediate to verify that $\mathcal{I}$ is indeed an interpretation, and $\mathcal{I} \models \mathcal{T}$, since $\mathcal{I}_1 \models \mathcal{T}$ and $\mathcal{I}_2 \models \mathcal{T}$.

Since $\mathcal{I}_1 \models \neg\mathcal{Q}$ and $\mathcal{I}_2 \models \neg\mathcal{Q}$, from (iii), we can immediately verify $\mathcal{I} \models \neg\mathcal{Q}$, i.e., $(\neg\mathcal{Q})^{\mathcal{I}} = (\neg\mathcal{Q})^{\mathcal{I}_1} \cup (\neg\mathcal{Q})^{\mathcal{I}_2}$, where $\mathcal{I}$ is the interpretation of $\mathcal{A}$. From (ii), (iii) and (iv), we can also infer that $(\mathcal{A})^{\mathcal{I}} = (\mathcal{A}_1)^{\mathcal{I}_1} \cup (\mathcal{A}_2)^{\mathcal{I}_2}$, i.e., $\mathcal{I} \models \mathcal{A}$.

Since $\mathcal{A}_1$ and $\mathcal{A}_2$ are assumed to be consistent by themselves, we only need to prove that there is no clash between $\mathcal{A}_1$ and $\mathcal{A}_2$. For an arbitrary concept C, by general definition in description logic, we get $C^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$. In addition, we get $(\neg C)^{\mathcal{I}} = (\Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}) \subseteq \Delta^{\mathcal{I}}$. Thus, for arbitrary C, $C^{\mathcal{I}_1} \subseteq \Delta_1^{\mathcal{I}}$ and $(\neg C)^{\mathcal{I}_2} \subseteq \Delta_2^{\mathcal{I}}$. Since $\Delta_1^{\mathcal{I}} \cap \Delta_2^{\mathcal{I}} = \emptyset$, therefore, $C^{\mathcal{I}_1} \cap (\neg C)^{\mathcal{I}_2} = \emptyset$, which infers that no clash can occur between $\mathcal{A}_1$ and $\mathcal{A}_2$.

Thus for the interpretation $\mathcal{I}$ of $\mathcal{A}$, we have $(\mathcal{A})^{\mathcal{I}} \neq \emptyset$ and $(\neg\mathcal{Q})^{\mathcal{I}} \neq \emptyset$, i.e., $\mathcal{I} \models \mathcal{A} \wedge \mathcal{I} \models \neg\mathcal{Q}$ which is the definition of $\mathcal{A} \not\models \mathcal{Q}$. Therefore, $\mathcal{A} \models \mathcal{Q}$ only if $\mathcal{A}_1 \models \mathcal{Q}$ or $\mathcal{A}_2 \models \mathcal{Q}$ which infers $\langle \mathcal{T}, \mathcal{A} \rangle \models \mathcal{Q}$ only if $\langle \mathcal{T}, \mathcal{A}_1 \rangle \models \mathcal{Q}$ or $\langle \mathcal{T}, \mathcal{A}_2 \rangle \models \mathcal{Q}$.

For the if direction, we assume that either 1) $\langle \mathcal{T}, \mathcal{A}_1 \rangle \models \mathcal{Q}$ or 2) $\langle \mathcal{T}, \mathcal{A}_2 \rangle \models \mathcal{Q}$. In both cases, by monotonicity, we obtain $\langle \mathcal{T}, \mathcal{A}_1 \cup \mathcal{A}_2 \rangle \models \mathcal{Q}$ which is $\langle \mathcal{T}, \mathcal{A} \rangle \models \mathcal{Q}$. □

In the second step of the approach, we associate each Abox (or group of Aboxes) with a source description, using create_source_description($\mathcal{A}^g$). A source description can be view as a surrogate of each data source. Surrogate refers to a brief representation of an information source that is designed to convey an indication of the information source's intent [7]. A good surrogate has two major properties: (1) it corresponds to some common understanding in the user's community, and (2) it can be organized in a way that is searchable.

