6-2-2008

A Forgetting-Based Approach for Reasoning with Inconsistent Distributed Ontologies

Guilin Qi

Yimin Wang

Peter Haase

Pascal Hitzler
pascal.hitzler@wright.edu

A Forgetting-based Approach for Reasoning with Inconsistent Distributed Ontologies

Guilin Qi, Yimin Wang, Peter Haase, Pascal Hitzler

Institute AIFB, Universität Karlsruhe, D-76128 Karlsruhe, Germany
{gqi,ywa,pha,phi}@aifb.uni-karlsruhe.de

Abstract. In the context of multiple distributed ontologies, we are often confronted with the problem of dealing with inconsistency. In this paper, we propose an approach for reasoning with inconsistent distributed ontologies based on *concept forgetting*. We firstly define *concept forgetting* in description logics. We then adapt the notions of *recoveries* and *preferred recoveries* in propositional logic to description logics. Two consequence relations are then defined based on the preferred recoveries.

1 Introduction

Ontologies play a central role for the success of the Semantic Web as they provide shared vocabularies for different domains, such as Medicine. Web ontologies in practical applications are usually distributed and dynamic. One of the major challenges in managing these distributed and dynamic ontologies is to handle potential inconsistencies introduced by integrating multiple distributed ontologies.

For inconsistency handling in single, centralized ontologies, several approaches are known, see e.g. [10]. There are two main ways to deal with inconsistent ontologies [10]. One way is trying to live with the inconsistency and to apply a non-standard reasoning method to obtain meaningful answers [11, 14]. For example, a general framework for reasoning with inconsistent ontologies based on *concept relevance* was proposed in [11]. The second way to deal with logical contradictions is to resolve logical modeling errors whenever a logical problem is encountered. For example, several methods have been proposed to debug erroneous terminologies and have them repaired when inconsistencies are detected [17, 16].

At the same time, formalisms for modular ontologies – such as Distributed Description Logics (DDL) [4] and $\mathcal{E}$-Connections [15] – provide an opportunity to handle multiple and distributed ontologies. Our approach is based on DDLs, where a special interpretation, called a *hole*, is proposed to interpret locally inconsistent ontologies [4, 19]. However, this approach is not fine-grained to deal with inconsistency that arises due to several connected local sources.

In [12], an approach based on *variable forgetting* is proposed to resolve inconsistency in a propositional belief base. The basic idea of this approach is to restore consistency of a belief base by *forgetting* propositional variables in some formulas of the belief base. It has been shown that this approach provides a general and flexible framework that encompasses several previous approaches for handling inconsistency as spe-

cific cases, such as coherence-based approaches for reasoning from preferred consistent subsets and approaches for merging multiple sources of information.

In this paper we define an approach for handling inconsistencies in distributed ontologies based on *concept forgetting*. We first define *concept forgetting* in description logics. We then adapt the notion of *recoveries* and *preferred recoveries* in [12] to description logics. Based on the preferred recoveries, two forgetting consequence relations are defined to draw nontrivial conclusions from inconsistent distributed ontologies. Our approach can generally be applied to any formalism for distributed ontologies. However, in this paper we present our approach in terms of DDL for two reasons. First, DDL has been proven to be a nice framework to represent distributed ontologies [2]. Second, there is some work which discusses inconsistency handling in DDL [19, 6], whilst relatively little work on dealing with inconsistency has been given for alternative approaches. By comparing our approach with existing approaches, we can show that our approach is indeed sufficiently powerful to handle inconsistency in distributed ontologies.

The rest of this paper is organized as follows. We first review DDL and variable forgetting in Section 2 and discuss related work in Section 5. We then introduce the notion of forgetting in description logic $\mathcal{ALC}$ in Section 3. We then adapt the notions of *recoveries* and *preferred recoveries* in propositional logic to DDL in Section 4. We finally conclude the paper and provide future work in Section 6.

2 Preliminaries

2.1 Distributed Description Logics

In this paper, we presume that the reader is familiar with Description Logics (DLs) and we refer to Chapter 2 of the DL handbook [1]. We consider the well-known DL $\mathcal{ALC}$ [18]. Let N_C and N_R be pairwise disjoint and countably infinite sets of *concept names* and *role names* respectively. We use the letters A and B for concept names, the letter R for role names. $\top$ and $\bot$ denote the top concept and the bottom concept respectively. The set of $\mathcal{ALC}$ concepts is the smallest set such that: (1) every concept name is a concept; (2) if C and D are concepts, R is a role name, then the following expressions are also concepts: $\neg C$ (full negation), $C \sqcap D$ (concept conjunction), $C \sqcup D$ (concept disjunction), $\forall R.C$ (value restriction on role names) and $\exists R.C$ (existential restriction on role names). A DL knowledge base (or ontology)[1] $\mathcal{KB}$ consists of a *TBox* $\mathcal{T}$ and an *ABox* $\mathcal{A}$. A TBox is a finite set of terminology axioms which are of the form $C \sqsubseteq D$, where C and D are concepts. An ABox is a finite set of assertions which is of the form $C(a)$ or $R(a, b)$, where C is a concept, R is a role, and a, b are individuals.

Distributed Description Logics (DDL) [4] adopt a linking mechanism. It is a well-developed logical language for representing both ontologies and context [5]. In DDL, the *distributed knowledge base (DKB)*, $\mathfrak{D} = \langle \{\mathcal{KB}_i\}_{i \in I}, \mathfrak{B} \rangle$ consists a set of local knowledge bases $\{\mathcal{KB}_i\}_{i \in j}$ in $\mathcal{ALC}$ and bridge rules $\mathfrak{B} = \{\mathfrak{B}_{ij}\}_{i \neq j \in I}$ that represent the correspondences between their terminologies. The semantic mappings between modules $\mathcal{KB}_i$ and $\mathcal{KB}_j$ are represented in terms of axioms of the following form:

[1] In this paper, we consider a DL knowledge base as an ontology and use them interchangeably.

– INTO: $i : C \xrightarrow{\sqsubseteq} j : D$
– ONTO: $i : C \xrightarrow{\sqsupseteq} j : D$

The semantics of DKB is given by the interpretation $\mathfrak{J} = (\{\mathcal{I}\}_{i\in I}, \{r_{ij}\}_{i\neq j\in I})$ that consists of standard DL interpretations $\mathcal{I}_i$ of $\mathcal{KB}_i$, and binary relations $r_{ij} \subseteq \Delta^{\mathcal{I}_i} \times \Delta^{\mathcal{I}_j}$. We use $r_{ij}(X)$ to denote $\cup_{x\in X}\{y \in \Delta^{\mathcal{I}_j} : (x, y) \in r_{ij}\}$ for any $X \subseteq \Delta^{\mathcal{I}_i}$. A distributed interpretation $\mathfrak{J}$ d-satisfies (denoted as $\mathfrak{J} \models_d$) the element of the DKB $\mathfrak{D} = \langle\{\mathcal{KB}_i\}_{i\in I}, \mathfrak{B}\rangle$ if

