ARTÍCULO

A Non-Monotonic Description Logics Model for Merging Terminologies* †

Martín O. Moguillansky & Marcelo A. Falappa

Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET)
Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA)
Departamento de Ciencias e Ingeniería de la Computación (DCIC)
Universidad Nacional del Sur (UNS)
Av. Alem 1253 - (B8000CPB) Bahía Blanca - Argentina

{mom,mfalappa}@cs.uns.edu.ar

Abstract

In order to deal with the *Ontology Change* problem and considering an environment where *Description Logics* (DLs) are used to describe ontologies, the question of how to integrate distributed ontologies appears to be in touch with *Belief Revision* since DL terminologies may define same concept descriptions of a not necessarily same world model. A possible alternative to reason about these concepts is to generate unique concept descriptions in a different terminology. This new terminology needs to be consistently created, trying to deal with the minimal change problem, and moreover, yielding a non-monotonic layer to express ontological knowledge in order to be further updated with new distributed ontologies.

Keywords: Belief Revision, Description Logics, Non Monotonic Reasoning, Ontology Change, Semantic Web.

1 Introduction

In order to reason about different ontologies, probably allocated in different places round the web, we will consider translated OWL ontologies into description logics (DLs). In DLs, the concept of Knowledge Base (**KB**) is composed of two main parts, TBoxes or Terminologies and ABoxes or Assertions. In this paper we focuss our investigation on how to reason about *terminologies*. Here many possibilities come through.

Just think about two distinct terminologies modeling each two different worlds, but containing a same subset of concepts (referred as *Ontology Integration* in [6]). Or just two distinct terminologies modeling the same world, where naturally a

*This article assumes some extra knowledge on description logics and reasoning services from the reader. For a more exhaustive reading he may refer to [15].

†This article is an extended version of the original article [18].

Partially financed by CONICET (PIP 5050), Universidad Nacional del Sur (PGI 24/ZN11) and Agencia Nacional de Promoción Científica y Tecnológica (PICT 2002 Nro 13096).

common subset of concepts will be described as part of both terminologies (referred as *Ontology Merging* in [6]). Furthermore, it might be probably impossible to get two concepts defined by different persons with exactly the same logic intention. Here is where the theory change arises as relevant protagonist in order to join consistently two terminologies redefining or reinforcing subconcepts.

The remainder of this paper is disposed as follows. The next section gives a brief description of the DL formalism, continued by section 3 with the analogous description of the theory change model, section 4 contributes to the formalization of merging DL terminologies, and describes an example operation of two different terminologies. Finally section 5 concludes and explains the related and future work in the area.

2 The DL Basic Formalism

A Knowledge Representation (KR) system based on Description Logics (DL) provides a formalization to specify the knowledge base (KB) contents, a way to reason about it, and a process to infer implicit knowledge. A KB is composed by two components. A TBox to manage the *terminology* of the application world and an ABox containing the *assertions* about named individuals in terms of the previous concepts.

A *terminology* is composed by atomic *concepts* which denote sets of individuals and atomic *roles* to manage relationships between individuals. Besides, complex concepts and roles are built from the atomics using given constructors. Reasoning tasks are dedicated to determine whether a description is satisfiable (*i.e.*, non-contradictory), or whether one description is more general than another one, that is, whether the first subsumes the second.

For an ABox, the problem is to verify the consistency of each set of assertions (*i.e.*, test if there is a model for the set) and find out whether a particular individual is an instance of a concept description in the TBox depending on the assertions in the ABox. The environment will interact with the KR by querying the KB and finally by adding and retracting concepts, roles and assertions.

