Towards Practical ABox Abduction in Large OWL DL Ontologies

Jianfeng Du
Guangdong University of Foreign Studies, Guangzhou 510006, China
State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences

Guilin Qi
School of Computer Science and Engineering, Southeast University, NanJing 211189, China
State Key Laboratory for Novel Software Technology, Nanjing University, China

Yi-Dong Shen
State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, Beijing 100190, China

Jeff Z. Pan
Department of Computing Science, The University of Aberdeen, Aberdeen AB243UE, UK

Abstract

ABox abduction is an important aspect for abductive reasoning in Description Logics (DLs). It finds all minimal sets of ABox axioms that should be added to a background ontology to enforce entailment of a specified set of ABox axioms. As far as we know, by now there is only one ABox abduction method in expressive DLs computing abductive solutions with certain minimality. However, the method targets an ABox abduction problem that may have infinitely many abductive solutions and may not output an abductive solution in finite time. Hence, in this paper we propose a new ABox abduction problem which has only finitely many abductive solutions and also propose a novel method to solve it. The method reduces the original problem to an abduction problem in logic programming and solves it with Prolog engines. Experimental results show that the method is able to compute abductive solutions in benchmark OWL DL ontologies with large ABoxes.

Introduction

Since the W3C organization proposed the standard Web Ontology Language (OWL) embracing Description Logics (DLs) as the logical underpinning, abductive reasoning in DLs has become an attractive area. This area has been initiated by Elsenbroich, Kutz, and Sattler (2006) to support different application scenarios. In principle, every community interested in applying DLs to specific applications, such as *e*-Science, ontology engineering, medical informatics and computational linguistics, can easily extend the list of use cases for abduction over DL-based ontologies (Klarman, Endriss, and Schlobach 2011).

A DL-based ontology is composed of a TBox (storing intensional information) and an ABox (storing extensional information). ABox abduction is to find all minimal sets of ABox axioms that should be added to a background ontology to enforce entailment of an observation (which is usually a specified set of ABox axioms). This facility is crucial when one needs to understand why some extensional information cannot be entailed as expected and to acquire suggestions for enforcing entailment of the information.

As far as we know, by now there is only one ABox abduction method (Klarman, Endriss, and Schlobach 2011) in expressive DLs computing abductive solutions with certain minimality. The method works on the DL $\mathcal{ALC}$, which is a fragment of OWL DL — a species of the standard OWL corresponding to the DL $\mathcal{SHOIN}(\mathbf{D})$ (Horrocks, Patel-Schneider, and van Harmelen 2003). The problem addressed in the method allows abductive solutions to be expressed in the $\mathcal{ALE}$ fragment of $\mathcal{ALC}$, e.g., allows existential restrictions in solutions; thus the problem may have infinitely many solutions. Consider an ontology containing only the following axiom, which says that something has a person as its parent is a person.

$$\exists \mathsf{hasParent}.\mathsf{Person} \sqsubseteq \mathsf{Person}$$

There are infinitely many abductive solutions for the observation $\{\mathsf{Person}(\mathsf{Amy})\}$ (i.e. Amy is a person). Each abductive solution consists of a concept assertion of the form $\exists\mathsf{hasParent}.\exists\mathsf{hasParent}.\ldots.\mathsf{Person}(\mathsf{Amy})$, in which the individual is Amy and the concept is an existential restriction having arbitrarily many nested $\exists\mathsf{hasParent}$ and the tail Person. Since the method proposed by Klarman, Endriss, and Schlobach (2011) needs to verify the minimality of a potential solution after all potential solutions are computed, it may not output any abductive solution in finite time.

To explore more practical approaches to ABox abduction, we propose a new ABox abduction problem which is derived from traditional abduction problems in logic programming. Traditional abduction problems in logic programming find all minimal sets Δ of sentences w.r.t. a background theory T and an observation G, such that $T \cup \Delta$ entails G and $T \cup \Delta$ is consistent (Kakas, Kowalski, and Toni 1998). The proposed problem allows the background theory to be any DL-based ontology and the observation to be a set of ABox axioms, and restricts that the abductive solution contains only ABox axioms over a specified set of *abducibles*, which are *literal* (i.e. atomic or negated atomic) concepts or atomic roles. The introduction of abducibles gives users flexibility to formulate the explanations for an observation. Restricting abducibles to literal concepts or atomic roles ensures that the problem has finitely many solutions.

To seek methods to solve the proposed problem, we consider successful tools on abductive reasoning in logic programming, such as two state-of-the-art abduction systems CIFF (Mancarella et al. 2009) and $\mathcal{A}$-system (Kakas, Nuffe-

len, and Denecker 2001). These tools only allow the background theory to be a *normal logic program*, which corresponds to a plain datalog program extended with negation-as-failure. However, OWL DL is much more expressive than any DL fragment of normal logic programs. Hence, it is rather restrictive to directly apply state-of-the-art abduction systems to realize ABox abduction.