– $\mathcal{I}_i \models_d A\sqsubseteq B$ for all $A \sqsubseteq B$ in $\mathcal{KB}_i$
– $\mathfrak{J} \models_d i : C \xrightarrow{\sqsubseteq} j : D$, if $r_{ij}(C^{\mathcal{I}_i}) \subseteq D^{\mathcal{I}_j}$ (INTO)
– $\mathfrak{J} \models_d i : C \xrightarrow{\sqsupseteq} j : D$, if $r_{ij}(C^{\mathcal{I}_i}) \supseteq D^{\mathcal{I}_j}$ (ONTO)
– $\mathfrak{J} \models_d \mathfrak{D}$, if for every $i, j\in I$, $\mathfrak{J} \models_d KB_i$ and $\mathfrak{J} \models_d \mathfrak{B}_{ij}$,

where $\mathcal{I}_i$ and $\mathcal{I}_j$ are local interpretations of $\mathcal{KB}_i$ and $\mathcal{KB}_j$, respectively, C, D are concepts, r_{ij} (called *domain relation*) is a relation that represents an interpretation of B_{ij}. Finally, $\mathfrak{D} \models_d i : C \sqsubseteq D$ if for every $\mathfrak{J}$, $\mathfrak{J} \models_d \mathfrak{D}$ implies $\mathfrak{J} \models_d i : C \sqsubseteq D$. More details of the semantics of DDL can be found in [4].

A DKB is inconsistent if there is no interpretation $\mathfrak{J}$ which satisfies all $\mathcal{KB}_i$, where $i \in I$. Given a DKB $\mathfrak{D} = \langle\{\mathcal{KB}_i\}_{i\in I}, \mathfrak{B}\rangle$, a DKB $\mathfrak{D}' = \langle\{\mathcal{KB}'_i\}_{i\in I}, \mathfrak{B}'\rangle$ is a sub-base of $\mathfrak{D}$, denoted $\mathfrak{D}'\sqsubseteq\mathfrak{D}$, iff, $\mathcal{KB}'_i\subseteq\mathcal{KB}_i$ for all $i\in I$ and $\mathfrak{B}'_i\subseteq\mathfrak{B}_i$.

2.2 Variable forgetting

Let L be a propositional language constructed from a finite alphabet P of propositional symbols, the Boolean constants $\top$ (true) and $\bot$ (false), using the usual operators $\neg$ (not), $\vee$ (or) and $\wedge$ (and). The classical consequence relation is denoted by $\vdash$. Given a propositional formula ϕ, $Var(\phi)$ denotes the set of propositional variables occuring in ϕ. $\phi_{x\leftarrow 0}$ (resp. $\phi_{x\leftarrow 1}$) denotes the formula obtained by replacing in ϕ every occurence of variable x by $\bot$ (resp. $\top$).

Variable forgetting, which is also known as projection, is originally defined in [13] and is applied to resolve inconsistency in [12].

Definition 1. *Let ϕ be a formula over P and $V\subseteq P$ be a set of propositional symbols. The forgetting of V in ϕ, denoted as* $\mathsf{Forget}(V, \phi)$*, is a formula over P, defined inductively as follows:*

– $\mathsf{Forget}(\varnothing, \phi)\equiv\phi$;
– $\mathsf{Forget}(\{x\}, \phi)\equiv\phi_{x\leftarrow 0} \vee \phi_{x\leftarrow 1}$;
– $\mathsf{Forget}(\{x\}\cup V, \phi)\equiv\mathsf{Forget}(V, \mathsf{Forget}(\{x\}, \phi))$.

$\mathsf{Forget}(V, \phi)$ is the logically strongest consequence of ϕ which is independent of V. By forgetting a set of variables in a formula ϕ we get a formula which is inferentially weaker than ϕ, i.e. $\phi \vdash \mathsf{Forget}(V, \phi)$. For example, $\mathsf{Forget}(\{a\}, \neg a\wedge b)\equiv b$ and $\mathsf{Forget}(\{a\}, a\vee b)\equiv\top$.

3 Forgetting in Description Logics

In this section, we define forgetting in a TBox of a DL knowledge base. The notion of forgetting can be applied to any description logic. Here, a concept name plays the same role as a variable in propositional logic. In particular, we define the forgetting of a set of concept names in a concept. We use $\mathsf{Con}(C)$ to denote the concept names in a concept C. $C_{A\leftarrow\top}$ (resp. $C_{A\leftarrow\bot}$) denotes the concept obtained by replacing in C every occurrence of concept name A by $\top$ (resp. $\bot$).

A straightforward way to forget a concept name A in a concept C can be given as follows: $\mathsf{Forget}(\{A\}, C)$=$C_{A\leftarrow\top}\sqcup C_{A\leftarrow\bot}$. However, this definition suffers from a problem. That is, by forgetting a concept name A in a concept C in a TBox $\mathcal{T}$, we do not necessarily obtain a concept which is logically weaker than C, i.e. $\mathcal{T} \models C \sqsubseteq \mathsf{Forget}(\{A\}, C)$, a very important property of forgetting. The main reason for this problem is that the DL concept constructors include value restrictions on role names and existential restriction on role names. Let us look at an example. Suppose $C = \forall R.(A\sqcap\forall R'.\neg A)$ and we want to forget A. Since $C_{A\leftarrow\top} = \forall R.(\top\sqcap\forall R'.\bot) = \forall R.(\forall R'.\bot)$ and $C_{A\leftarrow\bot} = \forall R.(\bot\sqcap\forall R'.\top) = \forall R.\bot$, we have $\mathsf{Forget}(\{A\}, C) = (\forall R.(\forall R'.\bot)) \sqcup \forall R.\bot$. In this case, we do not necessarily have $\mathcal{T} \models C \sqsubseteq \mathsf{Forget}(\{A\}, C)$.

We consider the following definition of forgetting.

Definition 2. *Let C be a (complex) concept and A a concept name. Suppose C is in negation normal form, i.e. negation can only appear in the front of a concept name. The forgetting of A in C, denoted* $\mathsf{Forget}(\{A\}, C)$*, is a concept obtained by replacing every occurrence of A and $\neg A$ by $\top$.*

In Definition 2, it is very important that the concept C is transformed into its negation normal form before we apply the forgetting operator to it. Otherwise, we cannot ensure that $C\sqsubseteq\mathsf{Forget}(\{A\}, C)$. For example, suppose $C = \neg(A_1\sqcap B_1)$ and we want to forget A_1. If we replace A_1 by $\top$ without transforming C into its negation normal form, then $C' = \neg B_1$, which is subsumed by C.

We give a running example to illustrate the notion of forgetting.

