Partitioning ABoxes Based on Converting DL to Plain Datalog

Jianfeng Du[1,2] and Yi-Dong Shen[1]

[1] State Key Laboratory of Computer Science,
Institute of Software, Chinese Academy of Sciences
[2] Graduate University of the Chinese Academy of Sciences
Beijing 100080, China
`jfdu,ydshen@ios.ac.cn`

Abstract. To make ABox reasoning scalable for large ABoxes in description logic (DL) knowledge bases, we develop a method for partitioning the ABox so that specific kinds of reasoning can be performed separately on each partition and the results trivially combined can achieve complete answers. Our method applies to $\mathcal{SHIQ}(\mathbf{D})$ knowledge bases. It first converts a DL knowledge base KB to a plain datalog program $H(KB)$, and then computes the least fixpoint of the definite part of $H(KB)$ while generating ABox partitions. Its time data complexity is polynomial in the ABox size, under some general assumption on concrete domains. Experimental results further demonstrate the advantages of our method.

1 Introduction

ABox reasoning (e.g., query answering) in DL knowledge bases is a great challenge, due to high complexity of reasoning in expressive DL languages [2] and the resource limitations (e.g., physical memory) for handling large ABoxes. As [3] pointed out, there are two approaches towards scalable ABox reasoning. One approach is to partition the ABox so that some kinds of reasoning can be performed separately on each partition [3, 5]. Another approach is to convert DL to disjunctive datalog and use deductive databases to reason over the ABox [7]. Based on [7], Motik *et al.* [8] propose a resolution based algorithm which evaluates non-ground queries in one pass so that the efficiency of the query answering is further improved. However, the algorithm requires exponential space in the worst case, thus it is particularly important to partition the ABox to cut down memory consumption.

In this paper, we present a new method for partitioning the ABox. For a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB, we first reduce KB to an equisatisfiable disjunctive datalog program $DD(KB)$, using the methods in [7]. Then, we convert $DD(KB)$ to a plain datalog program $H(KB)$ by replacing disjunctions with definite implications, and compute the least fixpoint of the definite part of $H(KB)$. During the fixpoint computation, we put sets of assertions that (indirectly) trigger rules in $H(KB)$ to the same part and keep track of the triggering information on each assertion. At last, we adjust parts to partitions according to the tracking data. The time data complexity of our method is polynomial in the ABox size, assuming a polynomial oracle for reasoning with concrete domains and a constant bound on the arity of the concrete domain predicates. Our method always produces a disjoint and independent partitioning, such that each assertion over atomic concepts or simple roles is entailed by the whole knowledge base if and only if it is independently entailed by some partition.

2 Reducing $\mathcal{SHIQ}$(D) to Disjunctive Datalog

A $\mathcal{SHIQ}(\mathbf{D})$ [7] knowledge base $KB = (KB_{\mathcal{T}}, KB_{\mathcal{R}}, KB_{\mathcal{A}})$ consists of a TBox $KB_{\mathcal{T}}$, an RBox $KB_{\mathcal{R}}$ and an ABox $KB_{\mathcal{A}}$. $KB_{\mathcal{T}}$ is a finite set of *concept inclusion axioms*. $KB_{\mathcal{R}}$ is a finite set of *transitivity axioms* and *role inclusion axioms*. $KB_{\mathcal{A}}$ is a set of *concept and role membership assertions* $(\neg)A(a), R(a,b), T(a,c)$, and *(in)equality assertions* $a = b, a \neq b$, where A is an atomic concept, R an abstract role, T a concrete role, c a concrete individual, a and b abstract individuals.