In this paper, we propose a new method for solving the proposed problem. In principle the method works with arbitrary DLs, but for simplicity we only describe the method with $\mathcal{SHOIQ}$ (i.e. OWL DL without datatypes but allowing qualifying number restrictions). The method consists of several steps, including normalizing the given problem, weakening a $\mathcal{SHOIQ}$ ontology (such as removing nominals), compiling a $\mathcal{SHIQ}$ (i.e. $\mathcal{SHOIQ}$ without nominals) ontology to a disjunctive datalog program (Hustadt, Motik, and Sattler 2007), compensating entailed rules, encoding a disjunctive datalog program to a Prolog program, and extracting abductive solutions from the output of the encoded Prolog program. We prove that the method guarantees the soundness of results, and it also guarantees the completeness when the observation has no complex role assertions and the background theory in the normalized problem is a Horn-$\mathcal{SHIQ}_{\not\leq}$ ontology without negated complex role assertions, where a Horn-$\mathcal{SHIQ}_{\not\leq}$ ontology is roughly a Horn-$\mathcal{SHIQ}$ ontology without maximum number restrictions or equality assertions, and a Horn-$\mathcal{SHIQ}$ ontology (Hustadt, Motik, and Sattler 2007) is a $\mathcal{SHIQ}$ ontology that can be translated to a First-order Horn logic program.

We conduct experiments on large benchmark ontologies. Experimental results show that the proposed method is able to compute abductive solutions for randomly generated observations in all test ontologies. This demonstrates that the proposed method paves a way towards practical ABox abduction in large OWL DL ontologies.

A New ABox Abduction Problem

We assume that the reader is familiar with OWL DL and its corresponding DL $\mathcal{SHOIQ}$. A $\mathcal{SHOIQ}$ ontology $\mathcal{O}$ is composed of a TBox and an ABox, both of which are sets of axioms. TBox axioms include *concept inclusion axioms* $C \sqsubseteq D$, *transitivity axioms* $\mathsf{Trans}(R)$ and *role inclusion axioms* $R \sqsubseteq S$, whereas ABox axioms include *concept assertions* $C(a)$, *role assertions* $R(a, b)$, *negated role assertions* $\neg R(a, b)$, *equality assertions* $a \approx b$ and *inequality assertions* $a \not\approx b$, where C and D are $\mathcal{SHOIQ}$ concepts, R and S roles, and a and b individuals. A role appearing in a qualifying number restriction should be *simple*, i.e., it has no transitive sub-roles. A role is *complex* iff it is not simple. $\mathcal{O}$ is treated as a set of axioms in the paper.

We use the traditional semantics for $\mathcal{SHOIQ}$ e.g. given in (Horrocks, Patel-Schneider, and van Harmelen 2003). By $\mathcal{M}(ax)$ (resp. $\mathcal{M}(S)$) we denote the set of models of an axiom ax (resp. a set S of axioms). Then $\mathcal{M}(S) = \bigcap_{ax \in S} \mathcal{M}(ax)$ for any set S of axioms. A $\mathcal{SHOIQ}$ ontology $\mathcal{O}$ is said to be *consistent* if $\mathcal{M}(\mathcal{O}) \neq \emptyset$. A set S of axioms is said to be *entailed by* $\mathcal{O}$, denoted by $\mathcal{O} \models S$, if $\mathcal{M}(\mathcal{O}) \subseteq \mathcal{M}(S)$.

In the area of abductive reasoning in logic programming, an abductive solution is often restricted in a pre-specified class of sentences called *abducibles*, so as to define modes to enforce entailment of an observation (Kakas, Kowalski, and Toni 1998). By exploring this idea, we propose a new ABox abduction problem. That is, given an ontology $\mathcal{O}$ as the background theory, a set A of abducibles which are literal concepts or atomic roles, and an observation G which is a set of concept assertions or atomic role assertions, compute all *abductive solutions* for $(\mathcal{O}, A, G)$, where an abductive solution for $(\mathcal{O}, A, G)$ is a (set-inclusion) minimal set Δ of ABox axioms such that all ABox axioms in Δ are directly composed of individuals in $\mathcal{O}$ and concepts/roles in A, $\Delta \not\models G$, $\mathcal{O} \cup \Delta \models G$ and $\mathcal{O} \cup \Delta$ is consistent. An example of the proposed problem is shown below.

Example 1. Let the background theory be an ontology $\mathcal{O}$ where its TBox consists of the following four axioms

$\exists \mathsf{hasIQ.High} \sqsubseteq \exists \mathsf{hasGrade.}(\mathsf{High} \sqcap \mathsf{Remarkable})$,
$\exists \mathsf{hasGrade.Remarkable} \sqsubseteq \mathsf{Good}$,
$\mathsf{Winner} \sqsubseteq \mathsf{Player}$, $\mathsf{Player} \sqsubseteq \mathsf{Winner} \sqcup \mathsf{Loser}$,

and its ABox consists of the following three axioms

$\mathsf{Player(Tom)}$, $\mathsf{hasIQ(Tom, A)}$, $\mathsf{hasGrade(Tom, A)}$.

Let the set of abducibles be $A = \{\mathsf{Remarkable}, \mathsf{High}, \mathsf{Good}\}$ and the observation be $G = \{\mathsf{Good(Tom)}\}$. Then there are two abductive solutions for $(\mathcal{O}, A, G)$, namely $\{\mathsf{High(A)}\}$ and $\{\mathsf{Remarkable(A)}\}$.

The proposed problem (simply called Problem A) mainly differs from the ABox abduction problem proposed by Klarman, Endriss, and Schlobach (2011) (simply called Problem B) in introducing abducibles and restricting them to literal concepts or atomic roles.