Example 1. Let us consider a TBox
$\mathcal{T} = \{\mathsf{Professor}\sqcup\mathsf{PostDoctor}\sqcup\mathsf{ResearchAssistant}\sqsubseteq\mathsf{Employee}, \mathsf{PhDStudent}\sqsubseteq \mathsf{ResearchAssistant}, \mathsf{PhDSupervisor} \equiv \mathsf{Professor}\sqcap\exists\mathsf{isSupervisorof}.\mathsf{PhDStudent}\}$. Then,

$\mathsf{Forget}(\{\mathsf{Professor}\}, \mathsf{Professor}\sqcup\mathsf{PostDoctor}\sqcup\mathsf{ResearchAssistant}) = \top$ and

$\mathsf{Forget}(\{\mathsf{Professor}\}, \mathsf{Professor}\sqcap\exists\mathsf{isSupervisorof}.\mathsf{PhDStudent}) = \exists\mathsf{isSupervisorof}.\mathsf{PhDStudent}$.

The following proposition tells us that the forgetting of A in C indeed results in a concept which is a super concept of C.

Proposition 1. *Let C be a concept in a TBox $\mathcal{T}$ and A be a concept name. Then $\mathcal{T} \models C\sqsubseteq\mathsf{Forget}(\{A\}, C)$.*

We now define the forgetting of a set of concept names in a concept.

Definition 3. *Let C be a concept, and let S_N be a finite set of concept names. The forgetting of S_N in C, denoted* $\mathsf{Forget}(S_N, C)$*, is defined inductively as follows:*

– $\mathsf{Forget}(\varnothing, C)$=$C$;
– $\mathsf{Forget}(\{A\} \cup S_N, C) = \mathsf{Forget}(S_N, \mathsf{Forget}(\{A\}, C))$.

Example 2. (Example 1 Continued) Let $C = \mathsf{Professor} \sqcap \exists \mathsf{isSupervisorof}.\mathsf{PhDStudent}$. Suppose we want to forget Professor and PhDStudent in C, i.e., $S_N = \{\mathsf{Professor}, \mathsf{PhDStudent}\}$, then we have $\mathsf{Forget}(S_N, C) = \exists \mathsf{isSupervisorof}.\top$.

Unlike the forgetting notion in propositional theories, we do not have a corresponding model theoretic semantics for concept forgetting. However, we have the following properties which precisely characterize our notion of concept forgetting.

Proposition 2. *Let C, D and E be concepts, and let A and A' be two concept names. We then have*

1. *If $C = A$ or $\neg A$, then $\mathsf{Forget}(\{A\}, C) = \top$;*
2. *If $C = D \sqcap E$ (or $D \sqcup E$), where $A \in \mathsf{Con}(D)$ and $A \notin \mathsf{Con}(E)$, then $\mathsf{Forget}(\{A\}, C) = \mathsf{Forget}(\{A\}, D) \sqcap E$ (or $\mathsf{Forget}(\{A\}, C) = \mathsf{Forget}(\{A\}, D) \sqcup E$).*
3. *If $C = A \sqcup E$, then $\mathsf{Forget}(\{A\}, C) = \top$.*

The proof of Proposition 2 is clear by the definition of forgetting. Item 1 says that if C is a concept name or full negation of a concept name, then by forgetting it we get a top concept. Item 2 shows that forgetting a concept name A is a concept which is a conjunction (or disjunction) will not influence the conjunct (or disjunct) which is irrelevant to the concept name. Item 3 is a disjunction of the form $A \sqcup E$, then by forgetting A we get a top concept.

We now consider the weakening of a TBox $\mathcal{T}$ by forgetting a concept name A. A first thought would be that we simply forget A in every concept appearing in $\mathcal{T}$. However, this approach may cause some problems. Suppose a terminology axiom $C \sqsubseteq D$ is in $\mathcal{T}$ such that $C = A \sqcap B$, and $A \notin \mathsf{Con}(D)$ and $A \notin \mathsf{Con}(B)$. By forgetting A in C we get a new axiom $B \sqsubseteq D$. It is easy to check that $B \sqsubseteq D$ infers $C \sqsubseteq D$ whilst the converse does not always hold. Therefore, $C \sqsubseteq D$ is not weakened. On the contrary, it is reinforced. To solve this problem, we can equivalently transform every axiom of the form $C \sqsubseteq D$ into $\top \sqsubseteq \neg C \sqcup D$. According to Proposition 1, we have the following corollary.

Corollary 1. *Let ϕ be a terminology axiom of the form $\top \sqsubseteq C$ and A a concept name. Suppose $A \in \mathsf{Con}(C)$, then $\top \sqsubseteq C \models \top \sqsubseteq \mathsf{Forget}(\{A\}, C)$, but not vice versa.*

Corollary 1 tells us that an axiom is weakened by forgetting a concept name in the right-side concept.

We give the notion of forgetting in a TBox.

Definition 4. *Let $\mathcal{T}$ be a TBox. Let ϕ be a terminology axiom in $\mathcal{T}$ which has been transformed into the form $\top \sqsubseteq C$ and S_N a finite set of concept names. The forgetting of S_N in ϕ is defined as $\mathsf{Forget}(S_N, \phi) = \top \sqsubseteq \mathsf{Forget}(S_N, C)$ and the forgetting of S_N in $\mathcal{T}$ is defined as $\mathsf{Forget}(S_N, \mathcal{T}) = \{\mathsf{Forget}(S_N, \phi) : \phi \in \mathcal{T}, Sig(\phi) \cap S_N \neq \varnothing\}$, where $Sig(\phi)$ denotes the signature of ϕ.*

By Corollary 1, it is clear that we have that every axiom in $\mathsf{Forget}(S_N, \mathcal{T})$ can be inferred from $\mathcal{T}$, for any S_N.

Example 3. (Example 1 Continued) Suppose we want to forget Professor in $\mathcal{T}$. We need some pre-processing. First, Professor⊔PostDoctor⊔ResearchAssistant⊑ Employee is transformed into $\top \sqsubseteq (\neg\mathsf{Professor} \sqcap \neg\mathsf{PostDoctor} \sqcap \neg\mathsf{ResearchAssistant}) \sqcup \mathsf{Employee}$. Second, we split PhDSupervisor ≡ Professor⊓∃isSupervisorof.PhDStudent into PhDSupervisor ⊑ Professor⊓∃isSupervisorof.PhDStudent and

Professor⊓∃isSupervisorof.PhDStudent ⊑ PhDSupervisor, then transform them into $\top \sqsubseteq \neg\mathsf{PhDSupervisor} \sqcup (\mathsf{Professor} \sqcap \exists\mathsf{isSupervisorof.PhDStudent})$ and $\top \sqsubseteq \neg\mathsf{Professor} \sqcup \neg\exists\mathsf{isSupervisorof.PhDStudent} \sqcup \mathsf{PhDSupervisor}$. Therefore, we have

$$\begin{aligned}\mathsf{Forget}(\{\mathsf{Professor}\}, \mathcal{T}) = \{&\top \sqsubseteq (\neg\mathsf{PostDoctor} \sqcap \neg\mathsf{ResearchAssistant}) \sqcup \mathsf{Employee},\\ &\top \sqsubseteq \neg\mathsf{PhDSupervisor} \sqcup \exists\mathsf{isSupervisorof.PhDStudent}\\ &\mathsf{PhDStudent} \sqsubseteq \mathsf{ResearchAssistant}\}.\end{aligned}$$

4 Recoveries for Distributed DL Knowledge Bases

In this section, we apply the forgetting-based approach for resolving inconsistency in a propositional knowledge base [12] to deal with inconsistency in a DKB by forgetting concepts in TBoxes of local ontologies. We do not consider inconsistency which is caused only by inter-ABoxes as this kind of inconsistency may be better addressed by performing some kind of weakening on the ABoxes of local ontologies. Dealing with this problem is also important, but it is out of scope for this paper.