In [7], a resolution framework is proposed to reduce a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB to a disjunctive datalog program $DD(KB) = \Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}}) \cup KB_{\mathcal{A}} \cup \Delta_{KB}$. $\Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}})$ is a positive disjunctive datalog program computed regardless of $KB_{\mathcal{A}}$ and consists of rules of the form

$$A_1 \vee \ldots \vee A_m \leftarrow B_1, \ldots, B_n \;\; (m \geq 0, n > 0) \;\; .$$

$A_1, \ldots, A_m, B_1, \ldots, B_n$ are all positive atoms which can be over the equality predicate $=$, (possibly inverse) roles, original atomic concepts or new atomic concepts introduced during the structural transformation. In addition, a body atom can also be a concrete domain atom, an atom of the form $HU(x)$ which makes the rule safe, or an atom of the form $S_f(x, x_f)$ which is introduced for eliminating the function symbol f. Δ_{KB} is made up of $HU(a)$, $HU(a_f)$ and $S_f(a, a_f)$, instantiated for each individual a and each function symbol f.

To enable equality reasoning in disjunctive datalog, the equality predicate $=$ is interpreted as a congruence relation and treated as an ordinary predicate, with required properties axiomatized explicitly [7, 8]. A disjunctive datalog program P with equality is thus transformed into a disjunctive datalog program $P_=$ without equality, by stating that $=$ is reflexive, symmetric, and transitive, and by appending replacement rules of the form "$R(x_1, \ldots, y_i, \ldots, x_n) \leftarrow R(x_1, \ldots, x_i, \ldots, x_n), x_i = y_i$", instantiated for each distinct predicate R and each position i. In what follows, we assume that $=$ has been treated as an ordinary predicate in $DD(KB)$, and with $\models_c$ we denote the cautious entailment in positive disjunctive datalog programs.

Theorem 1 ([7]). *For KB a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base, (1) KB is unsatisfiable if and only if $DD(KB)$ is unsatisfiable; (2) $KB \models \alpha$ if and only if $DD(KB) \models_c \alpha$ for each assertion α of the form $A(a)$ or $S(a,b)$, where A is an atomic concept and S a simple role.*

3 Partitioning the ABox

With the reduction, $DD(KB)$, of a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base KB, we can reduce the reasoning on KB to the reasoning on $DD(KB)$. In what follows, a ground atom is also called an *assertion*; an atom (or assertion) is called *basal* if it is over concrete domain predicates or the predicates in Δ_{KB}; an atom (or assertion) is called *normal* if it is not basal. A set S of assertions is said to *trigger* a rule R if $S = Body(R\sigma)$ for some ground substitution σ. A set S of assertions is said to *indirectly trigger* a rule R in logic program Π if there is a set S' of assertions such that $S \cup \Pi \models S'$ and S' triggers R. An assertion a is said to *participate in* the (indirect) triggering of R (in Π) if there is a set S of assertions such that $a \in S$ and S (indirectly) triggers R (in Π). If Π is clear from the context, it is omitted. Consider Π_0 in Example 1. We say $\{b, c, d\}$ triggers r_1, $\{e, c, d\}$ indirectly triggers r_1, and e participates in the indirect triggering of r_1. If the logic program contains rules with disjunctions, such as r_3 in Π_0, the indirect triggering is nondeterministic. We say a set S of assertions *might* indirectly trigger R in Π if S indirectly triggers R in a Horn logic program Π' which is converted from Π by replacing disjunctions with definite implications. Continue with Example 1, we say $\{b, g, d\}$ might indirectly trigger r_1 in Π_0.

Example 1. Let Π_0 be a logic program consisting of the following ground rules.

$$r_1 :\ a \leftarrow b, c, d\ . \qquad r_2 :\ b \leftarrow e\ . \qquad r_3 :\ c \vee f \leftarrow g\ .$$
$$r_4 : c\ . \qquad r_5 : d\ . \qquad r_6 : e\ . \qquad r_7 : g\ .$$

We intend to partition $KB_{\mathcal{A}}$ so that the subsequent reasoning on $DD(KB)$ can be performed separately on each of its partitions. So we should avoid communication between partitions during reasoning on $DD(KB)$. Consider Example 1. $\{c, d, e\}$ should be placed in the same partition, otherwise a cannot be independently entailed over any partition of $\{c, d, e, g\}$. This shows that sets of assertions that (might) indirectly trigger rules should be placed in the same partition. However, this intuition is too rough. Consider Example 1 again. $\{d, e, g\}$ might indirectly triggers r_1, but we need not put g to the same partition where $\{d, e\}$ locates, since $\{c, d, e\}$ need be placed in the same partition and then a can be independently entailed over $\{c, d, e\}$. The intuition behind such case is that for two sets of assertions S_1 and S_2, when $S_1 \cup S_2$ (might) indirectly trigger R in logic program Π, i.e., there exists S' such that $S_1 \cup S_2 \cup \Pi \models S'$ and S' triggers R, S_1 need not be placed in the same partition where S_2 locates, if $S_2 \cup \Pi \models S'$.