There may exist some drawbacks in Problem A. First, when some instance of Problem B has solutions, its counterpart in Problem A may not. For example, given $\mathcal{O} = \{\forall \mathsf{hasChild.Good} \sqsubseteq \mathsf{Happy}\}$ and $G = \{\mathsf{Happy(Amy)}\}$, the corresponding instance of Problem B has one abductive solution $\forall \mathsf{hasChild.Good(Amy)}$ in $\mathcal{O}$, but its counterpart in Problem A has not. Second, the representation of solutions in Problem A may be less concise than its counterpart in Problem B. For example, given $\mathcal{O} = \{\exists \mathsf{hasFather.}\top \sqsubseteq \mathsf{Person}, \neg\mathsf{hasFather}(a_1, a_1), \mathsf{Person}(a_2), ..., \mathsf{Person}(a_n)\}$, $A = \{\mathsf{hasFather}\}$ and $G = \{\mathsf{Person}(a_1)\}$, the corresponding instance of Problem A has $n - 1$ abductive solutions $\{\mathsf{hasFather}(a_1, a_2)\}$, ..., $\{\mathsf{hasFather}(a_1, a_n)\}$ in $\mathcal{O}$, while its counterpart in Problem B has only one, namely $\{\exists \mathsf{hasFather.}\top(a_1)\}$.

However, there exist some important merits in Problem A. First, the number of abductive solutions is finite because the number of possible axioms in an abductive solution is finite. Second, the minimality of potential solutions can be simply determined by set-inclusion checking, rather than by the complex renaming and entailment checking as in Problem B. Since termination and efficiency are crucial for practical ABox abduction, our work focuses on Problem A.

Computing All Abductive Solutions

Although the number of abductive solutions is finite for the proposed problem, a search-based method is imprac-

tical because the search space is very large. Considering that the state-of-the-art abduction systems CIFF (Mancarella et al. 2009) and $\mathcal{A}$-system (Kakas, Nuffelen, and Denecker 2001) work on plain datalog programs with negation-as-failure, we seek to adapt the methods for abductive reasoning in plain datalog programs to ABox abduction. To obtain an adaptation without restricting the background theory to DL fragments of plain datalog programs, we consider the KAON2 transformation (Hustadt, Motik, and Sattler 2007), which has been implemented in the KAON2 system (`http://kaon2.semanticweb.org/`).

Given an *extensionally reduced* $\mathcal{SHIQ}$ ontology $\mathcal{O}$, i.e., all concept assertions in $\mathcal{O}$ are over literal concepts, the KAON2 transformation computes a disjunctive datalog program with equality, denoted by $\mathsf{DD}(\mathcal{O})$, which is the union of a set of function-free rules compiled from the TBox of $\mathcal{O}$ by exploiting certain resolution operations and a set of ground rules directly translated from the ABox of $\mathcal{O}$. The main properties of $\mathsf{DD}(\mathcal{O})$ are shown below.

Theorem 1 ((Hustadt, Motik, and Sattler 2007))**.** *Let $\mathcal{O}$ be an extensionally reduced $\mathcal{SHIQ}$ ontology. Then: (1) for any literal concept/role assertion ax, $\mathsf{DD}(\mathcal{O} \cup \{ax\}) = \mathsf{DD}(\mathcal{O}) \cup \{ax\}$; (2) when $\mathcal{O}$ has no negated complex role assertions, $\mathcal{O}$ is consistent iff $\mathsf{DD}(\mathcal{O})$ is satisfiable.*

In the following, an example for the result of the KAON2 transformation is shown.

Example 2. Consider ontology $\mathcal{O}$ given in Example 1. By compiling $\mathcal{O}$ through the KAON2 system, we obtain $\mathsf{DD}(\mathcal{O})$ which consists of the following rules (1)–(7).

$$\mathsf{Good}(x) \leftarrow \mathsf{hasGrade}(x, y), \mathsf{Remarkable}(y). \quad (1)$$
$$\mathsf{Good}(x) \leftarrow \mathsf{hasIQ}(x, y), \mathsf{High}(y). \quad (2)$$
$$\mathsf{Player}(x) \leftarrow \mathsf{Winner}(x). \quad (3)$$
$$\mathsf{Loser}(x) \vee \mathsf{Winner}(x) \leftarrow \mathsf{Player}(x). \quad (4)$$
$$\mathsf{Player}(\mathsf{Tom}) \leftarrow . \quad (5)$$
$$\mathsf{hasIQ}(\mathsf{Tom}, \mathsf{A}) \leftarrow . \quad (6)$$
$$\mathsf{hasGrade}(\mathsf{Tom}, \mathsf{A}) \leftarrow . \quad (7)$$

We also define an *abductive solution* for $(\mathsf{DD}(\mathcal{O}), A, G)$ as a minimal set of Δ of ABox axioms such that all ABox axioms in Δ are directly composed of individuals in $\mathcal{O}$ and concepts/roles in A, $\Delta \not\models G$, $\mathsf{DD}(\mathcal{O}) \cup \Delta \models G$ and $\mathsf{DD}(\mathcal{O}) \cup \Delta$ is consistent. By Theorem 1, we have a correspondence between abductive solutions for $(\mathsf{DD}(\mathcal{O}), A, G)$ and abductive solutions for $(\mathcal{O}, A, G)$.

Theorem 2. *Let $\mathcal{O}$ be an extensionally reduced $\mathcal{SHIQ}$ ontology without negated complex role assertions, A a set of literal concepts or atomic roles, and G a set of literal concept assertions or atomic simple role assertions. For any set Δ of ABox axioms, Δ is an abductive solution for $(\mathsf{DD}(\mathcal{O}), A, G)$ iff it is an abductive solution for $(\mathcal{O}, A, G)$.*

Based on the above theorem, we first propose a restrictive method for some restrictive class that allows a reduction from the proposed problem to an abduction problem in disjunctive datalog, then enhance it to address the full class.