4.1 Recoveries

In [12], two key notions of the approach are forgetting context and recovery vector. The former one is a collection of sets of variables to be forgotten in each formula from the knowledge base, and the latter one is a vector of subsets of the set of all the variables appearing in the knowledge base, which satisfies the constraints in the forgetting context. Forgetting of the variables in the recovery vector will result in a consistent knowledge base. We adapt the definitions of forgetting context and recovery vector to DDL.

In the following, we assume that $I = [n]$ where $n \in \mathbb{N}$. The forgetting context in DDL is defined as follows.

Definition 5. *Given a DKB $\mathfrak{D} = \langle \{\mathcal{KB}_i\}_{i\in I}, \mathfrak{B}\rangle$, where $\mathcal{KB}_i = \langle \mathcal{R}_i, \mathcal{T}_i, \mathcal{A}_i\rangle$, a forgetting context $\mathcal{C}_\mathcal{D}$ for it is a triple $\langle F, G, H\rangle$ where:*

- *$F = \langle F_i\rangle_{i\in I}$ such that $F_k \subseteq N_C$ with $k \in I$, and for each $i \in I$, F_i is the set of concept names that can be forgotten in $\mathcal{T}_i$.*
- *$G = \langle G_i\rangle_{i\in I}$ such that for each i, $G_i = \{(A_i, A_i^{'}) : A_i, A_i' \in CN(\mathcal{T}_i)\}$, where $CN(\mathcal{T}_i)$ is the set of all concept names in $\mathcal{T}_i$ and a pair (A_i, A_i') in G_i means that A_i' is meaningful only if A_i exists;*

– $H \subseteq N_C \times \{1, ..., n\} \times N_C \times \{1, ..., n\}$ *is a set of quadruples* (A, i, B, j) *such that* $A \in F_i$, $B \in F_j$ *and* $i \neq j$, *which means that if* A *is forgotten in* F_i, *then* B *must be forgotten as well in* F_j.

F_i imposes the constraints over the concept names which can be forgotten in $\mathcal{T}_i$. That is, if a concept name $A \notin F_i$, then forgetting A in $\mathcal{T}_i$ is forbidden. G_i contains the pairs of concept names in local TBox $\mathcal{T}_i$ such that the existence of the former is meaningful or significant only in presence of the latter. In other words, forgetting the latter imposes to forget the former. H imposes the constraints over inter-TBoxes. It forces that if A is forgotten in $\mathcal{T}_i$ then B should also be forgotten in $\mathcal{T}_j$. The forgetting context is dependent on the application at hand. A simple example for forgetting context is a *standard* forgetting context which is given by $F_i = N_C$ for every $i = 1, ..., n$, $G_i = \varnothing$ for every $i = 1, ..., n$, and $H = \varnothing$. Note that our approach does currently not include weakening of bridge rules. Such a weakening may be preferable in some situations (for example, to improve mappings created by different matching systems [6]), but its treatment is out of scope for this paper.

We next define a recovery vector in DDL. It is a vector of subsets of concept names of N_C, which satisfies the constraints in the forgetting context.

Definition 6. *Given a DKB* $\mathfrak{D} = \langle \{\mathcal{KB}_i\}_{i \in I}, \mathfrak{B} \rangle$, *where* $\mathcal{KB}_i = \langle \mathcal{R}_i, \mathcal{T}_i, \mathcal{A}_i \rangle$, *and a forgetting context* $\mathcal{C}_\mathfrak{D}$, *a recovery vector (or recovery for short)* $\overrightarrow{V}$ *for* $\mathfrak{D}$ *given* $\mathcal{C}_\mathfrak{D}$ *is a vector* $\overrightarrow{V} = \langle V_1, ..., V_n \rangle$ *of subsets of* N_C *s.t.:*

– *for every* $i \in I$, $V_i \subseteq \mathsf{Con}(\mathcal{T}_i)$,
– *for every* $i \in I$, *for any* $(A, A') \in G_i$, *if* $A \in V_i$ *then* $A' \in V_i$,
– *for every* $(A, i, B, j) \in H$, $A \in V_i$ *implies* $B \in V_j$,
– $\mathfrak{D}|\overrightarrow{V} = \langle \{\langle \mathsf{Forget}(V_i, \mathcal{T}_i), \mathcal{A}_i \rangle\}_{i \in I}, \mathfrak{B} \rangle$ *is consistent.*

$\mathfrak{D}|\overrightarrow{V}$ *is called the cure of* $\overrightarrow{V}$ *for* $\mathfrak{D}$ *given* $\mathcal{C}_\mathcal{D}$. *We denote the set of all recoveries for* $\mathfrak{D}$ *given* $\mathcal{C}_\mathfrak{D}$ *as* $\mathcal{R}_{\mathcal{C}_\mathfrak{D}}(\mathfrak{D})$.

According to the definition of recovery, the cure of $\overrightarrow{V}$ for $\mathfrak{D}$ is always consistent if the recovery $\overrightarrow{V}$ exists. When $\mathcal{R}_{\mathcal{C}_\mathfrak{D}}(\mathfrak{D}) = \varnothing$, we say that $\mathfrak{D}$ is not recoverable $w.r.t.$ $\mathcal{C}_\mathfrak{D}$. This may happen when some concepts, which cannot be forgotten according to the forgetting context, must be forgotten to restore consistency.

We give an example to illustrate the definitions in this section.

Example 4. Suppose we have a
DKB $\mathfrak{D} = \langle \{\mathcal{KB}_1, \mathcal{KB}_2\}, \{\mathfrak{B}_{12}, \mathfrak{B}_{21}\} \rangle$, where
(1) $\mathcal{KB}_1 = \langle \mathcal{R}_1, \mathcal{T}_1, \mathcal{A}_1 \rangle$:

– $\mathcal{R}_1 = \varnothing$;
– $\mathcal{T}_1 = \{\mathsf{Professor} \sqcup \mathsf{PostDoctor} \sqcup \mathsf{ResearchAssistant} \sqsubseteq \mathsf{Employee}$,
  $\mathsf{PhDStudent} \sqsubseteq \mathsf{ResearchAssistant}$,
  $\mathsf{PhDSupervisor} \equiv \mathsf{Professor} \sqcap \exists \mathsf{isSupervisorof}.\mathsf{PhDStudent}\}$;
– $\mathcal{A}_1 = \{\mathsf{PhDStudent}(Mary), \mathsf{PhDSupervisor}(Tom)\}$;

