

LABORATOIRE



INFORMATIQUE, SIGNAUX ET SYSTÈMES
DE SOPHIA ANTIPOLIS
UMR 6070

COMBINING REVISION PRODUCTION RULES AND DESCRIPTION LOGICS

Chan LE DUC, Nhan LE THANH

Projet MECOSI

Rapport de recherche
ISRN I3S/RR-2003-03-FR

Février 2003

RÉSUMÉ :

Les connaissances des domaines d'application sont souvent hétérogènes et évoluées dans le temps. Par conséquent, une combinaison de plusieurs formalismes dans laquelle les opérations de révision sont définies, est nécessaire pour la représentation de tel système hybride. Dans ce papier, après une introduction des différentes politiques de révision de bases de connaissances basées sur les DL, nous montrons comment évaluer les opérateurs de révision dans le langage ALE de DL. A partir de ces résultats, nous définissons le concept de règle de production par révision, dont la conséquence est un opérateur de révision. Ces règles permettront de représenter les règles contextuelles nécessaires dans la translation entre les ontologies basées sur les DL. Enfin, nous abordons la sémantique formelle de ce formalisme combinant les règles de production par révision et les logiques de descriptions (ALE)

MOTS CLÉS :

Logiques de description, politique de révision, modèles hybrides, règles de production

ABSTRACT:

Knowledges of every application area are heterogeneous and they develop in time. Therefore, a combination of several formalisms on which revision operations are defined, is required for knowledge representation of such a hybrid system. In the paper, we will introduce several revision policies in DL-based knowledge bases and show how revision operators are computed for DL language ALE. From this, we define revision production rules consequent of which is a revision operator. Such rules allow us to represent context rules in translation between DL-based ontologies. Finally, we will introduce formal semantics of the formalism combining revision production rules and Description Logics (ALE)

KEY WORDS :

Description Logics, Revision policies, Hybrid Models, Production Rules

Combining Revision Production Rules and Description Logics

Chan Le Duc, Nhan Le Thanh
Email: cleduc@i3s.unice.fr, Nhan.Le-Thanh@unice.fr

Laboratoire I3S, Université de Nice - Sophia Antipolis, France

Abstract. Knowledges of every application area are heterogeneous and they develop in time. Therefore, a combination of several formalisms on which revision operations are defined, is required for knowledge representation of such a hybrid system. In this paper, we will introduce several revision policies in DL-based knowledge bases and show how revision operators are computed for DL language $\mathcal{AL}\mathcal{E}$. From this, we define revision production rules consequent of which is a revision operator. Such rules allow us to represent context rules in translation between DL-based ontologies. Finally, we will introduce formal semantics of the formalism combining revision production rules and Description Logics ($\mathcal{AL}\mathcal{E}$).

1 Introduction

Description Logics can be used as a formalism for design of ontologies of an application domain. In order to reduce the size of ontologies, different ontologies for different subdomains or user profiles are derived from a shared common ontology. These derived ontologies share atomic concepts, atomic roles and defined basic concepts in the common ontology. The shared concepts can be redefined in a derived ontology with the aim of fitting an adaptable context. This is necessary to determine the more sufficient meaning of a shared concept in the context of current users. The redefining can be performed in run-time owing to context rules. The antecedent of a context rule is a predicate which describes context condition and its consequent is an operation which translates a concept definition into another. For this reason, we need an other formalism to capture the semantics of such ontologies. It is specific production rules, namely *revision production rules*, whose consequent is an operation allowing one to change a shared concept definition.

In previous hybrid languages, for example CARIN in [4], it combines Horn rules and Description Logics to obtain a formalism the expressiveness of which is significantly improved. However, it seems that these languages could not capture context rules since the consequent of a Horn rule is still a predicate.

On the other hand, since the definition of a concept in knowledge base can be modified by the application of a revision production rule, revision operations

must be defined on DL-based terminology. A profound study and a framework for revision operations given in [1] together provide a good background for the extension of these operations to more expressive DL languages. Nevertheless, there are several different viewpoints concerning the revision of DL-based knowledge base which argue both for and against the conservation of subsumption relationships in the terminological hierarchy. One viewpoint considers subsumption relationships as literal and invariant knowledges. That is, the subsumption relationships have never been changed despite a change in concept definitions. Another viewpoint states that subsumption relationships are derived knowledges and that they may be changed when concept definitions are modified. It is clear that these viewpoints lead to revision policies the complexities of which are different.

Previous works on how to define revision operations have concentrated on operations TELL (to add a concept description fragment to a concept definition), FORGET (to delete a concept description fragment from the concept definition), and on simple languages which do not permit existential restrictions [1]. Moreover, operation FORGET requires that a deleted fragment must be a part of the concept definition. Thus, the main contributions of this paper are, first, to propose several revision policies some of which require the definition of a new operation, namely PROPAG, in order to conserve subsumption relationships. Second, we propose to extend revision operations to language $\mathcal{AL}\mathcal{E}$. This extension will allow us to define the semantics of a formalism combining revision production rules and language $\mathcal{AL}\mathcal{E}$.

The paper will begin with revision policies in which a framework for revision operators will be given. The next section will concentrate on the computing of revision operators for language $\mathcal{AL}\mathcal{E}$ by using the structural characterization of subsumption developed in [8]. The last section will introduce a formal definition of revision production rules, before focusing on the semantics of the combining formalism.

2 Knowledge Base Revision and Revision Policies

2.1 Preliminaries

Revision Operators Informally, TELL is interpreted as an operation which adds a knowledge to KB, whereas FORGET deletes a knowledge from KB. If we use a DL language which allows us both to normalize concept descriptions and rewrite concept definitions as conjunctions of simple concepts, the revision operators are more formally written as follows :

- $\text{TELL}(\Delta, C, Expr) = C \sqcap Expr$
- $\text{FORGET}(\Delta, C \sqcap Expr) = C$

where Δ is knowledge base (KB), C is concept to be revised and $Expr$ is a simple concept.