The Restrictive Method

In this subsection, let $\mathcal{O}$ be an extensionally reduced $\mathcal{SHIQ}$ ontology without negated complex role assertions, A be a set of literal concepts or atomic roles, and G be a set of literal concept assertions or atomic simple role assertions. According to Theorem 2, we seek methods for computing abductive solutions for $(\mathsf{DD}(\mathcal{O}), A, G)$.

Considering that most state-of-the-art abduction systems are built on Prolog engines that work on plain datalog programs, we seek to encode $(\mathsf{DD}(\mathcal{O}), A, G)$ into a Prolog program and solve it with Prolog engines. Since CIFF and $\mathcal{A}$-system currently do not guarantee termination in handling cyclic logic programs, but $\mathsf{DD}(\mathcal{O})$ can have cycles between predicates, we do not apply CIFF or $\mathcal{A}$-system to compute solutions for $(\mathsf{DD}(\mathcal{O}), A, G)$. Instead, we apply a modern Prolog engine, B-Prolog (`http://www.probp.com/`), which exploits a linear tabling mechanism (Shen et al. 2001) to handle cycles and guarantee termination.

Note that, when the equality predicate $\approx$ occurs in some rule heads in $\mathsf{DD}(\mathcal{O})$, $\approx$ should be interpreted as a congruence relation. In order to ensure B-Prolog to correctly handle $\approx$, the equality should be axiomatized by adding some standard rules e.g. given in (Fitting 1996). Let $\mathsf{DD}'(\mathcal{O})$ be obtained from $\mathsf{DD}(\mathcal{O})$ by adding rules to axiomatize equality if $\approx$ occurs in some rule heads in $\mathsf{DD}(\mathcal{O})$, or be $\mathsf{DD}(\mathcal{O})$ otherwise. Consider Example 2, since $\approx$ does not occurs in any rule head in $\mathsf{DD}(\mathcal{O})$, $\mathsf{DD}'(\mathcal{O})$ is the same as $\mathsf{DD}(\mathcal{O})$.

We briefly introduce the method for encoding $(\mathsf{DD}'(\mathcal{O}), A, G)$ into a Prolog program, which has four steps.

In the first step, all predicates P occurring in cycles in $\mathsf{DD}'(\mathcal{O})$ are declared to be tabled. This means that the same subgoal over P is prevented from executing multiple times and computed facts over P are cached. Consider Example 2, Player and Winner are in cycles, so they are declared to be tabled. Moreover, the goal predicate of the resulting Prolog program is also declared to be tabled for pruning explicit non-solutions. In the second step, atomic concept/role assertions in the observation G are encoded into Prolog rules. For example, the observation $\{\mathsf{Good}(\mathsf{Tom}), \mathsf{Winner}(\mathsf{Tom})\}$ is encoded into the following two Prolog rules, where `S` is an output parameter for storing solutions, and `check(S)` returns true iff there is not any cached `goal(S')` such that `S'` $\subseteq$ `S`.

```
go :- goal(S), write(S), fail.
goal(S) :- a_Good('Tom', S1), check(S1),
        a_Winner('Tom', S2), append(S1, S2, S), check(S).
```

In the third step, every *definite* rule (i.e. rule having a single head atom) in $\mathsf{DD}'(\mathcal{O})$ is encoded into a Prolog rule. Consider Example 2, all rules except rule (4) are encoded. E.g., rules (2) and (6) are encoded into the following rules.

```
a_Good(X, S) :- a_hasIQ(X, Y, S1), check(S1),
            a_High(Y, S2), append(S1, S2, S), check(S).
a_hasIQ('Tom', 'A', []).
```

In the last step, every atomic concept/role in A is encoded into a Prolog rule. For example, the abducibles High and hasIQ are encoded into the following two rules, where `ground(X)` returns true iff `X` is a constant term.

```
a_High(X, [(X, 'rdf:type', 'High')]) :- ground(X).
a_hasIQ(X, Y, [(X, 'hasIQ', Y)]) :- ground(X), ground(Y).
```

Let $\mathsf{prolog}(\mathsf{DD}'(\mathcal{O}), A, G)$ denote the set of lists output by the Prolog program encoded from $(\mathsf{DD}'(\mathcal{O}), A, G)$ when executing `go`. Let $\mathsf{decode}(L)$ denote the set of concept/role assertions decoded from a list L (treated as a set) by rewriting every element of L of the form $(a, \texttt{'rdf:type'}, C)$ to

$C(a)$ and every element of L of the form (a, R, b) to $R(a, b)$.

The following theorem shows the conditions where all abductive solutions for $(\mathsf{DD}(\mathcal{O}), A, G)$ can be extracted from $\mathsf{prolog}(\mathsf{DD}'(\mathcal{O}), A, G)$.

Theorem 3. *If $\mathsf{DD}(\mathcal{O})$ is a plain datalog program (possibly with equality), A is a set of atomic concepts/roles and G is a set of atomic concept/role assertions, then the set of abductive solutions for $(\mathsf{DD}(\mathcal{O}), A, G)$ is the set of minimal sets in $\{\mathsf{decode}(L) \mid L \in \mathsf{prolog}(\mathsf{DD}'(\mathcal{O}), A, G), \mathsf{decode}(L) \not\models G, \mathcal{O} \cup \mathsf{decode}(L)$ is consistent$\}$.*

According to the above theorem, the proposed restrictive method is to compute $\{\mathsf{decode}(L) \mid L \in \mathsf{prolog}(\mathsf{DD}'(\mathcal{O}), A, G), \mathsf{decode}(L) \not\models G, \mathcal{O} \cup \mathsf{decode}(L)$ is consistent$\}$. However, it only guarantees soundness and completeness of results in a restrictive class of the proposed problem. Moreover, it is impractical when $\mathsf{DD}(\mathcal{O})$ has equational head atoms, because in this case $\mathsf{DD}'(\mathcal{O})$ will have some rules like $P(y) \leftarrow P(x), x \approx y$ for axiomatizing equality. These rules are hard to be handled by the encoded Prolog program, as shown in our experiments, because every predicate occurring in these rules appears in cycles. Hence, we propose a general method to address all these issues.