To exploit above intuitions, we first convert $DD(KB)$ to a plain datalog program $H(KB)$ by replacing disjunctions with conjunctions and adding constraints for negative atoms. That is, a rule of the form "$A_1 \vee \ldots \vee A_m \leftarrow B_1, \ldots, B_n (m > 0, n > 0)$" in $DD(KB)$ is converted to "$A_1 \wedge \ldots \wedge A_m \leftarrow B_1, \ldots, B_n$" and other rules in $DD(KB)$ remain. In addition, we treat negative atoms in $H(KB)$ as positive ones by adding constraints of the form "$\leftarrow a, \neg a$" to $H(KB)$ if $\neg a \in H(KB)$. We then separate $H(KB)$ into the definite part $H_1(KB)$, which consists of rules with heads, and the constraint part $H_0(KB)$, which consists of rules without heads, for different treatments.

Our partitioning algorithm is shown in Figure 1. Some tracking data are used. For each normal assertion a, we use $marked(a)$ to store whether a is marked. We *mark* a, i.e., set $marked(a)$ to true, if and only if a participates in triggering rules in $H(KB)$. Secondly, we use $parts(a)$ to store the set of identifiers of the parts where a locates, i.e., $parts(a) = \{id(p)|a \in p\}$. Besides we put a to the merged part when a participates in triggering rules, we might put a to a part p if there is some $b \in p$ *supporting* a through rule $r \in ground(H(KB)$, i.e., $a \in Head(r)$ and $b \in Body(r)$, according to the intuition that a set of assertions indirectly triggering rules should be placed in the same partition. However, as another intuition shows, when a is entailed over one of its parts, a need not be placed in the parts where its supporters locate. So $parts(a)$ is unchanged in such case. We can see that the set of *marked* assertions in $\bigcup_{id(p) \in parts(a)} p$ approximates the *support closure* of a. Thirdly, we use $entailed(a)$ to store whether a is entailed by $DD(KB)$, i.e., $DD(KB) \models_c a$. For efficiency, we use a simple recursive evaluation of $entailed(a)$, which is sound but incomplete. That is, if there exists a definite rule $r \in ground(DD(KB))$ such that $Head(r) = \{a\}$ and $entailed(b) = true$ for all $b \in Body(r)$, we set $entailed(a)$ to true.

There are some remarks on the merging procedure MergeParts. First, we merge parts instead of assertions for efficiency. Second, only when a participates in triggering rules can $marked(a)$ be set to true. After $marked(a)$ is set to true, $parts(a)$ remains a single set. Third, for all $a \in S_H$ with $entailed(a) = false$, $parts(a)$ is updated for enlarging the support closure of a: the support closures of all $b \in S_B$ (approximated with $id(p')$) are appended to the support closure of a, by adding $id(p')$ to $parts(a)$, and a to p' correspondingly.

Our proposed method will always produce a disjoint and independent partitioning of the ABox (Theorem 2), which ensures that a query over atomic concepts or simple roles can be performed separately on each generated partition and the results trivially combined yield complete answers. Another benefit of our method is the ability to filter unmarked assertions to the unique *unmarked partition* p_U (other partitions are called *marked partitions* correspondingly). Unmarked assertions do not participate in triggering rules in $DD(KB)$, and thus the reasoning over p_U can be performed on a fragment of $DD(KB)$ consisting of a kind of rules whose body has no normal atoms. This implies that reasoning over p_U is trivial.