In order to have reasonable definitions of the revision operators, it is paramount that several constraints are respected. However, if all of these constraints are to be respected, certain problems are inevitable. In the following subsection, we attempt to identify some of important constraints for the revision operators.

Criteria for Revision The majority of these criteria is proposed and discussed in [1]. We only recall and reformulate some of them for the purposes of this paper.

1. *Adequacy of the Request Language.* The request language should be interpretable in the formalism the KB uses.
2. *Closure.* Any revision operation should lead to a state of the KB representable by the formalism used.
3. *Success.* The criterion requires that any revision operation must be successful, *i.e.* after a TELL the new knowledge ($Expr$) must be derivable from the KB and after a FORGET the deleted knowledge ($Expr$) must not be derivable from the KB no longer. More formally, there exists an interpretation \mathcal{I} and an individual $a^{\mathcal{I}} \in \mathcal{O}^{\mathcal{I}}$ domain, such that $a^{\mathcal{I}} \in C^{\mathcal{I}}$ and $a^{\mathcal{I}} \notin (TELL(C, Expr))^{\mathcal{I}}$. Similarly, there exists an interpretation \mathcal{I} and an individual $a^{\mathcal{I}} \in \mathcal{O}^{\mathcal{I}}$ domain, such that $a^{\mathcal{I}} \notin C^{\mathcal{I}}$ and $a^{\mathcal{I}} \in (FORGET(C, Expr))^{\mathcal{I}}$.
4. *Minimal Change.* This criterion requires that revision operations cause a minimal change of the KB. It is obvious that it needs a definition concerning the distance between two KBs. Therefore, a pragmatic consideration is necessary in each real application.
5. *Subsumption Conservation.* The terminological subsumption hierarchy of the KB should be conserved by any revision operation.

A priori, criteria 4 and 5 are difficult to guarantee simultaneously since subsumption conservations may change the majority of the hierarchy.

2.2 Revision Policies

In the previous subsection, we introduced the general principles for revision on DL-based knowledge base. In certain cases there may exist one principle that we cannot entirely adhere to. We now present pragmatic approaches in which some of these criteria are respected while the others are not. In order to facilitate the choice of policy made by the user, we will attempt to identify the advantages and disadvantages. The first policy takes into account criterion 5 (*Subsumption Conservation*) whereas the second policy allows us to change subsumption hierarchy. The third policy can be considered as a compromise between the first and the second.

Conservative Policy This policy allows us to modify definition of a concept C (concept description) by adding or deleting a fragment to or from the definition. Concept C and all concepts defined via C now have a new definition. However, the modification of C should not violate any subsumption relationship. By that we mean, if $A \sqsubseteq B$ holds before the modification for some concepts A, B in the Tbox, then $A \sqsubseteq B$ still holds after the modification. This requirement is in accordance with the viewpoint in which the subsumption relationships are considered as invariant knowledges of the KB. This condition is clearly compatible with the object-oriented database model provided that the hierarchical structure of classes is persistent, although class definitions may be changed. In this model, each time a class definition is changed, all existent instances of the old class must remain the instances of the new class. In fact, the evolution of our understanding about the world leads to more precise definitions. Also, the objects held for the old definition are now described more sufficiently by the new definition. In other words, we need to change the definition of a concept C while all subsumption relationships are conserved and all assertions $C(a)$ (as well as all $D(a)$ where D is changed because of the modification propagation) for the new concept C remains held as well. This can lead to some incompatibilities between the criteria for inference definitions in the open world.

Indeed, we suppose that revision operator TELL is defined as follows : $TELL(C, Expr) = C \sqcap Expr$. According to the assertion conservation proposed by this policy, if $C^{\mathcal{I}}(a^{\mathcal{I}})$ holds then $(TELL(C, Expr))^{\mathcal{I}}(a^{\mathcal{I}})$ holds as well for all interpretation \mathcal{I} . This means that $C \sqsubseteq TELL(C, Expr)$ for all interpretation \mathcal{I} . Hence $C \equiv TELL(C, Expr)$ since $TELL(C, Expr) \sqsubseteq C$ for all interpretation \mathcal{I} . This contradicts criterion 3 (*success*).

In order to avoid this incompatibility, an addition of the epistemic operator is required [5]. We denote W_1 a set of interpretations of the KB before the revision. In order to respect criterion 3, the subsumption $C \sqsubseteq TELL(C, Expr)$ must only hold on a set W_2 of interpretations where $W_2 \subsetneq W_1$. By that we mean, the assertion conservation makes less models possible for the KB. As a consequence, all inferences on the KB must be defined in a closed world.

We now try to provide a framework for the revision operators. Suppose that Tbox is acycle and does not contain general inclusions. Since the subsumption relationship conservation is required in this policy, a modification on C can be propagated in all Tbox. In the following, we will investigate this propagation for each operator TELL and FORGET. The definitions of TELL, FORGET and PROPAG with respect to criteria 3, 4 (*success* and *minimal change*) for a concrete language are investigated in the following section.

- $TELL(C, Expr)$. Obviously, this operation causes the semantic modifications on all concepts D which contain concept C in its definition since concepts D become more specific. On the other hand, it is possible that concepts not containing concept C can be modified as well because of the conservation of the subsumption relationships. The unfolded definition of D containing C is

written as $\tilde{D}(C)$ and the unfolded definition of D no containing C is written as $\tilde{D}(\bar{C})$. $\tilde{D}(C)$ and $\tilde{D}(\bar{C})$ can be obtained by replacing all concept names by their definitions except concept C . Let E be a *direct super* concept of F in the hierarchy. There are two cases as follows : i) if $E = \tilde{E}(\bar{C})$ then relationship $F \sqsubseteq E$ is automatically conserved since E is not changed and F may be not changed or become more specific. ii) if whether $F = \tilde{F}(\bar{C})$ and $E = \tilde{E}(C)$ or $F = \tilde{F}(C)$ and $E = \tilde{E}(C)$, we need a new operator, namely PROPAG_TELL, that allows us to compute a new definition of F such that the subsumption relationship is conserved. Operator PROPAG_TELL should be defined with respect to criteria 4 in some way. Note that the modification is started at concept TOP of the hierarchy and propagated forward concept BOTTOM. Hence, the propagation can influence every concept.