The General Method

To apply Prolog engines to compute abductive solutions for $(\mathcal{O}, A, G)$, we need to reduce the original problem to a problem of computing abductive solutions for some $(\mathcal{P}, A', G')$, where $\mathcal{P}$ is a plain datalog program, A' is a set of atomic concepts/roles, G' is a set of atomic concept/role assertions. Suppose there is a one-to-one mapping function f from symbols in A' to symbols in A. For a set Δ of ABox axioms over atomic concepts/roles in A', let $f(\Delta)$ simply denote a set of ABox axioms obtained from Δ by replacing all symbols X in A' with $f(X)$. We aim to ensure that $\mathcal{P} \cup \Delta \models G'$ implies $\mathcal{O} \cup f(\Delta) \models G$ for all sets Δ of ABox axioms over atomic concepts/roles in A', because $\mathcal{O} \cup f(\Delta) \models G$ implies that some subsets of $f(\Delta)$ can be abductive solutions. We propose a general method to compute abductive solutions for $(\mathcal{O}, A, G)$, which is based on constructing the aforementioned $(\mathcal{P}, A', G')$ and has five steps.

Step 1. The first step is to normalize $(\mathcal{O}, A, G)$. First, $(\mathcal{O}, A, G)$ is converted to $(\mathcal{O}^\dagger, A', G')$ by renaming non-atomic concepts. That is, A' is obtained from A by replacing every negated atomic concept $\neg X$ with a fresh atomic concept $X^\dagger$; G' is obtained from G by replacing every non-atomic concept assertion $C(a)$ with $Q_C(a)$, where Q_C is a fresh atomic concept for C; $\mathcal{O}^\dagger$ is obtained from $\mathcal{O}$ by adding axioms $\neg X \sqsubseteq X^\dagger$ and $X^\dagger \sqsubseteq \neg X$ for every negated atomic concept $\neg X$ in A, and by adding axioms $C \sqsubseteq Q_C$ and $Q_C \sqsubseteq C$ for every non-atomic concept assertion $C(a)$ in G. Afterwards, $\mathcal{O}^\dagger$ is extensionally reduced to $\mathcal{O}'$. That is, $\mathcal{O}'$ is obtained from $\mathcal{O}^\dagger$ by replacing $C(a)$ with two axioms $Q_C(a)$ and $Q_C \sqsubseteq C$ for every non-literal concept assertion $C(a)$ in $\mathcal{O}^\dagger$.

Example 3. *Consider computing all abductive solutions for $(\mathcal{O}, A, G)$, where $\mathcal{O}$ is the ontology given in Example 1, $A = \{\neg\mathsf{Loser}\}$ and $G = \{\mathsf{Winner}(\mathsf{Tom})\}$. $(\mathcal{O}, A, G)$ is normalized to $(\mathcal{O}', A', G')$, where $A' = \{\mathsf{Loser}^\dagger\}$, $G' = G$ and $\mathcal{O}' = \mathcal{O} \cup \{\neg\mathsf{Loser} \sqsubseteq \mathsf{Loser}^\dagger, \mathsf{Loser}^\dagger \sqsubseteq \neg\mathsf{Loser}\}$.*

Let f be a one-to-one mapping function on all symbols $X \in A'$ such that $f(X) = \neg Y$ if X is of the form $Y^\dagger$, or $f(X) = X$ otherwise. We have the following lemma.

Lemma 1. *For any set Δ of ABox axioms over atomic concepts/roles in A', Δ is an abductive solution for $(\mathcal{O}', A', G')$ iff $f(\Delta)$ is an abductive solution for $(\mathcal{O}, A, G)$.*

Step 2. The second step is to weaken $\mathcal{O}'$ so as to apply the KAON2 transformation. Let $\mathcal{O}^\ddagger$ be obtained from $\mathcal{O}'$ by replacing every nominal $\{a\}$ with a fresh atomic concept C_a and adding a concept assertion $C_a(a)$. Since $\mathsf{DD}(\mathcal{O}^\ddagger)$ may contain equational head atoms, which introduces the necessity for axiomatizing equality in $\mathsf{DD}(\mathcal{O}^\ddagger)$, $\mathcal{O}^\ddagger$ should be weakened further. Let $\mathsf{NNF}(E)$ denote the *negation normal form* of a concept E, which can be computed by standard methods e.g. given in (Hustadt, Motik, and Sattler 2007), and $\mathsf{norm}(\mathcal{O}^\ddagger)$ denote the ontology normalized from $\mathcal{O}^\ddagger$ by replacing every concept inclusion axiom $C \sqsubseteq D$ with $\top \sqsubseteq \mathsf{NNF}(\neg C \sqcup D)$. Let $\mathcal{O}''$ be obtained from $\mathsf{norm}(\mathcal{O}^\ddagger)$ by removing all equality assertions and by replacing $\leq_n R.C$ with $\top$ for every $\leq_n R.C$ occurring in the right hand sides of concept inclusion axioms in $\mathsf{norm}(\mathcal{O}^\ddagger)$. Then $\mathsf{DD}(\mathcal{O}'')$ does not contain any equational head atom.