Source descriptions are used to determine the relevancy of each Abox $\mathcal{A}_h \in \mathcal{A}^g$ with respect to a query. Source descriptions depend on the type of the query. For boolean queries, the source description of each Abox $\mathcal{A}_h \in \mathcal{A}^g$ can be a simple list of all individuals appearing in the Abox $\mathcal{A}_h$. The idea is if the query does not satisfy $SD(\mathcal{A}_h)$ (necessary and sufficient conditions), it is guaranteed that the query over Abox $\mathcal{A}_h$ will fail, i.e., it returns False. This is done by the function query_relevancy($SD(\mathcal{A}_h)$,*query*). This function returns False if the query does not satisfy $SD(\mathcal{A}_h)$, i.e., the Abox $\mathcal{A}_h$ is fully irrelevant to the query, and will contribute nothing to the answer of the query. The function works by extracting an individual from the query, then checking if it is in the source description $SD(\mathcal{A}_h)$ or not. If it is, then it queries the Abox $\mathcal{A}_h$, using normal boolean query answering procedure instance_checking($\mathcal{A}_h$,*query*).

This can be formalised as follows:

Definition 5. *Let $\mathcal{A}$ be an Abox, the boolean query source description for $\mathcal{A}$ ($SD_b(\mathcal{A})$) is the the Herbrand universe of $\mathcal{A}$, i.e., $SD_b(\mathcal{A}) = H_{\mathcal{A}}$.*

We can now prove soundness and completeness of the above choice of source descriptions.

Theorem 2 (Soundness and Completeness of Instance Checking Optimization). *Let $\mathcal{Q}$ be a boolean query. Let $\mathcal{A} \not\sharp \mathcal{Q}$ represents when query_relevance($SD_b(\mathcal{A}),\mathcal{Q}$) returns False, i.e., $\mathcal{A}$ is not relevant the query $\mathcal{Q}$, and let $\mathcal{A}\sharp\mathcal{Q}$ represents otherwise. If $\mathcal{A} \not\sharp \mathcal{Q}$ then $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$ if and only if $\langle\mathcal{T},\bar{\mathcal{A}}\rangle \models \mathcal{Q}$.*

Proof. Suppose $\mathcal{A} \not\sharp \mathcal{Q}$. This means $a \notin SD_B(\mathcal{A})$, where $\mathcal{Q}$ is $C(a)$.

First, we prove the only if direction. Suppose $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$. However, $\bar{\mathcal{A}}-\{\mathcal{A}\} \subseteq \bar{\mathcal{A}}$. By monotonicity, we obtain $\langle\mathcal{T},\bar{\mathcal{A}}\rangle \models \mathcal{Q}$.

Therefore, $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$ only if $\langle\mathcal{T},\bar{\mathcal{A}}\rangle \models \mathcal{Q}$.

For the if direction, suppose $\langle\mathcal{T},\bar{\mathcal{A}}\rangle \models \mathcal{Q}$. We, then, prove by case.

Case 1: $\mathcal{Q}$ is a tautology. It is immediate to verify that $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$ is true.

Case 2: $\mathcal{Q}$ is not a tautology. From Lemma 1, we obtain $\langle\mathcal{T},\mathcal{A}\rangle \not\models \mathcal{Q}$. In addition, $\langle\mathcal{T},\bar{\mathcal{A}}\rangle \models \mathcal{Q}$ is equal to $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\cup\{\mathcal{A}\}\rangle \models \mathcal{Q}$. By Theorem 1, we get $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$ or $\langle\mathcal{T},\mathcal{A}\rangle \models \mathcal{Q}$. Since $\langle\mathcal{T},\mathcal{A}\rangle \not\models \mathcal{Q}$, we obtain $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$.

These cases cover all possibilities. Therefore, $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$ if $\langle\mathcal{T},\bar{\mathcal{A}}\rangle \models \mathcal{Q}$.