(2) $\mathcal{KB}_2 = \langle \mathcal{R}_2, \mathcal{T}_2, \mathcal{A}_2 \rangle$:

- $\mathcal{R}_2 = \varnothing$;
- $\mathcal{T}_2$=$\{\mathsf{Professor} \sqcup \mathsf{Lecturer} \sqsubseteq \mathsf{AcademicStaff}, \mathsf{Student} \sqcap \mathsf{AcademicStaff} \sqsubseteq \bot$, $\mathsf{PhDSupervisor} \equiv \mathsf{AcademicStaff} \sqcap \exists \mathsf{isSupervisorof}.\mathsf{PhDStudent}$, $\mathsf{PhDStudent} \sqsubseteq \mathsf{Student}\}$;
- $\mathcal{A}_2 = \{\mathsf{PhDStudent}(John)\}$;

(3) $\mathfrak{B}_{21} = \{2 : \mathsf{AcademicStaff} \xrightarrow{\sqsupseteq} 1 : \mathsf{Employee}, 2 : \mathsf{PhDStudent} \xrightarrow{\sqsupseteq} 1 : \mathsf{PhDStudent}$, $2 : \neg\mathsf{AcademicStaff} \xrightarrow{\sqsubseteq} 1 : \neg\mathsf{Employee}, 2 : \neg\mathsf{PhDStudent} \xrightarrow{\sqsubseteq} 1 : \neg\mathsf{PhDStudent}\}$ and $\mathfrak{B}_{12} = \varnothing$. $\mathcal{KB}_1$ and $\mathcal{KB}_2$ are two University ontologies which are connected by bridge rules.

Let $\mathfrak{D}' = \langle\{\mathcal{T}_1, \mathcal{T}_2\}, \{\mathfrak{B}_{12}, \mathfrak{B}_{21}\}\rangle$. Since $\mathfrak{D}' \models_d \mathsf{PhDStudent} \sqsubseteq \bot$ and $\mathsf{PhDStudent}(Mary) \in \mathcal{A}_1$, $\mathfrak{D}$ is inconsistent. We have the following forgetting context $\mathcal{C}_{\mathfrak{D}}$ for $\mathfrak{D}$:

- F=$\langle F_i\rangle_{i=1,2}$, where F_i= $\{\mathsf{AcademicStaff}, \mathsf{Employee}, \mathsf{Student}, \mathsf{Professor}, \mathsf{Lecturer}$, $\mathsf{ResearchAssistant}, \mathsf{PhDStudent}, \mathsf{PhDSupervisor}\}_{i=1,2}$.
- G=$\langle G_i\rangle_{i=1,2}$, where G_1=$\{(\mathsf{PhDStudent}, \mathsf{PhDSupervisor})\}$; G_2=$\{(\mathsf{PhDStudent}, \mathsf{PhDSupervisor})\}$.
- $H = \{(\mathsf{PhDStudent}, 1, \mathsf{PhDStudent}, 2)\}$.

For each local ontology, all the concept names in it can be forgotten. G_1 is used to show that the concept *PhD supervisor* is not meaningful if the concept *PhD student* or *professor* is forgotten. G_2 can be similarly explained. H ensures that the concepts *PhD student*, *professor*, *student* in $\mathcal{KB}_1$ are respectively equal to *PhD student*, *professor*, *student* in $\mathcal{KB}_2$ $w.r.t.$ weakening from the point of view of $\mathcal{KB}_2$.

Given the forgetting context $\mathcal{C}_{\mathfrak{D}}$, the following are two recoveries for $\mathfrak{D}$:

- $\overrightarrow{V}_1 = \langle V_1, V_2\rangle$, where $V_1 = \{\mathsf{ResearchAssistant}\}$ and $V_2 = \varnothing$: $\mathsf{Forget}(V_1, \mathcal{T}_1) = \{\mathsf{Professor} \sqcup \mathsf{PostDoctor} \sqsubseteq \mathsf{Employee}$, $\mathsf{PhDSupervisor} \equiv$ $\mathsf{Professor} \sqcap \exists \mathsf{isSupervisorof}.\mathsf{PhDStudent}\}$, $\mathsf{Forget}(V_2, \mathcal{T}_2) = \mathcal{T}_2$ It is easy to check that the DKB $\mathfrak{D}_1 = \langle\{\langle\mathsf{Forget}(V_i, \mathcal{T}_i), \mathcal{A}_i\rangle\}_{i=1,2}, \mathfrak{B}\rangle$ is consistent.
- $\overrightarrow{V}_2 = \langle V_1, V_2\rangle$, where $V_1 = \varnothing$ and $V_2 = \{\mathsf{PhDStudent}, \mathsf{PhDSupervisor}\}$: $\mathsf{Forget}(V_1, \mathcal{T}_1) = \mathcal{T}_1$, $\mathsf{Forget}(V_2, \mathcal{T}_2) = \{\mathsf{Professor} \sqcup \mathsf{Lecturer} \sqsubseteq \mathsf{AcademicStaff}, \mathsf{Student} \sqcap \mathsf{AcademicStaff} \sqsubseteq \bot\}$ The DKB $\mathfrak{D}_2 = \langle\{\langle\mathsf{Forget}(V_i, \mathcal{T}_i), \mathcal{A}_i\rangle\}_{i=1,2}, \mathfrak{B}\rangle$ is consistent.

However, the following vector is not a recovery for $\mathfrak{D}$ given $\mathcal{C}_{\mathfrak{D}}$: $\overrightarrow{V} = \langle V_1, V_2\rangle$, where $V_1 = \{\mathsf{PhDStudent}\}$ and $V_2 = \varnothing$, because it does not satisfy the constraints of H in $\mathcal{C}_{\mathfrak{D}}$.

4.2 Preferred Recoveries

Given a forgetting context, there may exist several different recoveries. In many cases, we are only interested in some of these recoveries, called *preferred recoveries*, which is defined by some preference relation on recoveries. For example, among all the recoveries, we may be only interested in those contain minimal number of concepts to forgotten. The notion of preferred recovery is originally defined in [12] in propositional logic. We adapt it to DDLs here. We need to define a preference relation on the set of all recoveries for a DKB $\mathfrak{D}$ given a forgetting context $\mathcal{C}_{\mathfrak{D}}$ for it. For any two recoveries $\overrightarrow{V}$ and $\overrightarrow{V}'$ of length n, $\overrightarrow{V} \subseteq_c \overrightarrow{V}'$ means that $V_i \subseteq V_i'$ for each $i \in 1, ..., n$.