We define a $\mathcal{SHIQ}_{\not\leq}$ ontology $\mathcal{O}$ as a $\mathcal{SHIQ}$ ontology such that $\mathsf{norm}(\mathcal{O})$ has not any equality assertion and does not contain any maximum number restriction $\leq_n R.C$ in the right hand sides of concept inclusion axioms. Obviously $\mathcal{O}''$ is a $\mathcal{SHIQ}_{\not\leq}$ ontology. We have the following lemma.

Lemma 2. *For any set Δ of ABox axioms over atomic concepts/roles in A', $\mathcal{O}' \cup \Delta \models G'$ if $\mathcal{O}'' \cup \Delta \models G'$.*

Step 3. The third step is to modify $\mathsf{DD}(\mathcal{O}'')$. Since redundant rules that do not affect the results of the subsequent resolution operations have been eliminated in the KAON2 transformation, $\mathsf{DD}(\mathcal{O}'')$ may not contain all entailed definite rules. The absence of some redundant definite rules may make the later encoded Prolog program have no results, as shown in the following example.

Example 4. Consider the normalized problem $(\mathcal{O}', A', G')$ given in Example 3. The step for weakening $\mathcal{O}'$ yields a semantically equivalent ontology $\mathcal{O}''$ since $\mathcal{O}'$ is already a $\mathcal{SHIQ}_{\not\leq}$ ontology. By compiling $\mathcal{O}''$ through the KAON2 system, we obtain $\mathsf{DD}(\mathcal{O}'')$ which consists of the rules (1)–(7) given in Example 2 and the following two rules.

$$\mathsf{Loser}(x) \vee \mathsf{Loser}^\dagger(x) \leftarrow . \quad (8)$$

$$\leftarrow \mathsf{Loser}(x), \mathsf{Loser}^\dagger(x). \quad (9)$$

The predicate Winner does not occur in heads of definite rules in $\mathsf{DD}(\mathcal{O}'')$, thus $\mathsf{prolog}(\mathsf{DD}(\mathcal{O}''), A', G')$ is empty.

We introduce two simple steps to compensate definite rules. In step one, for every rule R in $\mathsf{DD}(\mathcal{O}'')$, all head atoms $P(x)$, such that there is an atomic concept P' such that $\top \sqsubseteq P \sqcup P' \in \mathcal{O}''$ and $\top \sqsubseteq \neg P \sqcup \neg P' \in \mathcal{O}''$, are rewritten to $P'(x)$ and moved to the body of R, yielding a new rule R' added to $\mathsf{DD}(\mathcal{O}'')$. In step two, for every constraint R in $\mathsf{DD}(\mathcal{O}'')$ and every body atom $P(x)$ of R, such that there is an atomic concept P' such that $\top \sqsubseteq P \sqcup P' \in \mathcal{O}''$, $\top \sqsubseteq \neg P \sqcup \neg P' \in \mathcal{O}''$ and R is not of the form "$\leftarrow P(x), P'(x)$", $P(x)$ is written to $P'(x)$ and moved to the head of R, yielding a new rule R' added

to $\mathsf{DD}(O'')$. Consider Example 4, the following rule (10) is added to $\mathsf{DD}(\mathcal{O}'')$ in the aforementioned two steps.

$$\mathsf{Winner}(x) \leftarrow \mathsf{Player}(x), \mathsf{Loser}^{\dagger}(x). \quad (10)$$

We can see that, now the Prolog program encoded from ($\mathsf{DD}(\mathcal{O}'')$, A', G') will output a list [('Tom', 'rdf:type', 'Loser† ')] when executing go.

Although more definite rules can be obtained by other resolution operations, we only compensate definite rules by simple operations for the consideration of efficiency.

Let $\mathcal{P}$ denote the plain datalog program obtained from $\mathsf{DD}(\mathcal{O}'')$ by keeping only definite rules. The following lemma shows the relationship between $\mathcal{P}$ and $\mathcal{O}'$.

Lemma 3. *For any set Δ of ABox axioms over atomic concepts/roles in A', $\mathcal{O}' \cup \Delta \models G'$ if $\mathcal{P} \cup \Delta \models G'$.*

Step 4. The fourth step is to encode a Prolog program from ($\mathcal{P}$, A', G') using the encoding method given in the previous subsection and then execute go.

Step 5. The last step is to extract abductive solutions from $\mathsf{prolog}(\mathcal{P}, A'\ G')$. Consider an arbitrary set Δ' of ABox axioms over concepts/roles in A such that $\Delta' \not\models G$. It can be seen that, all minimal subsets Δ of Δ' such that $\mathcal{O} \cup \Delta$ is consistent and entails G are abductive solutions for ($\mathcal{O}$, A, G). However, it is probable that these subsets of Δ' do not exist when $\mathcal{O} \cup \Delta' \not\models G$. In contrast, consider a list $L \in \mathsf{prolog}(\mathcal{P}, A'\ G')$, since $\mathcal{P} \cup \mathsf{decode}(L) \models G'$, by Lemma 3, $\mathcal{O}' \cup \mathsf{decode}(L) \models G'$ and thus $\mathcal{O} \cup f(\mathsf{decode}(L)) \models G$. Hence, we do not extract abductive solutions from arbitrary hypotheses but only from lists L in $\mathsf{prolog}(\mathcal{P}, A'\ G')$ such that $\mathsf{decode}(L) \not\models G'$. The following theorem shows that this method guarantees the soundness of results.