Therefore, if $\mathcal{A} \not\sharp \mathcal{Q}$ then $\langle\mathcal{T},\bar{\mathcal{A}}-\{\mathcal{A}\}\rangle \models \mathcal{Q}$ if and only if $\langle\mathcal{T},\bar{\mathcal{A}}\rangle \models \mathcal{Q}$. □

Lemma 1. *Let $\mathcal{Q}$ be a boolean query $C(a)$. If $a \notin SD_B(\mathcal{A})$ and $\mathcal{Q}$ is not a tautology, then $\langle\mathcal{T},\mathcal{A}\rangle \not\models \mathcal{Q}$.*

Proof. Suppose $a \notin SD_B(\mathcal{A})$. By definition, this means $a \notin \Delta^{\mathcal{I}}_{\mathcal{A}}$. In other words, there is no a in $\mathcal{A}$, i.e., $C(a)$ is definitely not in $\mathcal{A}$. Supppose $\mathcal{Q}$ is not a tautology. At this state, we want to prove $\langle\mathcal{T},\mathcal{A}\rangle \not\models C(a)$ which is equal to proving that $\langle\mathcal{T},\mathcal{A}\rangle \cup \{\neg C(a)\}$ is satisfiable. Since $\langle\mathcal{T},\mathcal{A}\rangle$ is consistent, $\langle\mathcal{T},\mathcal{A}\rangle \cup \{\neg C(a)\}$ will be unsatisfiable only if either $C(a)$ is in $\mathcal{A}$ or $C(a)$ is a tautology. Since neither of them is true, $\langle\mathcal{T},\mathcal{A}\rangle \cup \{\neg C(a)\}$ is satisfiable. Consequently, we prove $\langle\mathcal{T},\mathcal{A}\rangle \not\models C(a)$.

Therefore, if $a \notin SD_B(\mathcal{A})$ and $\mathcal{Q}$ is not a tautology, then $\langle\mathcal{T},\mathcal{A}\rangle \not\models \mathcal{Q}$. □

Finally, in the last step we simply combine the answers together using disjunction. Again this step is justified by Theorem 1.

5 A Comprehensive Example

Suppose we have a knowledge bases $\mathcal{K} = \langle\mathcal{T},\mathcal{A}\rangle$, where the data is distributed over four Aboxes, i.e., $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3 \cup \mathcal{A}_4$.

Suppose the Tbox $\mathcal{T}$ is a follows:

$$
\begin{aligned}
\mathcal{T} = \{&Man \doteq \neg Female, Woman \doteq Human \sqcap Female, Man \doteq Human \sqcap Male,\\
&Mother \doteq Woman \sqcap \exists hasChild.Human, Father \doteq Man \sqcap \exists hasChild.Human,\\
&Parent \doteq Mother \sqcup Father, Organization \doteq Profit \sqcup Charity,\\
&Employee \doteq Human \sqcap \exists workAt.Organization,\\
&Profit \doteq Company \sqcup Partnership \sqcup SoleOwnerCharity \doteq \neg Profit,\\
&CommunicationCompany \doteq Company \sqcap \exists provideService.CommunicationService,\\
&CommunicationService \doteq MobileService \sqcup TelephoneService \sqcup InternetService\}
\end{aligned}
$$