Definition 7. *Given a DKB $\mathfrak{D}$ and a forgetting context $\mathcal{C}_{\mathfrak{D}}$ for it, a preference relation $\preceq$ on $\mathcal{R}_{\mathcal{C}_{\mathfrak{D}}}(\mathfrak{D})$ is a reflexive and transitive relation which satisfies the following property:*

$$\forall \overrightarrow{V}, \overrightarrow{V}' \in \mathcal{R}_{\mathcal{C}_{\mathfrak{D}}}(\mathfrak{D}),\ if\ \overrightarrow{V} \subseteq_c \overrightarrow{V}',\ then\ \overrightarrow{V} \preceq \overrightarrow{V}'.$$

As usual, we denote $\overrightarrow{V} \prec \overrightarrow{V}'$ for $\overrightarrow{V} \preceq \overrightarrow{V}'$ and $\overrightarrow{V}' \not\preceq \overrightarrow{V}$, and $\overrightarrow{V} \sim \overrightarrow{V}'$ for $\overrightarrow{V} \preceq \overrightarrow{V}'$ and $\overrightarrow{V}' \preceq \overrightarrow{V}$.

In Definition 7, a recovery $\overrightarrow{V}$ is at least preferred to another one $\overrightarrow{V}'$ ($\overrightarrow{V} \preceq \overrightarrow{V}'$) if each V_i in $\overrightarrow{V}$ is a subset of V_i' in $\overrightarrow{V}'$.

There are many ways to define a preference relation. For instance, we can prefer recoveries that lead to forget minimal sets of concepts $w.r.t.$ the set containment. We can also ask the experts or users for such a preference relation. We adapt two specific preference relations in [12] as follows.

- *binaricity* preference relation $\preceq_{bin}$: $\forall \overrightarrow{V}, \overrightarrow{V}' \in \mathcal{R}_{\mathcal{C}_{\mathfrak{D}}}(\mathfrak{D})$, if for every $i = 1, ..., n$, $(V_i \neq \varnothing \Leftrightarrow V_i' \neq \varnothing)$, then $\overrightarrow{V} \sim_{bin} \overrightarrow{V}'$.
- *ranking function* based preference relation $\preceq_{\mu}$: suppose μ is a *ranking function* from $\mathcal{R}_{\mathcal{C}_{\mathfrak{D}}}(\mathfrak{D})$ to $\mathbb{N}$ such that $\mu(\langle \varnothing, ..., \varnothing \rangle) = 0$, and $\forall \overrightarrow{V}, \overrightarrow{V}' \in \mathcal{R}_{\mathcal{C}_{\mathfrak{D}}}(\mathfrak{D})$, if $\overrightarrow{V} \subseteq_c \overrightarrow{V}'$, then $\mu(\overrightarrow{V}) \leq \mu(\overrightarrow{V}')$. The preference relation $\preceq_{\mu}$ induced by μ is the total preordering defined as follows. For all $\overrightarrow{V}, \overrightarrow{V}' \in \mathcal{R}_{\mathcal{C}_{\mathfrak{D}}}(\mathfrak{D})$, $\overrightarrow{V} \preceq_{\mu} \overrightarrow{V}'$ if and only if $\mu(\overrightarrow{V}) \leq \mu(\overrightarrow{V}')$.

The ranking function based preference relation is defined by a ranking function from $\mathcal{R}_{\mathcal{C}_{\mathfrak{D}}}(\mathfrak{D})$ to $\mathbb{N}$. A particular ranking function can be defined by $\mu_{card}(\overrightarrow{V}) = |\bigcup \overrightarrow{V}|$, where $\bigcup \overrightarrow{V} = V_1 \cup ... \cup V_n$. We denote the preference relation based on μ_{card} as $\preceq_{card}$. A recovery $\overrightarrow{V}$ is preferred to another one $\overrightarrow{V}'$ $w.r.t.$ the preference relation $\preceq_{card}$ if and only if the cardinality of the union of the sets V_i $(i = 1, ..., n)$ is smaller than that of the union of the sets V_i' $(i = 1, ..., n)$.

By Corollary 1, when a concept name is forgotten in a DL knowledge base, the resulting DL knowledge base is weaker than the original one $w.r.t.$ the inference power. Given $\overrightarrow{V}, \overrightarrow{V}' \in \mathcal{R}_{\mathcal{C}_{\mathfrak{D}}}(\mathfrak{D})$, if $\overrightarrow{V} \subseteq_c \overrightarrow{V}'$ then $\mathfrak{D}|\overrightarrow{V} \models \mathfrak{D}|\overrightarrow{V}'$. Therefore, among all the recoveries from $\mathcal{R}_{\mathcal{C}_{\mathfrak{D}}}(\mathfrak{D})$, those which are minimal $w.r.t.$ $\subseteq_c$ lead to cures preserving as much information as possible given the forgetting context $\mathcal{C}_{\mathfrak{D}}$.

Next, we define two consequence relations. We denote by $\overrightarrow{V}^{\preceq}_{\mathfrak{D},\mathcal{C}_{\mathfrak{D}}}$ the set of all minimal recoveries from $\mathcal{R}_{\mathcal{C}_{\mathfrak{D}}}(\mathfrak{D})$ $w.r.t.$ $\preceq$.

Definition 8. *Given a DKB $\mathfrak{D}$ and a forgetting context $\mathcal{C}_{\mathfrak{D}}$ for it, let $\preceq$ be a preference relation on $\mathcal{R}_{\mathcal{C}_{\mathfrak{D}}}(\mathfrak{D})$. Let ϕ be a DL axiom. Then ϕ is said to be skeptically inferred from $\mathfrak{D}$ $w.r.t.$ $\preceq$, denoted by $\mathfrak{D} \models^{\mathcal{C}_{\mathfrak{D}}}_{\preceq,Ske} \phi$, if and only if $\mathfrak{D}|\overrightarrow{V} \models \phi$ for all $\overrightarrow{V} \in \overrightarrow{V}^{\preceq}_{\mathfrak{D},\mathcal{C}_{\mathfrak{D}}}$. ϕ is called a skeptical consequence of $\mathfrak{D}$.*

A DL axiom is a skeptical consequence of a DKB $\mathfrak{D}$ if and only if it can be inferred from all the most preferred recoveries.

Example 5. (Example 4 Continued) Suppose $\preceq = \preceq_{card}$. Then $\overrightarrow{V}_1$ is a preferred recovery because $|\bigcup \overrightarrow{V}_1| = 1$ and no other recovery can have cardinality smaller than it. Other preferred recoveries are given as follows:

- $\overrightarrow{V}_3 = \langle V_1, V_2 \rangle$, where $V_1 = \{\mathsf{Employee}\}$ and $V_2 = \varnothing$. We have $\mathsf{Forget}(V_3, \mathcal{T}_1) =$ $\{\mathsf{PhDStudent} \sqsubseteq \mathsf{ResearchAssistant}, \mathsf{PhDSupervisor} \equiv \mathsf{Professor} \sqcap \exists \mathsf{isSupervisorof}.\mathsf{PhDStudent}\}$ and $\mathsf{Forget}(V_3, \mathcal{T}_2) = \mathcal{T}_2$.
- $\overrightarrow{V}_4 = \langle V_1, V_2 \rangle$, where $V_1 = \varnothing$ and $V_2 = \{\mathsf{AcademicStaff}\}$. We have $\mathsf{Forget}(V_4, \mathcal{T}_1) = \mathcal{T}_1$ and $\mathsf{Forget}(V_4, \mathcal{T}_2) = \{\mathsf{PhDStudent} \sqsubseteq \mathsf{Student}, \mathsf{PhDSupervisor} \sqsubseteq \mathsf{AcademicStaff} \sqcap \exists \mathsf{isSupervisorof}.\mathsf{PhDStudent}\}$.
- $\overrightarrow{V}_5 = \langle V_1, V_2 \rangle$, where $V_1 = \varnothing$ and $V_2 = \{\mathsf{Student}\}$. We have $\mathsf{Forget}(V_5, \mathcal{T}_1) = \mathcal{T}_1$ and $\mathsf{Forget}(V_5, \mathcal{T}_2) = \{\mathsf{Professor} \sqcup \mathsf{SeniorLecturer} \sqcup \mathsf{Lecturer} \sqcup \mathsf{ResearchAssistant} \sqsubseteq \mathsf{AcademicStaff}\}$.