Lemma 1. *Let KB be a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base such that $(KB_{\mathcal{T}}, KB_{\mathcal{R}}, \emptyset)$ is consistent, $KB_{\mathcal{A},1}, \dots, KB_{\mathcal{A},n}$ the parts returned by PartitionABox(KB). Then (1) $\{KB_{\mathcal{A},1}, \dots, KB_{\mathcal{A},n}\}$ is a disjoint partitioning of $KB_{\mathcal{A}}$; (2) $\bigcup_{i=1}^{n} M_i$ is a model of $DD(KB) = \Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}}) \cup KB_{\mathcal{A}} \cup \Delta_{KB}$ if for all $i = 1, \dots, n$, M_i is a minimal model of $DD(KB)_i = \Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}}) \cup KB_{\mathcal{A},i} \cup \Delta_{KB}$.*

Proof Sketch. (1) For all assertions $a \in KB_{\mathcal{A}}$, if a is marked in PartitionABox, it is placed in a unique part, otherwise it is moved to the unmarked partition. So $KB_{\mathcal{A},1}, \dots, KB_{\mathcal{A},n}$ is a disjoint partitioning of $KB_{\mathcal{A}}$.

(2) Let $\mathcal{C}$ be the set of all satisfiable basal assertions, $\mathcal{D} = \{a \in lfp(H_1(KB))| entailed(a) = true\}$ the set of all entailed normal assertions in the least fix-

```
**MergeParts**$(S_H, S_B, \mathcal{P})$
1.  **for** each $h \in S_H$ **with** $parts(h)$ undefined **do**
2.    $parts(h) := \emptyset$; $marked(h) := false$; $entailed(h) := false$;
3.  **if** $S_H = \{h\}$ **then** $entailed(h) := \bigwedge_{b \in S_B} entailed(b)$;
4.  $merge := \bigcup_{b \in S_B} parts(b) \cup \bigcup_{h \in S_H, marked(h)=true} parts(h)$;
5.  $p' := \bigcup_{id(p) \in merge} p$; $\mathcal{P} := \mathcal{P} \cup \{id(p')\} - merge$;
6.  **for** each $b \in S_B$ **do** $marked(b) := true$;
7.  **for** each $h \in S_H$ **with** $marked(h) = false$ **do**
8.    $parts(h) := parts(h) \cup \{id(p')\} - merge$; $p' := p' \cup \{h\}$;
9.  **for** each $a \in p'$ **do**
10.   **if** $marked(a) = true$ **then** $parts(a) := \{id(p')\}$;
11.   **else** $parts(a) := parts(a) \cup \{id(p')\} - merge$;

**PartitionABox**$(KB)$
**Input:** a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base $KB$ such that $(KB_{\mathcal{T}}, KB_{\mathcal{R}}, \emptyset)$ is consistent.
**Output:** the set of partitions of $KB_{\mathcal{A}}$.
1.  **for** each $a \in KB_{\mathcal{A}}$ **do**
2.    $p_a := \{a\}$; $parts(a) := \{id(p_a)\}$; $marked(a) := false$; $entailed(a) := true$;
3.  $\mathcal{P} := \bigcup_{a \in KB_{\mathcal{A}}} parts(a)$; Compute the least fixpoint $M_A$ of $H_1(KB)$;
4.  Meanwhile **for** each rule $r \in ground(H_1(KB))$ such that all concrete domain atoms
    of $Body(r)$ are satisfiable and all abstract domain atoms of $Body(r)$ are in $M_A$ **do**
5.    MergeParts$(\{h \in Head(r) | entailed(h) = false\}, \{b \in Body(r) | parts(b)$ is defined$\}, \mathcal{P})$;
6.  **for** each rule $r \in ground(H_0(KB))$ such that all concrete domain atoms of $Body(r)$
    are satisfiable and all abstract domain atoms of $Body(r)$ are in $M_A$ **do**
7.    MergeParts$(\emptyset, \{b \in Body(r) | parts(b)$ is defined$\}, \mathcal{P})$;
8.  $p_U := KB_{\mathcal{A}} \cap \bigcup_{id(p) \in \mathcal{P}} \{a \in p | marked(a) = false\}$;
9.  **for** each part $p$ such that $id(p) \in \mathcal{P}$ **do** $p := KB_{\mathcal{A}} \cap \{a \in p | marked(a) = true\}$;
10. **return** $\{p \neq \emptyset | id(p) \in \mathcal{P}\} \cup p_U$;
```