Theorem 4. *Let L be a list in $\mathsf{prolog}(\mathcal{P}, A'\ G')$ such that $\mathsf{decode}(L) \not\models G'$, and Δ be a minimal subset of $f(\mathsf{decode}(L))$ such that $\mathcal{O} \cup \Delta$ is consistent and entails G, then Δ is an abductive solution for ($\mathcal{O}$, A, G).*

This method also guarantees the completeness of results in some restrictive class of the proposed problem.

Theorem 5. *If $\mathcal{O}'$ is a Horn-$\mathcal{SHIQ}_{\leq}$ ontology without negated complex role assertions and G has no complex role assertions, then every abductive solution Δ for ($\mathcal{O}$, A, G) is a minimal subset of $f(\mathsf{decode}(L))$ such that $\mathcal{O} \cup \Delta$ is consistent and entails G for some list L in $\mathsf{prolog}(\mathcal{P}, A'\ G')$ such that $\mathsf{decode}(L) \not\models G'$.*

Experimental Evaluation

We conducted experiments on thirteen benchmark ontologies with large ABoxes. The first two ontologies are Semintec (about financial services) and Vicodi (about European history). The next five are the Lehigh University Benchmark (LUBM) (Guo, Pan, and Heflin 2005) ontologies LUBMn (n = 1, ..., 5), where LUBMn denotes the LUBM ontology containing the data of n universities. The above ontologies were previously used to compare different DL reasoners (Motik and Sattler 2006). The last six ontologies are the University Benchmark (UOBM) (Ma et al. 2006) ontologies UOBM-Lite$n$ and UOBM-DLn ($n = 1, 2, 3$). We could not test larger UOBM ontologies that involve more than three universities, because B-Prolog ran out of memory when loading the Prolog programs encoded from these ontologies. The characteristics of all test ontologies are shown in Table 1. All experiments were done on a PC with Pentium Dual Core 2.60GHz CPU and 2GB RAM, running Windows XP, where the maximum Java heap size was set to 1GB.

Table 1: The characteristics of test ontologies

Ontology	#C	#R	#TA	#AA	#I
Semintec	60	16	219	65,240	17,941
Vicodi	194	12	223	116,181	33,238
LUBM1∼5	43	32	93	100,543∼624,532	17,174∼102,368
UOBM-Lite1∼3	51	43	145	245,740∼575,380	37,704∼71,901
UOBM-DL1∼3	69	44	206	260,900∼607,248	37,927∼72,059

Note: "#C" / "#R" / "#TA" / "#AA" / "#I" is the number of atomic concepts/atomic roles/TBox axioms/ABox axioms/individuals.

We first compared the general method with the restrictive method on handling test ontologies for which the results of the KAON2 transformation are plain datalog programs with equality. These ontologies include Semintec and all UOBM-Lite$n$ ontologies. We randomly generated atomic concept assertions and treated every singleton set made up of a generated concept assertion as an observation and all atomic concepts as abducibles. The general method successfully handles all observations that have abductive solutions in one hour, but the restrictive method always exceeds 12 hours when handling any of these observations. This result shows that the rules added to axiomatize equality are very harmful to the efficiency of ABox abduction, because these rules introduce cycles in the encoded Prolog program.

We then focused on the general method. For each test ontology, we randomly generated 40 concept assertions not entailed by the ontology, out of which twenty are over atomic concepts and the other twenty over negated atomic concepts. For all test ontologies that have the same TBox (such as LUBM), we generated the same set of concept assertions. We performed two experiments for each test ontology. In the first experiment we treated all atomic concepts as abducibles, whereas in the second one we treated all literal concepts as abducibles. Moreover, in both experiments we treated every singleton set made up of a generated concept assertion as an observation. In each experiment, the implemented system works in two phases. The first phase (i.e. the preprocess phase) compiles the ontology and all observations to a Prolog program and loads it to B-Prolog. The second phase (i.e. the query phase) handles every observation one by one for computing abductive solutions.

Some test results are shown in Table 2 due to the space limitation. For Semintec and Vicodi, all observations are *handled successfully* (i.e. their abductive solutions are computed in one hour) in both experiments. For other ontologies, all observations are handled successfully in the first experiment. In the second experiment, all (resp. 29 and 16) observations are handled successfully for each LUBMn (resp. UOBM-Lite$n$ and UOBM-DLn) ontology. All failure cases are caused by running out of memory when B-Prolog executes the encoded Prolog program. It shows that the bottleneck of the method lies in executing the encoded Prolog program, which is time and memory consuming.

Table 2: The execution time (sec) for some test ontologies

	The First Experiment			The Second Experiment		
Ontology	Preprocess	Max	Avg	Preprocess	Max	Avg
Semintec	21.4	47.5	6.1	21.7	47.9	7.3
Vicodi	70.1	20.2	3.6	70.8	20.6	9.9
LUBM1	35.5	3.5	0.3	35.7	3.6	1.0
LUBM5	947.9	29.3	2.4	949.2	59.8	11.1
UOBM-Lite3	217.1	71.1	7.0	224.9	71.1	11.8
UOBM-DL3	227.2	91.3	6.9	230.2	113.3	15.8

Note: "Max" / "Avg" is the maximum/average execution time for successfully handling an observation in the query phase.