Since $\mathsf{PhDSupervisor} \equiv \mathsf{Professor} \sqcap \exists \mathsf{isSupervisorof}.\mathsf{PhDStudent}$ is in $\mathsf{Forget}(V_i, \mathcal{T}_i)$ for all $i = 1, 3, 4, 5$, we have that $\mathfrak{D}|\overrightarrow{V} \models \mathsf{Professor}(Tom)$, for all $\overrightarrow{V} \in \overrightarrow{V}^{\preceq}_{\mathfrak{D},\mathcal{C}_{\mathfrak{D}}}$. That is, we can conclude that Tom is a professor using the skeptical inference.

Suppose $\mathfrak{D} = \langle \{\mathcal{KB}_i\}_{i \in I}, \mathfrak{B} \rangle$ is a DKB. A maximal consistent sub-base of $\mathfrak{D}$ $w.r.t$ terminology is a sub-base $\mathfrak{D}' = \langle \{\mathcal{KB}'_i\}_{i \in I}, \mathfrak{B} \rangle$ (with $\mathcal{KB}' = \langle \mathcal{R}', \mathcal{T}', \mathcal{A}' \rangle$) of $\mathfrak{D}$ which satisfies the following conditions: (1) $\mathfrak{D}'$ is consistent; (2) $\{\mathcal{T}'_i : \mathcal{T}'_i \neq \varnothing\} \subseteq \{\mathcal{T}_i : i = 1, ..., n\}$, $\mathcal{R}' = \mathcal{R}$, and $\mathcal{A}'_i = \mathcal{A}_i$; (3) there does not exist a sub-base $\mathfrak{D}''$ of $\mathfrak{D}$ which satisfies (1) and (2), and $\{\mathcal{T}'_i : \mathcal{T}'_i \neq \varnothing\} \subset \{\mathcal{T}''_i : \mathcal{T}''_i \neq \varnothing\}$.

Proposition 3. *Let $\mathfrak{D}$ be a DKB and $\mathsf{MaxCons}(\mathfrak{D})$ the set of all maximal consistent sub-bases of $\mathfrak{D}$. Suppose $\mathcal{C}_{\mathfrak{D}}$ is the standard forgetting context and the preference relation $\preceq$ is a binaricity preference relation. Then for every DL axiom ϕ, $\mathfrak{D} \models^{\mathcal{C}_{\mathfrak{D}}}_{\preceq,Ske} \phi$ if and only if $\mathfrak{D}_i \models \phi$, for all $\mathfrak{D}_i \in \mathsf{MaxCons}(\mathfrak{D})$.*

Proposition 3 tells us that the skeptical consequence relation based on binaricity preference relation is the same as the consequence relation based on maximal consistent sub-bases $w.r.t$ terminology.

In the case that many preferred recoveries exist (see Example 4), the skeptical consequence relation is rather weak (i.e., only few DL axioms can be inferred). Therefore, we propose another consequence relation.

Definition 9. *Given a DKB $\mathfrak{D}$ and a forgetting context $\mathcal{C}_{\mathfrak{D}}$ for it, let $\preceq$ be a preference relation on $\mathcal{R}_{\mathcal{C}_{\mathfrak{D}}}(\mathfrak{D})$. Let ϕ be a DL axiom and $\neg\phi$ is its negation*[2]*. Then ϕ is said to be argumentatively inferred from $\mathfrak{D}$ $w.r.t.$ $\preceq$, denoted by $\mathfrak{D} \models^{\mathcal{C}_{\mathfrak{D}}}_{\preceq,Arg} \phi$, if and only if there exists a $\overrightarrow{V} \in \overrightarrow{V}^{\preceq}_{\mathfrak{D},\mathcal{C}_{\mathfrak{D}}}$ such that $\mathfrak{D}|\overrightarrow{V} \models \phi$ and there does not exist $\overrightarrow{V}' \in \overrightarrow{V}^{\preceq}_{\mathfrak{D},\mathcal{C}_{\mathfrak{D}}}$ such that $\mathfrak{D}|\overrightarrow{V}' \models \neg\phi$. ϕ is called an argumentative consequence of $\mathfrak{D}$.*

A DL axiom is an argumentative consequence if and only if it can be inferred from a most preferred recovery and its negation cannot be inferred from any most preferred recoveries. Therefore, the argumentative consequence relation will not result in contradictory conclusions. Note that our argumentative consequence relation is different from consequence relation defined in the logic-based framework for argumentation [3] because a most preferred recovery can be viewed as maximal recovered consistent information in the DKB whilst an argument in an argumentation theory is a minimal consistent subset that can infer a conclusion. Unlike an argument in the argumentation theory, each axiom in a most preferred recovered is self-justified. A weakness of the argumentative consequence relation is that it may be too adventurous because an argumentative consequence may not be inferred by other most preferred recoveries (also its negation will not be inferred). The relationship between this consequence relation and the skeptical consequence relation is summarized by the following proposition.

Proposition 4. *Given a DKB $\mathfrak{D}$ and a forgetting context $\mathcal{C}_{\mathfrak{D}}$ for it, suppose ϕ is a DL axiom. If $\mathfrak{D} \models^{\mathcal{C}_{\mathfrak{D}}}_{\preceq,Ske} \phi$, then $\mathfrak{D} \models^{\mathcal{C}_{\mathfrak{D}}}_{\preceq,Arg} \phi$.*

That is, every skeptical consequence of a DKB is an argumentative consequence.

Example 6. (Example 5 Continued) It is easy to check that, for $i = 3, 4, 5$, $\mathfrak{D}|\overrightarrow{V}_i \models \mathsf{ResearchAssistant}(Mary)$ because $\mathsf{PhDStudent}(Mary) \in \mathcal{A}_1$ and $\mathsf{PhDStudent} \sqsubseteq \mathsf{ResearchAssistant} \in \mathsf{Forget}(V_i, \mathcal{T}_1)$. Since $\mathsf{ResearchAssistant} \notin \mathsf{Con}(\mathsf{Forget}(V_1, \mathcal{T}_1))$ and $\mathsf{ResearchAssistant} \notin \mathsf{Con}(\mathcal{T}_2)$, we must have that $\mathfrak{D}|\overrightarrow{V}_1 \not\models \neg\mathsf{ResearchAssistant}(Mary)$. Therefore, $\mathfrak{D} \models^{\mathcal{C}_{\mathfrak{D}}}_{\preceq,Arg} \mathsf{ResearchAssistant}(Mary)$.