Suppose that the four Aboxes are as follows:

$$\begin{aligned}
\mathcal{A}_1 = \{&Man(Tom), Man(Peter), Woman(Mary), SoleOwner(ThaiOrchid),\\
&hasChild(Tom, Peter), workAt(Peter, AIS), attend(Peter, MobileSys2003),\\
&hasChild(Tom, Mary), workAt(Mary, ThaiOrchid), studyAt(Mary, ITEE)\}\\
\mathcal{A}_2 = \{&Department(ITEE), Faculty(EPSA),\\
&University(UQ), Conference(MobileSys2003), partOf(ITEE, EPSA),\\
&facultyIn(EPSA, UQ), locatedIn(UQ, Australia)\}\\
\mathcal{A}_3 = \{&CommunicationCompany(AIS), Conference(MobileSys2003),\\
&CommunicationCompany(DTAC), locatedIn(AIS, Thailand),\\
&sponsor(MobileSys2003, AIS), sponsor(MobileSys2003, DTAC),\\
&hold(ITEE, MobileSys2003)\}\\
\mathcal{A}_4 = \{&Charity(PinTao), CharityProject(MMM), CharityProject(TOLS),\\
&locatedIn(PinTao, Malaysia), propose(PinTao, MMM), propose(PinTao, TOLS)\}
\end{aligned}$$

We simply create the Abox graph for each Abox, yielding four Abox graphs: $AG(\mathcal{A}_1)$, $AG(\mathcal{A}_2)$, $AG(\mathcal{A}_3)$, and $AG(\mathcal{A}_4)$. For every combination of two Aboxes of $\mathcal{A}$, we determine the Abox dependency between them using the find_abox_dependency function. The function will combine the two Aboxes together, and apply UCONN algorithm to the graph of the combined Abox.

If the graph is connected, then the two Aboxes depend on each other. We, then, add dep($\mathcal{A}_i, \mathcal{A}_j$) to the set depset as they are dependent, i.e., the function returns True. We get

$$\text{depset} = \{\text{dep}(\mathcal{A}_1, \mathcal{A}_2), \text{dep}(\mathcal{A}_1, \mathcal{A}_3), \text{dep}(\mathcal{A}_2, \mathcal{A}_3)\}$$

Since we know that $\mathcal{A}_1$ depends on $\mathcal{A}_2$, and also on $\mathcal{A}_3$, we virtually group them together, i.e., we will consider the data in both three Aboxes together. This is done by the combine_dependent_abox($\mathcal{A}$,depset) function. As a result we have $\mathcal{A}^g = \{\mathcal{A}_{123}, \mathcal{A}_4\}$, where $\mathcal{A}_{123} = \mathcal{A}_1 \cup \mathcal{A}_2 \cup \mathcal{A}_3$.

At this stage, we create source description for each Abox in $\mathcal{A}^g$, using the create_source_description($\mathcal{A}^g$) procedure:

- $SD(\mathcal{A}_{123}) = (AIS, Australia, DTAC, EPSA, ITEE, Mary,$ $MobileSys2003, Peter, Thailand, ThaiOrchid, Tom, UQ)$
- $SD(\mathcal{A}_4) = (Malaysia, MMM, PinTao, TOLS)$

Suppose, we have a boolean query $Tom : (Parent \sqcap \exists hasChild.Employee)$. For every $\mathcal{A}_h \in \mathcal{A}^g$, we determine relevancy with respect to the query, using the procedure query_relevancy($SD(\mathcal{A}_h)$,*query*). The procedure will extract "Tom" from the query, and search whether "Tom" is in $SD(\mathcal{A}_h)$ or not. In this case, "Tom" is in $SD(\mathcal{A}_{123})$, but not in $SD(\mathcal{A}_4)$. Consequently, we simply query $\mathcal{A}_{123}$, using the instance_checking($\mathcal{A}_h$, *query*) function. The result from instance checking test of the query in $\mathcal{A}_{123}$ is True. Thus, answer = False $\vee$ True = True, which is the same result as when we query the whole Abox $\mathcal{A}$.

6 Complexity Analysis

Each Abox Graph can be trivially generated in $O(n^2)$, where n is the number of assertions in Abox $\mathcal{A}$. The next part is Abox dependency determination. We need $k(k-1)/2$ comparisons (2-combinations) of unordered pair of Aboxes, where k is the number of Aboxes in $\mathcal{A}$. Each comparison needs UCONN algorithm. UCONN can be solved by DFS in $O(v^2)$, where v is the number of individuals in each Abox.