- FORGET($C, Expr$). This operator is, in fact, computed via an other operator, namely MINUS, that can be interpreted as a deletion of $Expr$ from C . In the case where $C = C' \sqcap Expr$, we must have $MINUS(C, Expr) = C'$. Similar as TELL, the modification made by FORGET may be propagated in all Tbox. An operator PROPAG_FORGET should also be defined in order to propagate the modification backwards concept TOP.

Revision of Literal Definition This revision policy had been proposed in [1]. The main idea of this policy is to allow us to modify the definition of a concept without modifying subsumption relationships. In addition, this policy allows us also to revise the Abox with respect to the new Tbox because the new Tbox can cause inconsistencies in the Abox. This policy is convenient for the case in which we need a correction of a misunderstanding in terminology design. The individuals of a “bad” definition now belong to a better definition. The misunderstanding correction costs a modification of subsumption relationships. In fact, in this approach concept definitions are taken into account as literal knowledges while subsumption knowledges are considered as derived knowledges. As a result, a restructuration of the concept hierarchy can be required after a revision. Note that this policy does not apply operator PROPAG since subsumption conservation is omitted.

As stated in the previous policy, we suppose that Tbox is acycle and does not contain general inclusions. Operators TELL($C, Expr$) and FORGET($C, Expr$) are defined owing to operator MINUS described as above. These operators should respect also criteria 3, 4 (*success* and *minimal change*).

- Revision on Abox. An important task in this policy is to revise the ABox when the Tbox is changed. All assertions $D(a)$ where D is defined directly or indirectly via C must be revised. It is clear that operator FORGET does not cause any inconsistency in the Abox since if $F(C)(a)$ holds where $F(C)$ is a concept description via C , then FORGET($F(C)(a), Expr$) holds as well. In contrast, there may be an inconsistency in the Abox after TELL($C, Expr$). In this case, the revision on Abox requires an inference which is based on

the added part $Expr$. In particular, $D(a)$ holds for some concept $D = F(C) = C$, iff $Expr(a)$ holds.

- Restructuration of the subsumption hierarchy. Another problem that arising is how much a restructuration of the subsumption hierarchy from the old hierarchy costs following a revision. Note that this policy does not apply operator PROPAG to conserve subsumption relationships. Since operator PROPAG is not more expensive than subsumption checking, the restructuration is not likely to increase the complexity.

Mixed Policy As presented above, this policy can be considered as a compromise between two policies presented, namely *mixed policy*. It allows us to modify the definition of a concept while respecting all subsumption relationships. Moreover, following a revision on the Tbox, a revision on the Abox is necessary. By that we mean that this revision policy conserves models for the knowledge base and we do not need to add the epistemic operator to the knowledge base. In other words, all inferences remain to be defined in open world.

Technically, this policy does not require any additional particular operator in comparison to the two policies above. Operators TELL(C , $Expr$) and FORGET(C , $Expr$) are defined via operator MINUS. In order to conserve subsumption relationships, operator PROPAG must be applied. In addition, the recomputing of the Abox is also necessary after a revision on the Tbox. Briefly, the mixed policy requires more computing than the two previous policies.

3 Revision Operators for language $\mathcal{AL}\mathcal{E}$

The work in [8] introduced a structural characterization of subsumption for language $\mathcal{AL}\mathcal{E}$. This characterization facilitates defining and computing the revision operators. This work also proposed a way for computing the *Least Common Subsumer* in $\mathcal{AL}\mathcal{E}$ resulting from this characterization. This allows us to compute efficiently revision operators (PROPAG_FORGET) with respect to criterion 4 (*minimal change*).

3.1 Preliminaries

Let N_C be a set of primitive concepts and N_R be a set of primitive roles. Language $\mathcal{AL}\mathcal{E}$ uses the following constructors to build concept descriptions: conjunction ($C \sqcap D$), value restriction ($\forall r.C$), existential restriction ($\exists r.C$), primitive negation ($\neg P$), concepts TOP and BOTTOM.

Let Δ be a non-empty set of individuals. Let $\cdot^{\mathcal{I}}$ be a function that transforms each primitive concept $P \in N_C$ into $P^{\mathcal{I}} \subseteq \Delta$ and each primitive role $r \in N_R$ into $r^{\mathcal{I}} \subseteq \Delta \times \Delta$. The semantics of a concept description are inductively defined owing to the interpretation $\mathcal{I} = (\Delta, \cdot^{\mathcal{I}})$ in the table below.

Syntax	Semantics
\top	Δ
\perp	\emptyset
$C \sqcap D$	$C^I \cap D^I$
$\forall r.C, r \in N_R$	$\{x \in \Delta \mid \forall y:(x,y) \in r^I \rightarrow y \in C^I\}$
$\exists r.C, r \in N_R$	$\{x \in \Delta \mid \exists y:(x,y) \in r^I \wedge y \in C^I\}$
$\neg P, P \in N_C$	$\Delta \setminus P^I$

Least common subsumer (lcs) [8]

Subsumption Let C, D be concept descriptions. D subsumes C , $C \sqsubseteq D$, iff $C^I \subseteq D^I$ for all interpretations \mathcal{I} .

Least Common Subsumer Let C_1, C_2 be concept descriptions in a DL language. The least common subsumer C is a least common subsumer of C_1, C_2 (for short $lcs(C_1, C_2)$) iff

1. $C_i \sqsubseteq C$ for all i , and
2. C is the least concept description which has this property, i.e if C' is a concept description such that $C_i \sqsubseteq C'$ for all i , then $C \sqsubseteq C'$.

According to [8], lcs of two or more descriptions always exists for language $\mathcal{AL}\mathcal{E}$ and the authors proposed an exponential algorithm for computing it.