5 Related Work

To handle inconsistency in DDLs, some special interpretation, called a *hole*, is proposed to interpret locally inconsistent ontologies in DDLs [4, 19]. In DDL, a *hole* for an ontology is an interpretation $\mathcal{I}^{\epsilon} = \langle \varnothing, \cdot^{\epsilon} \rangle$, where the domain is empty. It is proposed to deal with inconsistency. By doing this, even if a local ontology is inconsistent, there still exists a global model for the distributed knowledge base. However, if a hole is applied to a local ontology, it is equivalent to say that this ontology and all related bridge rules are removed from the distributed knowledge bases (see discussions in [4, 19]). This all-or-nothing semantics is clearly not very advisable. Furthermore, when the inconsistency is caused by several connected local ontolgoies via mapping, it is not clear how to apply the hole interpretation to deal with the inconsistency. Although it is possible to translate an inconsistent DKB to a DL knowledge base and then apply the

[2] The notion of axiom negation is defined in [9]

paraconsistent semantics given in [14] or [11] to infer non-trivial conclusions, this approach does not differentiate bridge rules and axioms in local ontologies and is different from our approach.

In [6], the authors deal with the problem of inconsistency in DDL by removing some bridge rules which are responsible for the inconsistency. The idea of their approach is similar to that of the approaches for debugging and repairing terminology in a centralized ontology. Our work takes a different view on inconsistency handling in DDL that the bridge rules are reliable and we weaken the axioms in local ontologies to restore consistency.

In [7], the notion of forgetting is given for a class of OWL/RDF(S) ontologies. The definition of forgetting is semantically meaningful. However, their definition of forgetting has some restriction on axioms in the ontologies. Moreover, the computational complexity of the algorithm for computing the forgetting literals may be exponential in the worst case [8]. In contrast, our definition of forgetting does not have any restriction on the axioms in an ontology and it can be computed in polynomial time. Although we do not have corresponding model-theoretical definition for our notion of forgetting, we justify our definition by analyzing its logical properties. Also, the focus of this paper is to apply the notion of forgetting to deal with inconsistency in DDL knowledge bases.

6 Conclusions and Future Work

In this paper, a forgetting-based approach for reasoning with inconsistent distributed ontologies was proposed. We first defined the notion of concept forgetting in DL $\mathcal{ALC}$. Some properties of the concept forgetting were given. The definitions of recoveries and preferred recoveries in propositional logic setting [12] were adapted to DDL. Based on the preferred recoveries, two consequence relations on inconsistent D-KB were defined. One is called skeptical consequence and the other is called argumentative consequence. We showed that every skeptical consequence of a DKB is an argumentative consequence. Our definition of concept forgetting does not have a corresponding model theoretic semantics. One of the reasons that prevent us from defining concept forgetting by simply extending variable forgetting is that we allow value restrictions on role names and existential restriction on role names. We will consider a less expressive DL that allow neither value restrictions on role names nor existential restriction on role names and see if we can define a semantic meaningful notion of concept forgetting. We will also provide discuss complexity issues of our approach in another work.

Acknowledgments

This work is partially supported by the EU under the IST project NeOn (IST-2006-027595, http://www.neon-project.org/)

References

1. Franz Baader, Diego Calvanese, Deborah McGuinness, Daniele Nardi, and Peter Patel-Schneider. *The Description Logic Handbook: Theory, implementation and application*. Cambridge University Press, 2003.

2. Jie Bao, Doina Caragea, and Vasant Honavar. On the semantics of linking and importing in modular ontologies. In *Proc. of ISWC'06*, pages 72–86, 2006.
3. Philippe Besnard and Anthony Hunter. A logic-based theory of deductive arguments. *Artif. Intell.*, 128(1-2):203–235, 2001.
4. Alexander Borgida and Luciano Serafini. Distributed description logics: Assimilating information from peer sources. *J. Data Semantics*, 1:153–184, 2003.
5. Paolo Bouquet, Fausto Giunchiglia, Frank van Harmelen, Luciano Serafini, and Heiner Stuckenschmidt. Contextualizing ontologies. *J. Web Sem.*, 1(4):325–343, 2004.
6. Andrei Tamilin Christian Meilicke, Heiner Stuckenschmidt. Repairing ontology mappings. In *Proc. of AAAI'07*, pages 1408–1413. 2007.
7. Thomas Eiter, Giovambattista Ianni, Roman Schindlauer, Hans Tompits, and Kewen Wang. Forgetting in managing rules and ontologies. In *Web Intelligence*, pages 411–419, 2006.
8. Thomas Eiter and Kewen Wang. Forgetting and conflict resolving in disjunctive logic programming. In *Proc. of AAAI'06*, 2006.
9. Giorgos Flouris, Zhisheng Huang, Jeff Z. Pan, Dimitris Plexousakis, and Holger Wache. Inconsistencies, negations and changes in ontologies. In *Proc. of AAAI'06*, pages 1295–1300, 2006.
10. Peter Haase, Frank van Harmelen, Zhisheng Huang, Heiner Stuckenschmidt, and York Sure. A framework for handling inconsistency in changing ontologies. In *Proc. of ISWC'05*, pages 353–367, 2005.
11. Zhisheng Huang, Frank van Harmelen, and Annette ten Teije. Reasoning with inconsistent ontologies. In *Proc. of IJCAI'05*, pages 254–259. 2005.
12. Jérôme Lang and Pierre Marquis. Resolving inconsistencies by variable forgetting. In *Proc. of KR'02*, pages 239–250, 2002.
13. F. Lin and R. Reiter. Forget it! In *AAAI Fall Symposium on Relevance*, 1994.
14. Yue Ma, Pascal Hitzler, and Zuoquan Lin. Algorithms for paraconsistent reasoning with owl. In *Proc. of ESWC'07*, pages 399–413, 2007.
15. Bijan Parsia and Bernardo Cuenca Grau. Generalized link properties for expressive epsilon-connections of description logics. In *Proc. of AAAI'05*, pages 657–662, 2005.
16. Bijan Parsia, Evren Sirin, and Aditya Kalyanpur. Debugging OWL ontologies. In *Proc. of WWW'05*, pages 633–640, 2005.
17. Stefan Schlobach and Ronald Cornet. Non-standard reasoning services for the debugging of description logic terminologies. In *Proc. of IJCAI'03*, pages 355–362. Morgan-Kaufmann, 2003.
18. Manfred Schmidt-Schaußand Gert Smolka. Attributive concept descriptions with complements. *Artificial Intelligence*, 48(1):1–26, 1991.
19. Luciano Serafini, Alexander Borgida, and Andrei Tamilin. Aspects of distributed and modular ontology reasoning. In *Proc. of IJCAI'05*, pages 570–575, 2005.