For source description, the create_source_description($\mathcal{A}^g$) procedure requires not more than $O(n^2)$ for all Aboxes, since it can be implemented using the quick sort algorithm. To determine the relevancy of an Abox to a query, we call the function query_relevancy($SD(\mathcal{A}_h)$,*query*) that operates in $O(n)$ for sequential search. Finally, we simply use the instance_checking($\mathcal{A}_h$,*query*) function to find the answer for each Abox, and simply combine the answer.

Till now, our space partitioning and reduction approach exploits at most PTIME algorithms in each part, i.e., the Abox dependency part and the source description part. Overall, our algorithm can be operated in PTIME, not including the instance checking part. Since the instance checking part is known to be solved in EXPTIME-complete, assuming P $\neq$ EXPTIME, the overall algorithm still operates in EXPTIME, but with a *reduced exponent*. The Abox dependency part will partition the search space, thus the exponent will be reduced if there are at least two partitions, e.g., the time complexity is reduced from 2^m to $2^n + 2^p$, where $m = n + p$. The source description part will further reduce the exponent if there are some Aboxes which can be eliminated from the process, e.g., the time complexity is reduced from 2^m to 2^q, where $q < m$.

7 The Extension

We intend to extend our work to non-boolean query answering (Abox retrieval) optimization. The space partitioning, Theorem 1, can be easily extended to cover Abox retrieval reasoning services. The main differences will be with the source description. Furthermore, we will investigate possible extensions of Theorem 1 to more expressive DLs.

In the nutshell, a source description of an Abox $\mathcal{A}$ for retrieval ($SD(\mathcal{A})$) consists of several types of source descriptions. In the basic setting, we shall have $SD^C(\mathcal{A})$ and $SD^R(\mathcal{A})$. $SD^C(\mathcal{A})$ is a set of *least common subsumer*s or *LCS*s of concepts appeared in concept membership assertions in the Abox $\mathcal{A}$. $SD^R(\mathcal{A})$ is a set of roles appeared in role membership assertions in the Abox $\mathcal{A}$. Let Q be a non-boolean query, C, D be concept expressions, and $\mathcal{A}_h$ be an Abox. The source description usage is different for each form of Q, for example:

- For the case $Q \leftarrow C$, $\mathcal{A}_h$ is relevant to Q if $\exists s \in SD^C(\mathcal{A}_h), Q \sqsubseteq s$
- For the case $Q \leftarrow C \sqcap D$, $\mathcal{A}_h$ is relevant to Q if $\mathcal{A}_h$ is relevant to C *and* $\mathcal{A}_h$ is relevant to D.
- For the case $Q \leftarrow C \sqcup D$, $\mathcal{A}_h$ is relevant to Q if $\mathcal{A}_h$ is relevant to C *or* $\mathcal{A}_h$ is relevant to D.
- For the case $Q \leftarrow \exists R.C$, $\mathcal{A}_h$ is relevant to Q if $R \in SD^R(\mathcal{A}_h)$ *and* $\mathcal{A}_h$ is relevant to C.

– For the case $Q \leftarrow \neg C$, $\mathcal{A}_h$ is relevant to Q if $\mathcal{A}_h$ is irrelevant to C.

These operations can be recursive. Hence, we can achieve a methodology for determine relevancy of the Abox $\mathcal{A}_h$ with respect to the arbitrary-formed non-boolean query Q. Note that the correctness of the above methodology follows from the independent Abox theorem, extended for non-boolean query.