Language $\mathcal{AL}\mathcal{E}$ and structural subsumption characterizing

Normalization of $\mathcal{AL}\mathcal{E}$ -concept description [8]

An $\mathcal{AL}\mathcal{E}$ -concept description is called *propagated normal* if the following rules are exhaustively applied

- i) $\forall r.E \sqcap \forall r.F \rightarrow \forall r.(E \sqcap F)$ ii) $\forall r.E \sqcap \exists r.F \rightarrow \forall r.E \sqcap \exists r.(E \sqcap F)$
- iii) $\forall r.\top \rightarrow \top$ iv) $E \sqcap \top \rightarrow E$ v) $P \sqcap \neg P \rightarrow \perp$ for each $P \in N_C$ vi) $\exists r.\perp \rightarrow \perp$
- vii) $E \sqcap \perp \rightarrow \perp$

where E, F are two $\mathcal{AL}\mathcal{E}$ -concept descriptions and $r \in N_R$

Description tree of an $\mathcal{AL}\mathcal{E}$ -concept description [8]

A description tree of $\mathcal{AL}\mathcal{E}$ -concept description is a tree of the form $\mathcal{G} = (N, E, n_0, l)$ where

- N is a finite set of nodes of \mathcal{G} ;
- $E \subseteq N \times (N_R \cup \forall N_R) \times N$ is a finite set of edges labeled with role names r (\exists -edges) or with $\forall r$ (\forall -edges); $\forall N_R := \{\forall r \mid r \in N_R\}$;
- n_0 is the root of \mathcal{G} ;
- l is a labeling function mapping the nodes in N to finite set $\{P_1, \dots, P_k\}$ where each P_i , $1 \leq i \leq k$, is one of the following forms : $P_i \in N_C$, $P_i = \neg P$ for some $P \in N_C$, or $P_i = \perp$. The empty label corresponds to the concept TOP.

Homomorphisms between $\mathcal{AL}\mathcal{E}$ -description trees [8]

A mapping $\varphi : N_H \rightarrow N_G$ from an $\mathcal{AL}\mathcal{E}$ -description tree $\mathcal{H} = (N_H, E_H, m_0, l_H)$ to an $\mathcal{AL}\mathcal{E}$ -description tree $\mathcal{G} = (N_G, E_G, n_0, l_G)$ is called homomorphism iff, the following conditions are satisfied :

1. $\varphi(m_0) = n_0$
2. For all $n \in N_H$, we have $l_H(n) \subseteq l_G(\varphi(n))$ or $\perp \in l_G(\varphi(n))$;
3. For all $nrm \in E_H$, either $\varphi(n)r\varphi(m) \in E_G$, or $\varphi(n) = \varphi(m)$ and $\perp \in l_G(\varphi(n))$; and
4. For all $n\forall rm \in E_H$, either $\varphi(n)\forall r\varphi(m) \in E_G$, or $\varphi(n) = \varphi(m)$ and $\perp \in l_G(\varphi(n))$.

Theorem for structural subsumption [8]

Let C, D be $\mathcal{AL}\mathcal{E}$ -concept descriptions. Then, $C \sqsubseteq D$ if and only if there exists a homomorphism from \mathcal{G}_D^T to \mathcal{G}_C .

Simple concept descriptions in Tbox In a Tbox, a concept definition can depend on other concept definitions. In other words, the right side of a concept definition can contain a defined concept name. For a static KB on which revision operators do not exist, we can transform an acycle TBox into the *equivalent* unfolded Tbox in which all right sides do not contain concept names. The equivalence is interpreted as follows: for any concept, its definitions in the two Tboxes are equivalent. Nevertheless, this is not true for a KB on which revision operators are defined. For example :

Let a Tbox $\mathcal{T}_1 = \{A := P \sqcap B; B := Q \sqcap R\}$.

and its unfolded Tbox : $\mathcal{T}_2 = \{A := P \sqcap Q \sqcap R; B := Q \sqcap R\}$.

If we apply FORGET(\mathcal{T}_1, B, Q) and FORGET(\mathcal{T}_2, B, Q), then \mathcal{T}'_1 and \mathcal{T}'_2 thus obtained are no longer equivalent. It is obvious that what we want to obtain after the revision is \mathcal{T}'_1 since concept A remains be defined as a conjunction of P and B . This means that the dependence must be conserved. For this reason, we cannot define simple concept as a conjunct in the propagated normal form. In what following, we propose a simple concept definition which fits the idea above.

Simple concept description

An $\mathcal{AL}\mathcal{E}$ -concept description is called *simple* if it does not contain any conjunction on the *top-level*. A concept description is called *normal* if it is a conjunction of simple concept descriptions. We denote \mathcal{S}_C as a set of simple concept descriptions of concept C in a TBox. In the example above, we have $\mathcal{S}_A = \{P, B\}$, $\mathcal{S}_B = \{Q, R\}$

Note that a simple concept can be considered part of the meaning of a concept. As a consequence, the sets of simple concepts of the equivalent concepts can be different. A discussion in detail of this subject is given in [1].

3.2 Operator TELL

Operator TELL

Let $C_1 \sqcap \dots \sqcap C_n$ be of the *normal form* of C .
The operator TELL is defined as follows :
 $\text{TELL}(C, Expr) = D$ where $Expr$ is simple concept description and,
If $C \sqsubseteq Expr$ then $D := C$
Otherwise, $D := C \sqcap Expr$ and $\mathcal{S}_D = \mathcal{S}_C \cup \{Expr\}$

If the added knowledge is derivable from C before the execution of TELL *i.e* $C \sqsubseteq Expr$, no more remains to be done and criterion 3 (success) is automatically satisfied. Otherwise, we need to guarantee that the added knowledge is derivable from C after the execution of TELL. In fact, since $\text{TELL}(C, Expr) = C \sqcap Expr$, we have $C \sqsubseteq Expr$. Obviously, if $C \not\sqsubseteq Expr$, this definition of operator TELL will satisfy criteria 3 (*success*) and 4 (*minimal modification*) as well.