Fig. 1. An algorithm for partitioning the ABox

point of $H_1(KB)$, $p_1, \ldots, p_N$ all parts generated before line 8 in PartitionABox. W.l.o.g., we assume that $KB_{\mathcal{A},n}$ is the unmarked partition, and $KB_{\mathcal{A},i} = KB_{\mathcal{A}} \cap \{a \in p_i | marked(a) = true\}$ for all $i < n$. For all $i < n$, we show that M_i can be divided into layers $L_1, \ldots, L_m$ such that $L_1 = KB_{\mathcal{A},i} \cup (M_i \cap \mathcal{C})$ and each assertion in L_{k+1} is supported by a set of assertions in $\bigcup_{j=1}^{k} L_j$. Then, we show that $M_i \subseteq p_i \cup \mathcal{C} \cup \mathcal{D}$ by using induction on the layers of M_i, according to a fact that if there exists a rule $r \in ground(DD(KB))$ such that $a \in Head(r)$ and $Body(r) \subseteq p_i \cup \mathcal{C} \cup \mathcal{D}$, either $a \in \mathcal{D}$ or a is put to p_i in MergeParts. Now, suppose $M = \bigcup_{i=1}^{n} M_i$ is not a model of $DD(KB)$. There must be a rule $r \in ground(DD(KB))$ such that $Body(r) \subseteq M$ and $Head(r) \cap M = \emptyset$. Let $b_1, \ldots, b_m$ be all the normal assertions in $Body(r)$. Since M_n is a subset of the least fixpoint of the fragment of $DD(KB)$ consisting of a kind of rules whose body has no normal atoms, any b_i cannot be in M_n and thus is marked in PartitionABox. On the other hand, since different parts do not together participate in triggering rules, $b_1, \ldots, b_m$ are all in the same part, say p_k. Then, each b_i must be in $M_k \cup \mathcal{D}$, otherwise there is M_j $(j \neq k)$ such that $b_i \in M_j - \mathcal{C} - \mathcal{D} \subseteq p_j$, contradicting that b_i is marked. In case $b_i \in \mathcal{D}$, b_i is entailed over p_k and thus b_i is in every model of $DD(KB)_k$. This implies that each b_i is in M_k and thus $Hear(r) \cap M_k \neq \emptyset$, contradicting that $Head(r) \cap M = \emptyset$. $\square$

Theorem 2 (independent partitioning). *Let KB be a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base such that $(KB_{\mathcal{T}}, KB_{\mathcal{R}}, \emptyset)$ is consistent, $\{KB_{\mathcal{A},1}, \ldots, KB_{\mathcal{A},n}\}$ the disjoint partitioning of $KB_{\mathcal{A}}$ returned by PartitionABox(KB), and $KB_i = (KB_{\mathcal{T}}, KB_{\mathcal{R}}, KB_{\mathcal{A},i})$ for all $i = 1, \ldots, n$. Then (1) KB is consistent if and only if KB_i is consistent for all $i = 1, \ldots, n$; (2) $KB \models \alpha$ if and only if there exists KB_i such that $KB_i \models \alpha$ for each assertion α of the form $A(a)$ or $S(a, b)$, where A is an atomic concept and S a simple role.*