8 Discussion and Related Works

The optimization approach presented in this work is based on the procedure normally adopted for deduction over a description logic knowledge base (the query answering process over such knowledge base is a deduction process). In particular we refer to the tableaux algorithm. Traditionally, tableaux algorithm was designed to prove the satisfiability problem. The main idea behind this algorithm is based on a notational variant of the first order tableaux calculus. In fact, a tableaux algorithm tries to prove the satisfiability of a concept expression C by demonstrating a nonempty model of C. It constructively builds a model for a given concept. The process of constructing a model proceeds by completing a constraint system [14], using a set of consistency-preserving completion (or expansion) rules. The process will continue if it can extend the existing constraint system. In $\mathcal{ALC}$ reasoning with $\mathcal{T}$ and $\mathcal{A}$, the process will proceed via a role membership assertion. The idea behind our work is to specify the condition where we guarantee that the reasoning process over $\mathcal{A}_1$ will never proceed to $\mathcal{A}_2$ if $\mathcal{A}_1$ and $\mathcal{A}_2$ are independent from each other. This optimization, in particular, the space partitioning part, can be seen as a divide-and-conquer technique. A general disadvantage of this kind of technique is the parts overlap. However, in this work we proposed a methodology to avoid overlapping part, thus, it does not suffer from such disadvantage of the divide-and-conquer technique.

Apparently, the major drawback of this approach is obviously the additional cost from Abox dependencies and source descriptions determination. But the cost is still in PTIME ($\subseteq$ PSPACE), as shown in previous section. One may argue that our space partitioning approach would support the reduction in apparent worst-case complexity for query answering, but at a high price in practice, in particular for a large knowledge base. However, this is not a scholarly argument, because the larger the knowledge base is, the less the relative cost is (recall that the normal query answering is in EXPTIME while the additional cost is in PTIME). Thus, our approach should behave well in practice. The only issue that we must give additional attention to is a design of effective Abox dependency information distribution, minimizing information exchange between nodes in the network, where each node represents an Abox. In addition, if the data pages do not change frequently, then there is no need to recompute the dependency of the Abox. In addition data source can be organised in indexes for fast retrieval.

This approach can be applied to a system which allows only one ontology (or Tbox). Though the Semantic Web technology tends to exploit multiple ontologies. In ontology-based integration of information area [15], we can divide the exploitation of ontology into 3 approaches: single-ontology approach, e.g., SIMS [2], multiple-ontologies approach, e.g., OBSERVER [11], and hybrid-ontology approach, e.g., COIN [6]. The multiple-ontologies approach requires additional mapping specifications between each pair of ontologies. Since such mappings are infact ontologies themselves [1], we need additional $n(n-1)/2$ ontologies for such an approach, where n is number of existing ontologies in the system. In hybrid-ontology approach, a global ontology and additional n mapping specifications (between global ontology and each local ontology) are required. Hence the single-ontology approach can be viewed as generalization of the other two approaches. Thus, we follow such approach. In addition, since the aim of our work is to study how to query multiple data sources, thus we do not need to add complexity arisen from ontology mapping in the last two approaches. Simple single-ontology approach, but not trivial for query answering, is enough. Note that we can extend our work to include multiple ontologies later when the research about ontology binding and ontology mapping and ontology integration is more mature.

This approach can be applied to a system which allows multiple data sources (or Aboxes). We can think of an Abox as an RDF document. In addition RDF databases try to partition RDF triples in disjoint graphs , where each graph can be understood as a data page of our approach. However, recent research has shown that there are several semantic problems when people tried to layer an ontology language, i.e., OWL, on top of the RDF layer [12]. Such problems stem from some features/limitations of RDF, e.g., no restriction on the use of built-in vocabularies, and no restriction on how an RDF statement can be constructed since it is just a triple. Thus, this implies that the ontology layer may be not compatible with the RDF layer of the Semantic Web. However, there is a work proposing additional layer on top of the RDF layer, i.e., RDF(FA) [12]. This layer corresponds directly to Aboxes, thus RDF(FA) may be very useful in the future.