Related Work

By now there are only few works addressing abductive reasoning in DLs. Most of these works focus on TBox abduction, including some based on automata (Hubauer, Lamparter, and Pirker 2010) and others exploiting tableaux-based algorithms (Noia, Sciascio, and Donini 2007; 2009). Although an ABox axiom can be rewritten to a TBox axiom (e.g. $C(a)$ can be rewritten to $\{a\} \sqsubseteq C$), the rewriting results contain nominals and cannot be handled by the above methods. Moreover, ABox abduction mainly differs from TBox abduction in allowing multiple individuals to appear in an abductive solution. Therefore it is very hard to adapt methods for TBox abduction to ABox abduction.

Some other works focus on ABox abduction, including one considering a DL-based ontology accompanying rules (Peraldi et al. 2007) and one exploiting tableaux and resolution algorithms (Klarman, Endriss, and Schlobach 2011). The abductive reasoning method presented in (Peraldi et al. 2007) is a backward inference method. It restricts axioms in the DL-based ontology to some special forms and does not use a notion of minimality for abductive solutions. The abductive reasoning method proposed in (Klarman, Endriss, and Schlobach 2011), as described in the first section, focuses on a fragment of OWL DL and a problem framework that cannot guarantee termination. In contrast, our proposed method guarantees termination and certain minimality of results. It also allows the background ontology to be expressed in arbitrary DLs. We are currently unable to empirically compare with the above two methods because for the first one, neither the ontology nor the system they used is publicly accessible, while for the second one, no evaluation results are available.

Conclusions and Future Work

This paper proposed an ABox abduction method which is based on reducing the original problem to an abduction problem in logic programming. By introducing abducibles and restricting them to literal concepts or atomic roles, the method guarantees termination. Since the reduction is approximate, the proposed method does not guarantee the completeness of results in some cases, but it always guarantees the soundness. Experimental results showed the feasibility of the method in computing abductive solutions in large ontologies with up to half a million ABox axioms.

Experimental results showed that the bottleneck of our method lies in solving the reduced abduction problem, so in future work we will investigate what fragments of plain datalog allow for efficient computation of abductive solutions. We also plan to refine the proposed ABox abduction problem to address the issue that allowing roles as abducibles may probably result in too many abductive solutions.

Acknowledgements. This work is partly supported by the NSFC grants (61005043, 61003157, 60970045, 60833001), the Jiangsu Science Foundation (BK2010412), the Key Laboratory of Computer Network and Information Integration (Southeast University), Ministry of Education, and the ITA and K-Drive projects.

References

Elsenbroich, C.; Kutz, O.; and Sattler, U. 2006. A case for abductive reasoning over ontologies. In *Proc. of OWLED'06 Workshop*.

Fitting, M. 1996. *First-order Logic and Automated Theorem Proving (2nd ed.)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Guo, Y.; Pan, Z.; and Heflin, J. 2005. LUBM: A benchmark for OWL knowledge base systems. *Journal of Web Semantics* 3(2–3):158–182.

Horrocks, I.; Patel-Schneider, P. F.; and van Harmelen, F. 2003. From $\mathcal{SHIQ}$ and RDF to OWL: the making of a web ontology language. *Journal of Web Semantics* 1(1):7–26.

Hubauer, T.; Lamparter, S.; and Pirker, M. 2010. Automata-based abduction for tractable diagnosis. In *Proc. of DL'10 Workshop*.

Hustadt, U.; Motik, B.; and Sattler, U. 2007. Reasoning in description logics by a reduction to disjunctive datalog. *Journal of Automated Reasoning* 39(3):351–384.

Kakas, A. C.; Kowalski, R. A.; and Toni, F. 1998. *The Role of Abduction in Logic Programming*. Oxford University Press. chapter 5, 235–324.

Kakas, A. C.; Nuffelen, B. V.; and Denecker, M. 2001. $\mathcal{A}$-System: Problem solving through abduction. In *Proc. of IJCAI'01*, 591–596.

Klarman, S.; Endriss, U.; and Schlobach, S. 2011. Abox abduction in the description logic $\mathcal{ALC}$. *Journal of Automated Reasoning* 46(1):43–80.

Ma, L.; Yang, Y.; Qiu, Z.; Xie, G.; Pan, Y.; and Liu, S. 2006. Towards a complete OWL ontology benchmark. In *Proc. of ESWC'06*, 125–139.

Mancarella, P.; Terreni, G.; Sadri, F.; Toni, F.; and Endriss, U. 2009. The ciff proof procedure for abductive logic programming with constraints: Theory, implementation and experiments. *Theory and Practice of Logic Programming* 9(6):691–750.

Motik, B., and Sattler, U. 2006. A comparison of reasoning techniques for querying large description logic aboxes. In *Proc. of LPAR'06*, 227–241.

Noia, T. D.; Sciascio, E. D.; and Donini, F. M. 2007. Semantic matchmaking as non-monotonic reasoning: A description logic approach. *Journal of Artificial Intelligence Research* 29:269–307.

Noia, T. D.; Sciascio, E. D.; and Donini, F. M. 2009. A tableaux-based calculus for abduction in expressive description logics: Preliminary results. In *Proc. of DL'09 Workshop*.

Peraldi, I.; Kaya, A.; Melzer, S.; Möller, R.; and Wessel, M. 2007. Towards a media interpretation framework for the semantic web. In *Proc. of WI'07*, 374–380.

Shen, Y.; Yuan, L.; You, J.; and Zhou, N. 2001. Linear tabulated resolution based on prolog control strategy. *Theory and Practice of Logic Programming* 1(1):71–103.