Proof. (1) The $(\Rightarrow)$ direction is trivial. For the $(\Leftarrow)$ direction, by Theorem 1, $DD(KB)_i = \Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}}) \cup KB_{\mathcal{A},i} \cup \Delta_{KB}$ is satisfiable for all $i = 1, \ldots, n$. $DD(KB)_i$ is positive and thus has minimal models. Let M_i be a minimal model of $DD(KB)_i$. By Lemma 1, $\bigcup_{i=1}^{n} M_i$ will be a model of $DD(KB)$. So KB is consistent by Theorem 1. (2) The $(\Leftarrow)$ direction is trivial. For the $(\Rightarrow)$ direction, we have $DD(KB) \models_c \alpha$ by Theorem 1. Suppose there is no KB_i such that $KB_i \models \alpha$. Let $DD(KB)_i = \Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}}) \cup KB_{\mathcal{A},i} \cup \Delta_{KB}$. By Theorem 1, there exist minimal models $M_1, \ldots, M_n$ of $DD(KB)_1, \ldots, DD(KB)_n$ respectively such that $\alpha \notin M_i$ for all $i = 1, \ldots, n$. By Lemma 1, $M = \bigcup_{i=1}^{n} M_i$ is a model of $DD(KB)$. That $\alpha \notin M$ contradicts that $DD(KB) \models_c \alpha$. $\square$

Regarding the complexity, we consider the *data complexity*, which is measured in $|KB_{\mathcal{A}}|$ only, under the assumption that $|KB_{\mathcal{TR}}| = |KB_{\mathcal{T}}| + |KB_{\mathcal{R}}|$ is bounded by a constant. In addition, we assume that there is a polynomial oracle for reasoning with concrete domains and a constant bound on the arity of the concrete domain predicates. Since the number of rules and the number of atoms in each rule in $\Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}})$, and the computation time of $\Gamma(KB_{\mathcal{T}}, KB_{\mathcal{R}})$ are all bounded by exponential of $|KB_{\mathcal{TR}}|$ [7], the number of different variables in each rule in $H(KB)$ is bounded by a constant and the number of rules in $ground(H(KB))$ is polynomial in $|KB_{\mathcal{A}}|$. Hence, PartitionABox runs in polynomial time in $|KB_{\mathcal{A}}|$.

4 Experimental Evaluation

We tested our partitioning method on top of the ontologies available on the KAON2 Web Site[3] and the Lehigh University Benchmark (LUBM) [6]. The implementation of the proposed method is based on secondary storage so as to handle large ABox data. Specifically, we used MySQL as the back-end DBMS. The input ABox data and the tracking data in run time are maintained in the database. We implemented the partitioning method in GNU C++, used the KAON2 system for the ontology reduction and performed testing on a 3.2GHz Pentium 4 CPU 2GB RAM machine running Windows XP.

[3] `http://kaon2.semanticweb.org/download/test_ontologies.zip`

Table 1. Test results on the partition time and granularity

Test Set	#assertions	Partition Time (hh:mm:ss)	#filtered (i.e., unmarked) assertions	#marked partitions	Avg. marked partition size (#assertions)	Max. marked partition size (#assertions)
Wine-0	496	00:02:29	78	43	9.72	336
Vicodi-0	53,653	00:05:58	0	53,653	1.00	1
Semintec-0	65,240	00:07:06	4,552	48,166	1.26	2
LUBM-1	100,543	00:03:02	22,418	45,931	1.70	2,190
LUBM-10	1,273 K	02:00:24	285,844	575,703	1.71	2,362

Table 1 shows the test results. In the table, all ontologies except Wine-0 are in a Horn fragment, i.e., their reductions are plain datalog programs. The partition granularity on these Horn-fragment ontologies is fine: the average size of marked partitions is very small, and the maximum size of marked partitions is so small that all partitions can be easily manipulated in physical memory. The ontology Wine-0 is rather complex, whose reduction is a disjunctive datalog program with more than 500 rules. The partition granularity on Wine-0 is not so fine as others, but still acceptable, since the submaximum size of marked partitions is 21 (without shown in Table 1) and the average size of marked partitions is small.

5 Related Work and Conclusion

Guo and Heflin [5] develop a set of tableau rules for partitioning the ABox in $\mathcal{SHIF}$ knowledge bases. Their partitioning method uses an intuition that assertions in the antecedent of an inference rule should be placed in the same partition. To estimate implicit inference in polynomial time, [5] uses some approximate tableau rules, such as $C_1 \sqsubseteq C_2$ for all concepts C_1 and C_2. To reduce partition size, [5] generates overlapped parts instead of partitions which need be disjoint. Though the overlapped parts preserve the independent partitioning property as ours, the performance of the subsequent reasoning may be impaired due to introducing many duplicated assertions. For example, with 1,311K input LUBM data, [5] generates 396,197 parts with average size 21.2 and maximum size 1,141. The number of duplicated assertions are about 7,000K. As a comparison, with 1,273K input data (LUBM-10), our method generates smaller partitions in average (see in Table 1). Though the largest marked partition generated by our method is near two times larger than the largest one in [5], no partitions generated by our method overlap.