There are few works related to our work, i.e., Instance Store [9] and RACER [8]. Both works propose retrieval optimization techniques. Hence, our approach seems to be the first approach for Abox instance checking optimization. Instance Store imposes an unnatural restriction on Abox, i.e., enforcing Abox to be role-free. This is a severe restriction since role names are included even for $\mathcal{FL}^-$ ($\mathcal{ALC}$ without atomic negation), a DL with limited expressive power. RACER proposes several innovative Abox reasoning optimization techniques. However, RACER allows single Abox, while our approach allows multiple Aboxes. Thus after we apply our techniques to reduce the reasoning search space, we can apply RACER techniques to reduce it further. Consequently, the approach taken in RACER seems to be complementary to ours. We will investigate the combination of our approach and RACER approach in the future.

Acknowledgements

This work was partially supported by the University of Queensland Early Career Researcher Grant no. 2004001458 on “A System for Automated Agent Negotiation with Defeasible Logic-Based Strategies”.

References

1. J. Akahani, K. Hiramatsu, and K. Kogure. Coordinating heterogeneous information services based on approximate ontology translation. In *Proceedings of AAMAS-2002 Workshop on Agentcities: Challenges in Open Agent Systems*, pages 10–14, 2002.

2. Y. Arens, C. Hsu, and C. A. Knoblock. Query processing in the sims information mediator. *Advanced Planning Technology*, 1996.
3. T. Berners-Lee. *Weaving the Web : the Original Design and Ultimate Destiny of the World Wide Web by its Inventor*. HarperSanFrancisco, San Francisco, 1999.
4. F. M. Donini, M. Lenzerini, D. Nardi, and W. Nutt. The complexity of concept languages. In J. Allen, R. Fikes, and E. Sandewall, editors, *Proceedings of the Second International Conference on Principles of Knowledge Representation and Reasoning (KR-91)*, pages 151–162, Massachusetts, 1991.
5. F. M. Donini and F. Massacci. Exptime tableaux for $\mathcal{ALC}$. *Artificial Intelligence*, 124(1):87–138, 2000.
6. C. H. Goh. *Representing and Reasoning about Semantic Conflicts in Heterogeneous Information Sources*. PhD thesis, MIT, 1997.
7. A. Goodchild. *Database Discovery in the Organizational Environment*. PhD thesis, University of Queensland, 1998.
8. V. Haarslev and R. Möller. Optimization strategies for instance retrieval. In *Proceedings of the International Workshop on Description Logics (DL 2002)*, 2002.
9. I. Horrocks, L. Li, D. Turi, and S. Bechhofer. The instance store: Description logic reasoning with large numbers of individuals. In *International Workshop on Description Logics (DL 2004)*, pages 31–40, 2004.
10. D. L. McGuinness, R. Fikes, L. A. Stein, and J. Hendler. Daml-ont: An ontology language for the semantic web. In D. Fensel, J. Hendler, H. Lieberman, and W. Wahlster, editors, *Spinning the Semantic Web: Bringing the World Wide Web to its Full Potential*. MIT Press, 2003.
11. E. Mena, V. Kashyap, A. Sheth, and A. Illarramendi. Observer: An approach for query processing in global information systems based on interoperability between pre-existing ontologies. In *Proceedings of the* 1^{st} *IFCIS: International Conference on Cooperative Information Systems (CoopIS '96)*, 1996.
12. J. Z. Pan and I. Horrocks. Rdfs(fa): A dl-ised sub-language of rdfs. In *Proceedings of the 2003 International Workshop on Description Logics (DL2003)*, 2003.
13. H. Stuckenschmidt. Query processing on the semantic web. *Künstliche Intelligenz*, 17, 2003.
14. S. Tessaris. *Questions and Answers: Reasoning and Querying in Description Logic*. PhD thesis, University of Manchester, 2001.
15. H. Wache, T. Vögele, U. Visser, H. Stuckenschmidt, G. Schuster, H. Neumann, and S. Hübner. Ontology-based integration of information - a survey of existing approaches. In *Proceedings of the IJCAI-01 Workshop: Ontologies and Information Sharing*, pages 108–117, 2001.