Operator PROPAG_TELL First, we recall the notions presented above. Let \tilde{E} be the unfolded definition of E which can be obtained by replacing all concept names by its definition. If the unfolding meets C then C is not unfolded and \tilde{E} is written as $\tilde{E}(C)$ *i.e* E is defined directly or indirectly via C . Otherwise, if the unfolding does not meet C , \tilde{E} is written as $\tilde{E}(\bar{C})$ *i.e* E does not depend on C . In the same way, we can write $E(C)$, $E(\bar{C})$, for original definitions. We denote $E(C, D)$, $\tilde{E}(C, D)$ as concept descriptions where each occurrence of C in original, unfolded forms, respectively, is replaced by D . Let E be a *direct super* concept of a concept F in the hierarchy. Operator PROPAG_TELL($E, F, Expr$) will compute a new concept description of F while taking into account criterion 4 (*minimal change*) and $F \sqsubseteq E$.

If $E = \tilde{E}(\bar{C})$ then relationship $F \sqsubseteq E$ is automatically conserved, and no more remains to be done. In the case $E = \tilde{E}(C)$, we denote $\mathcal{S}_E^0(C)$ as a set of all $s_E \in \mathcal{S}_E$ such that s_E depends on C , denoted by $s_E(C)$, and $\mathcal{S}_E^1(\bar{C})$ as a set of all $s_E \in \mathcal{S}_E$ such that s_E does not depend on C , denoted by $s_E(\bar{C})$. Operator PROPAG_TELL($E, F, Expr$) performs, first, the replacement of C with $(C \sqcap$

$Expr$) in all $s_E(C)$. Secondly, it adds $s_E(C, C \sqcap Expr)$ to F as new conjuncts *i.e*

$$\text{PROPAG_TELL}(E, F, Expr) := \tilde{F}(C, C \sqcap Expr) \prod_{s_E \in S_E^0(C)} s_E(C, C \sqcap Expr)$$

As a consequence, $F \sqsubseteq E$ still holds after an application of $\text{PROPAG_TELL}(E, F, Expr)$.

Note that $\text{PROPAG_TELL}(E, F, Expr)$ adds to \mathcal{S}_F the terms $s_E(C, C \sqcap Expr)$ where $s_E \in \mathcal{S}_E$. This is reasonable because $\text{PROPAG_TELL}(E, F, Expr)$ must be subsumed by these terms in any case. Moreover, $\text{PROPAG_TELL}(E, F, Expr)$ is the most concept description which is less than $\tilde{F}(C, C \sqcap Expr)$ and $\tilde{E}(C, C \sqcap Expr) = \prod_{s_E \in S_E^1(\bar{C})} s_E(\bar{C}) \sqcap \prod_{s_E \in S_E^0(C)} s_E(C, C \sqcap Expr)$. Indeed, $\text{PROPAG_TELL}(E, F, Expr)$ is the intersection of these concepts since $\tilde{F}(C, C \sqcap Expr) \sqsubseteq \tilde{F}(C) \sqsubseteq \prod_{s_E \in S_E^1(\bar{C})} s_E(\bar{C})$.

Example 1

Let $\mathcal{T}_1 = \{ C := \exists r.Q, E := \exists r.C, F := \exists r.\exists r.(Q \sqcap R) \}$ and $Expr := \exists r.P$.

We have $F \sqsubseteq E$ and $E = \tilde{E}(C), F = \tilde{F}(\bar{C}); \mathcal{S}_E = \{\exists r.C\}, \mathcal{S}_F = \{\exists r.\exists r.(Q \sqcap R)\}$

Note that $\tilde{F}(C, C \sqcap Expr) \not\sqsubseteq \tilde{E}(C, C \sqcap Expr)$ since $\tilde{E}(C, C \sqcap Expr) = \exists r.(\exists r.Q \sqcap \exists r.P)$ and $\tilde{F}(C, C \sqcap Expr) = \exists r.\exists r.(Q \sqcap R)$.

We have $\text{PROPAG_TELL}(E, F, Expr) = \exists r.\exists r.(Q \sqcap R) \sqcap \exists r.(C \sqcap \exists r.P)$ and $\mathcal{S}_{\text{PROPAG_TELL}(E, F, Expr)} = \{\exists r.\exists r.(Q \sqcap R), \exists r.(C \sqcap \exists r.P)\}$

□

3.3 Operator FORGET

This operator $\text{FORGET}(C, Expr)$ here defined, allows for $Expr \notin \mathcal{S}_C$. This extension is necessary because in some cases we need to forget a term $Expr$ from C where $C \sqsubseteq Expr$ and $Expr \notin \mathcal{S}_C$. Such a knowledge can be derived from C .

Operator MINUS between two $\mathcal{AL}\mathcal{E}$ -description trees

Let C, D be propagated normal $\mathcal{AL}\mathcal{E}$ -concept descriptions and $C \sqsubseteq D$. $\text{MINUS}(C, D)$ is defined as the remainder of \mathcal{G}_C description tree after the deletion of image $\varphi(\mathcal{G}_D)$ in \mathcal{G}_C for all homomorphisms φ from \mathcal{G}_D to \mathcal{G}_C . More formally, let $\mathcal{G}_C = (N_C, E_C, n_0, l_C)$ and $\mathcal{G}_D = (N_D, E_D, m_0, l_D)$. $\text{MINUS}(C, D) = (N_C, E_C, p_0, l'_C)$ where $l'_C(\varphi(m)) = l_C(\varphi(m)) \setminus l_D(m)$ for all $m \in N_D$ and all φ .

Example 2

Let E, F be concept descriptions in the *example 1*. We have $\text{MINUS}(F, E) = \exists r.\exists r.R$.