Fokoue *et al.* [3] develop some role filtering techniques for partitioning the (summary) ABox in $\mathcal{SHIN}$ knowledge bases. Their partitioning method filters role assertions whose absence will not affect the outcome of a consistency check, and then places assertions sharing individual names in the same partition. Due to different focuses, the partitioning method in [3] is rather restricted, it filters role assertions only, while our proposed method filters all kinds of assertions. Moreover, the filtering techniques proposed in [3] do not guarantee a subsequent independent partitioning. As an example, consider a DL knowledge base KB, where $KB_{\mathcal{T}} = \{A \sqsubseteq \leq_1 R\}$, $KB_{\mathcal{R}} = \emptyset$ and $KB_{\mathcal{A}} = \{A(a), R(a,b), R(a,c), S(a,b)\}$. $S(a,b)$ will be filtered using the method in [3]. Then, any generated partition cannot entail $S(a,c)$ independently, while $S(a,c)$ can be entailed by KB.

Grau *et al.* [4] propose $\mathcal{E}$-Connections as a formalism for representing combinations of OWL knowledge bases and an algorithm for decomposing an OWL knowledge base into connected components. Amir and McIlraith [1] present a greedy algorithm for decomposing a first-order logic theory into partitions and message passing algorithms for reasoning with the partitioned theory. Due to different goals, the partitioning produced by either [4] or [1] may not have the independent partitioning property. This is because both [4] and [1] generate partitions with potential links and the subsequent reasoning may require communication between partitions through the links.

We have presented a method for partitioning the ABox of a $\mathcal{SHIQ}(\mathbf{D})$ knowledge base, based on a conversion from $\mathcal{SHIQ}(\mathbf{D})$ to plain datalog. The primary advantage of our method is that it always produces a disjoint and independent partitioning, while all existing methods in the related work may not. For future work, we will continue to improve the performance of our partitioning method, conduct more investigations on handling nominals and complex roles, and extend the method to an incremental one for dealing with knowledge base updates.

Acknowledgements

This work is supported in part by NSFC grants 60673103, 60421001 and 60373052.

References

1. E. Amir and S. A. McIlraith. Partition-based logical reasoning for first-order and propositional theories. *Artif. Intell.*, 162(1-2):49–88, 2005.
2. F. M. Donini. Complexity of reasoning. In *Description Logic Handbook*, pages 96–136. Cambridge University Press, 2003.
3. A. Fokoue, A. Kershenbaum, L. Ma, E. Schonberg, and K. Srinivas. The summary abox: Cutting ontologies down to size. In *ISWC-06*, pages 343–356, 2006.
4. B. C. Grau, B. Parsia, E. Sirin, and A. Kalyanpur. Automatic partitioning of owl ontologies using $\mathcal{E}$-connections. In *Description Logics*, 2005.
5. Y. Guo and J. Heflin. A scalable approach for partitioning owl knowledge bases. In *SSWS-06*, pages 47–60, 2006.
6. Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for owl knowledge base systems. *Journal of Web Semantics*, 3(2–3):158–182, 2005.
7. U. Hustadt, B. Motik, and U. Sattler. Reducing $\mathcal{SHIQ}^-$ description logic to disjunctive datalog programs. In *KR-04*, pages 152–162, 2004. (Extended version: Reasoning for Description Logics around $\mathcal{SHIQ}$ in a Resolution Framework. *FZI Technical Report* 3-8-04/04).
8. B. Motik, U. Sattler, and R. Studer. Query answering for owl-dl with rules. *Journal of Web Semantics*, 3(1):41–60, 2005.