□

Proposition 3.1

Let C be a propagated normal $\mathcal{AL}\mathcal{E}$ -concept description, D be a propagated normal $\mathcal{AL}\mathcal{E}$ -simple concept description and $D = D_1 \sqcap \dots \sqcap D_n$. Moreover, $C \sqsubseteq D$. If $C = X \sqcap D$ and $X \not\sqsubseteq D_i$ for all i , then $\text{MINUS}(C, D) \equiv X$.

We denote \mathcal{S}_{C-D}^0 a set of all $s_C \in \mathcal{S}_C$ such that $s_C \not\sqsubseteq D_i$ for all i and \mathcal{S}_{C-D}^1 a set of all s'_C as the remainder of $s_C \notin \mathcal{S}_{C-D}^0$ following the operation MINUS. *Proposition 3.1* guarantees that $s_C \in \mathcal{S}_C^0$ are not changed. We have $\mathcal{S}_{\text{MINUS}(C,D)} = \mathcal{S}_{C-D}^0 \cup \mathcal{S}_{C-D}^1$.

We can now define operator FORGET.

Let $C_1 \sqcap \dots \sqcap C_n$ be of the normal form of C .
 Operator FORGET is defined as follows :
 $\text{FORGET}(C, Expr) = D$ where $Expr$ is a simple concept and,
 If $C \not\sqsubseteq Expr$ then $D := C$
 Otherwise, $D := \text{MINUS}(C, Expr)$ and $\mathcal{S}_D = \mathcal{S}_{C-Expr}^0 \cup \mathcal{S}_{C-Expr}^1$

If $C \not\sqsubseteq Expr$, *i.e* we may leave out the fragment $Expr$ as it does not contribute to the definition of C . In this case, there is nothing to do since C is *not derivable* from the knowledge to be forgotten. In fact, there exists an individual a such that $a^I \in C^I$ and $a^I \notin Expr^I$ where I is an interpretation. By that we mean, the knowledge $Expr$ is not derivable from the KB following the execution of operation FORGET. Therefore, the operator which is defined in such a way satisfies the criterion 3 (*success*)

If $C \sqsubseteq Expr$, operator MINUS ensures the unique existence of the result concept description. Hence, operator FORGET is well defined.

Also, operator MINUS guarantees the criterion 3 (*success*) as well. In fact, in order to explain more clearly we resort to operator TELL. If $\text{TELL}(C, Expr) = C$, *i.e* $Expr$ is derivable from C ($C \sqsubseteq Expr$), then no action is to be carried out. In this case, operator MINUS guarantees that $C \sqsubseteq D$, so the success of operator FORGET as well.

By contrast, if $\text{TELL}(C, Expr) = C \sqcap Expr$ ($C \not\sqsubseteq Expr$), then $\text{FORGET}(C \sqcap Expr, Expr) = C$ where $C \sqcap Expr \sqsubseteq C$ (this is deduced from *Proposition 3.1*). This guarantees at the same time the success of operator FORGET and the minimal change (*criterion 4*).

Operator PROPAG_FORGET Let E be a *direct super* concept of a concept F in the hierarchy. Operator PROPAG_FORGET($E, F, Expr$) will compute a new concept description of E while taking into account criterion 4 (*minimal change*). Computing *lcs* is, therefore, necessary for this aim.

If $F = \tilde{F}(C)$ then relationship ($F \sqsubseteq E$) is automatically conserved, therefore no more remains to be done. In the case $F = \tilde{F}(C)$, we define :

$\text{PROPAG_FORGET}(E, F, Expr) :=$
 $\text{lcs}\{\tilde{E}(C, \text{MINUS}(C, Expr)), \tilde{F}(C, \text{MINUS}(C, Expr))\}$

Computed in this way, $\text{PROPAG_FORGET}(E, F, Expr)$ is the least concept description such that it is more than $\tilde{E}(C, \text{MINUS}(C, Expr))$ and $\tilde{F}(C, \text{MINUS}(C, Expr))$.

We now compute $\mathcal{S}_{\text{PROPAG_FORGET}(E, F, Expr)}$ according to the definition above. Thus abbreviated, we denote $E_1 = \tilde{E}(C, \text{MINUS}(C, Expr))$ and $E_2 = \text{PROPAG_FORGET}(E, F, Expr)$. Since $E_1 \sqsubseteq E_2$, we can define an intersection of all images $\varphi(\mathcal{G}_{E_2})$ in \mathcal{G}_{E_1} : $K = \bigcap_{\varphi} \varphi(\mathcal{G}_{E_2})$ where φ is a homomorphism from \mathcal{G}_{E_2} to \mathcal{G}_{E_1} . We denote $\mathcal{S}_{E_1}^0$ a set of all $s_{E_1} \in \mathcal{S}_{E_1}$ such that $\mathcal{G}_{s_{E_1}} \subseteq K$ and $\mathcal{S}_{E_2}^1$ a set of all $s_{E_2} \in \mathcal{S}_{E_2}$ such that $\varphi(\mathcal{G}_{s_{E_2}}) \not\subseteq K$ where φ is some homomorphism from \mathcal{G}_{E_2} to \mathcal{G}_{E_1} . We define $\mathcal{S}_{E'} = \mathcal{S}_{E_1}^0 \cup \mathcal{S}_{E_2}^1$. It is not difficult to check that $\bigcap_{s \in \mathcal{S}_{E'}} s \equiv \text{PROPAG_FORGET}(E, F, Expr)$.

Example 3

Let $\mathcal{T}_1 = \{ C := \exists r.(P \sqcap Q) \sqcap \forall r.R, E := \exists r.S \sqcap \exists r.\exists r.(P \sqcap Q), F := \exists r.S \sqcap \exists r.C \}$ and $Expr := \exists r.P$.

We have $F \sqsubseteq E$ and $F = \tilde{F}(C)$, $E = \tilde{E}(C)$; $\mathcal{S}_E = \{\exists r.S, \exists r.\exists r.(P \sqcap Q)\}$, $\mathcal{S}_F = \{\exists r.S, \exists r.C\}$,

$\text{MINUS}(C, Expr) = \exists r.Q \sqcap \forall r.R$.

Note that $\tilde{F}(C, \text{MINUS}(C, Expr)) \not\sqsubseteq \tilde{E}(C, \text{MINUS}(C, Expr))$ since $\tilde{F}(C, \text{MINUS}(C, Expr)) = \exists r.S \sqcap \exists r.(\exists r.Q \sqcap \forall r.R)$ and

$\tilde{E}(C, \text{MINUS}(C, Expr)) = \exists r.S \sqcap \exists r.\exists r.(P \sqcap Q)$

We have $\text{PROPAG_FORGET}(E, F, Expr) = \exists r.S \sqcap \exists r.\exists r.Q$; $K = \{\exists r.S \sqcap \exists r.\exists r.Q\}$; $\mathcal{S}_{E_1}^0 = \{\exists r.S, \exists r.\exists r.Q\}$

and $\mathcal{S}_{E_2}^1 = \emptyset$.

As a consequence, $\mathcal{S}_{\text{PROPAG_FORGET}(E, F, Expr)} = \{\exists r.S, \exists r.\exists r.Q\}$

□

3.4 Revision on Abox

In the previous subsection, a concept can be considered as a conjunction of simple concepts. The revision operators on the Tbox add or delete some terms to/from these sets .

Let E be a modified concept definition in the Tbox. Let \mathcal{S}_E and $\mathcal{S}_{E'}$ be the sets of simple concepts of E , respectively, before and after a revision on the Tbox. Let \mathcal{I} be a model of the KB. For each assertion $E^I(a^I)$, revision on the Abox performs a checking of all assertions $s^I(a^I)$ where $s \in \mathcal{S}_{E'} \setminus \mathcal{S}_E$. If there is some $s^I(a^I)$ which is not verified, $E^I(a^I)$ becomes inconsistent and it is deleted from the Abox. Otherwise, if every $s^I(a^I)$ is verified, assertion $E^I(a^I)$ is conserved in the Abox.

4 Revision Production Rules and Description Logics

4.1 Revision Production Rules

The rule to be defined below is a specific form of production rule, namely *revision production rule*. In fact, a consequent of these rules is only an revision operator on a terminology. The general form of these rules in a KB $\Delta = (\Delta_{\mathcal{T}}, \Delta_{\mathcal{A}}, \Delta_{\mathcal{PR}}$) is given as follows :

$$r \in \Delta_{\mathcal{PR}}$$

$$r : p \Rightarrow \text{TELL}(\Delta_{\mathcal{T}}, \text{Concept}, \text{Expr}) \text{ or}$$

$$r : p \Rightarrow \text{FORGET}(\Delta_{\mathcal{T}}, \text{Concept}, \text{Expr})$$

where p is a predicate formed from assertions in $\Delta_{\mathcal{T}}$ with the constructors $p_1(\bar{X}_1) \wedge \dots \wedge p_n(\bar{X}_n)$. \bar{X}_i are tuples of variables or constants. The predicates p_1, \dots, p_n are whether concepts names, role names or an ordinary predicate. Revision operators TELL and FORGET have been defined in section 3.

Example 4

Let $\Delta_{\mathcal{T}}$ be Tbox :

$$\Delta_{\mathcal{T}} := \{ \text{Product}, \text{Described-product}, \text{European} \sqcap \text{American} \sqsubseteq \perp$$

$$\text{European-connect} := \exists \text{connect}. \text{European}$$

$$\text{American-connect} := \exists \text{connect}. \text{American} \}$$

The concepts *Product*, *Described-product*, *European*, *American* are primitive concepts. The concept *European-connect* (respectively, *American-connect*) is defined as the set of individuals having at least one filler of the role *connect* ; this filler is an individual of the concept *European* (respectively, *American*).

Let $\Delta_{\mathcal{A}}$ be ground facts:

$$\Delta_{\mathcal{A}} := \{ \text{Sold-by}(a,b), \text{Bought-by}(a,c), \text{European-connect}(b), \text{American-connect}(c) \}$$

The fact *Sold-by*(a,b) means that product a is sold by enterprise b . The fact *Bought-by*(a,c) means that product a is bought by enterprise c .

Let $\Delta_{\mathcal{R}}$ be only one revision production rule. This rule uses the facts : *Sold-by* and *Bought-by* :

$$\Delta_{\mathcal{R}} := \{$$

$$r : \text{Sold-by}(X, Y_1) \wedge \text{Bought-by}(X, Z_1) \wedge \text{connect}(Y_1, Y_2) \wedge \text{connect}(Z_1, Z_2) \wedge \text{European}(Y_2) \wedge \text{American}(Z_2) \Rightarrow \text{TELL}(X, \text{Described-product}) \}$$

The rule r is a revision production rule stating that if a product X is sold by an enterprise Y_1 which has a connection with an enterprise *European*, and the product X is bought by an enterprise Z_1 which has a connection with an enterprise *American*, then that product X must be a *Described-product*.

The fact *European-connect*(b) and *European-connect* := $\exists connect.European$ yield *connect*(b, b') and *European*(b'). Similarly, the fact *American-connect*(c) and *American-connect* := $\exists connect.American$ yield *connect*(c, c') and *American*(c'). Hence, the condition of the rule r is verified.

□

4.2 Semantics of combining formalism

We can now define the semantics of the formalism combining revision production rules and description logic $\mathcal{AL}\mathcal{E}$ based on the semantics of revision operators. We will define semantics of the formalism on the assumption that the propagation operators are not taken into account. It will not be too difficult to extend the following definition to a KB with these operators.

Let Δ be a KB composed of three components $\Delta = (\Delta_{\mathcal{T}}, \Delta_{\mathcal{A}}, \Delta_{\mathcal{PR}})$. An interpretation \mathcal{I} is a pair $(\mathcal{O}, \cdot^{\mathcal{I}})$ where \mathcal{O} is a non-empty set and a function $\cdot^{\mathcal{I}}$ maps every constant a in Δ to an object $a^{\mathcal{I}} \in \mathcal{O}$. An interpretation I is a model of Δ if it is a model of each component of Δ . Models of terminological component $\Delta_{\mathcal{T}}$ are normally defined as what of description logics. An interpretation I is a model of a ground fact $p(\bar{a})$ in $\Delta_{\mathcal{A}}$ if $\bar{a}^I \in p^I$.

An interpretation I is a model of $r : p \Rightarrow \text{TELL}(C, Expr)$ if, whenever the function $\cdot^{\mathcal{I}}$ maps the variables of r to the domain \mathcal{O} such that $(\bar{X}_i)^I \in p_i^I$ for every atom of the antecedent of r , then

1. if $a^I \in (C^I \cap Expr^I)$ then $a^I \in (\text{TELL}(C, Expr))^I$
2. if $a^I \in C^I$ and $a^I \notin Expr^I$ then $a^I \notin (\text{TELL}(C, Expr))^I$

An interpretation I is a model of $r : p \Rightarrow \text{FORGET}(C, Expr)$ if, whenever the function $\cdot^{\mathcal{I}}$ maps the variables of r to the domain \mathcal{O} such that $(\bar{X}_i)^I \in p_i^I$ for every atom of the antecedent of r , then

1. if $a^I \in C^I$ then $a^I \in (\text{FORGET}(C, Expr))^I$

5 Conclusion and Future Work

We introduced several revision policies and a framework for the revision operators on DL-based knowledge base. In the first policy, the assertion conservation on Abox, despite modifications of concept definition on Tbox, focuses our attention on inferences with the epistemic operator in a closed world. However, the interaction between the epistemic operator and the revision operators was not investigated in this paper. Next, we showed how to compute revision operators for language $\mathcal{AL}\mathcal{E}$. A deeper study of the complexity of these operators

is necessary for a possible improvement in the computing. An other question arises concerning the extension of the revision operators to languages allowing for disjunction (\mathcal{ALC}). This extension requires a structural characterization of subsumption for these languages.

On the other hand, if we allow for Horn rules as a third component, then which interactions will take place in the combining formalism ? In this case, the obtained formalism will be an extension of language CARIN [4]. Hence, an extension of the inference services of CARIN to the combining formalism deserves investigation.

References

1. B. Nebel. Reasoning and Revision in Hybrid Representation Systems, *Thesis 1990-1995*
2. B. Nebel. Terminological reasoning is inherently intractable. *Artificial Intelligence, 1990*
3. M. Schmidt-Schauß, G. Smolka. Attributive concept descriptions with complements. *Artificial Intelligence, 1991*
4. A. Y. Levy and M.C Rousset. Combining Horn Rules and Description Logics in CARIN, *Artificial Intelligence, Vol 104, 1998.*
5. F.M. Donini, M. Lenzerini, D. Nardi, W. Nutt, A. Schaerf. An Epistemic Operator for Description Logics, *Artificial Intelligence, Vol 100, 1998.*
6. F.M. Donini, M. Lenzerini, D. Nardi, A. Schaerf. Reasoning in Description Logics, *CSLI Publications, 1997*
7. R. Küsters. Non-Standard Inferences in Description Logics. *Thesis, Springer 2001*
8. F. Baader, R. Küsters and R. Molitor. Computing least common subsumer in Description Logics with existential restrictions, *Proc. of the 16th Int. Joint Conf. on Artificial Intelligence (IJCAI-99)*
9. C. Le Duc, N. Le Thanh. Problématique de l'ebXML et logiques de description. *Rapport de recherche à l'IS, 2002*

Appendix

Proposition 3.1

Let C be a propagated normal \mathcal{ALC} -concept description, D be a propagated normal \mathcal{ALC} -concept description and $C \sqsubseteq D$. If $C = X \sqcap D$ and $X \not\sqsubseteq D_i$ for all i where $D = D_1 \sqcap \dots \sqcap D_n$, then $\text{MINUS}(C, D) \equiv X$.

Proof:

Since $C = X \sqcap D$ are propagated normal concept descriptions, the description tree of C consists of two subtrees: one subtree for the description tree of X : \mathcal{G}_X , the other subtree for the description tree of D : \mathcal{G}_D . Operation $\text{MINUS}(C, D)$ performs the deletion of all images of the homomorphisms from \mathcal{G}_D to \mathcal{G}_C . This deletion makes the subtree \mathcal{G}_D in \mathcal{G}_C empty (top-concept) and it does not change the subtree \mathcal{G}_X in \mathcal{G}_C . Indeed, all of homomorphisms from \mathcal{G}_D to \mathcal{G}_C can be divided into three groups :

- The homomorphisms from \mathcal{G}_D to \mathcal{G}_D occur as part of \mathcal{G}_C . The deletion of the images of these homomorphisms makes the subtree \mathcal{G}_D of \mathcal{G}_C empty.
- The homomorphisms from \mathcal{G}_D to \mathcal{G}_X occur as part of \mathcal{G}_C . Since $X \not\sqsubseteq D$, the set of these homomorphisms is empty.
- The homomorphisms φ from \mathcal{G}_D to \mathcal{G}_C the image of which is composed *at least* of two subtrees starting at the root : one is found in the subtree \mathcal{G}_D and the other is found in the subtree \mathcal{G}_X . We have \mathcal{G}_D composed of subtrees \mathcal{G}_{D_i} and each subtree \mathcal{G}_{D_i} is composed of only one subtree starting at the root. Moreover, since the image $\varphi(\mathcal{G}_{D_i}) \not\sqsubseteq \mathcal{G}_X$ for all i , the set of these homomorphisms must be empty.

Thus, operator $\text{MINUS}(C, D)$ deletes only the images of homomorphisms of the first group i.e subtree \mathcal{G}_D in \mathcal{G}_C becomes top-concept and subtree \mathcal{G}_X in \mathcal{G}_C is not modified. As a consequence, $\text{MINUS}(C, D) \equiv X$.

□