2.1 Description Languages

Description Languages are defined by the constructors they provide. In this paper we will consider a subset of the large DL constructors set investigated so far. The basic Description Language introduced by [21] is the $\mathcal{AL}$ (Attribute Language). Let A be an atomic-concept, R an atomic-role, and C, D complex concepts, the grammar for the $\mathcal{AL}$ language is defined as follows,

$$C, D \rightarrow A|\top|\bot|\neg A|C \sqcap D|\forall R.C|\exists R.\top$$

To define the formal semantics of $\mathcal{AL}$-concepts we use *interpretations* ($\mathcal{I}$) that consist of a nonempty set $\Delta^{\mathcal{I}}$ (the domain of the interpretation) and an interpretation function $\cdot^{\mathcal{I}}$, that assigns to every atomic concept A a set $A^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}}$ and to every atomic role R a binary relation $R^{\mathcal{I}} \subseteq \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$. The interpretation function $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ is extended to concept descriptions as follows,

$$\top^{\mathcal{I}} = \Delta^{\mathcal{I}} \qquad \bot^{\mathcal{I}} = \emptyset$$
$$(\neg A)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash A^{\mathcal{I}} \quad (C \sqcap D)^{\mathcal{I}} = C^{\mathcal{I}} \cap D^{\mathcal{I}}$$
$$(\forall R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} |\ \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}$$
$$(\exists R.\top)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} |\ \exists b.(a,b) \in R^{\mathcal{I}}\}$$

In [2] is exhaustively detailed all language extensions depending on the constructors allowed in it, and namely $\mathcal{AL}[\mathcal{U}][\mathcal{E}][\mathcal{N}][\mathcal{C}][\mathcal{Q}]$[1]. In Table 1 a brief summary is given.

Example 1 : Considering a language constructor $\mathcal{N}$ and given a role completedCourse, a student of computer science is considered advanced if he has passed 15 courses out of a total of 25, $\geqslant$ 15 completedCourse $\sqcap \leqslant$ 25 completedCourse. For $\mathcal{Q}$, the number restrictions are concerned with roles limited to a certain concept. In this case, one can also say that a student should pass at least 6 logic courses and 9 computational courses to be considered advanced,

$$\geqslant 6\ \text{completedCourse.LogicCourse} \sqcap$$
$$\geqslant 9\ \text{completedCourse.ComputationalCourse} \sqcap$$
$$\leqslant 25\ \text{completedCourse}$$

□

[1] The use of $\mathcal{C}$ stands for complement. For $\mathcal{N}$ and $\mathcal{Q}$, n varies over the nonnegative integers, and $\|X\|$ stands for the cardinality of the set X.

Constructor	Written	Interpreted
Union ($\mathcal{U}$)	$C \sqcup D$	$(C \sqcup D)^{\mathcal{I}} = C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Negation ($\mathcal{C}$)	$\neg C$	$(\neg C)^{\mathcal{I}} = \Delta^{\mathcal{I}} \backslash C^{\mathcal{I}}$
Existential Quantification ($\mathcal{E}$)	$\exists R.C$	$(\exists R.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}$
Number Restrictions ($\mathcal{N}$)	$\geqslant nR$	$(\geqslant nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \|\{b \mid (a,b) \in R^{\mathcal{I}}\}\| \geq n\}$
	$\leqslant nR$	$(\leqslant nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \|\{b \mid (a,b) \in R^{\mathcal{I}}\}\| \leq n\}$
	$= nR$	$(= nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \|\{b \mid (a,b) \in R^{\mathcal{I}}\}\| = n\}$
Qualified Number Restrictions ($\mathcal{Q}$)	$\geqslant nR.C$	$(\geqslant nR.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \|\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\| \geq n\}$
	$\leqslant nR.C$	$(\leqslant nR.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \|\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\| \leq n\}$
	$= nR.C$	$(= nR.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid \|\{b \mid (a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\}\| = n\}$

Table 1: Constructors to extend the expressivity of $\mathcal{AL}$-languages.

2.2 Terminologies

Terminological axioms indicate how concepts or roles are related to each other following the *inclusion* form, $C \sqsubseteq D$ ($R \sqsubseteq S$), or the *equality* form, $C \equiv D$ ($R \equiv S$), where C and D are concepts (R and S are roles).

An interpretation $\mathcal{I}$ satisfies an inclusion $C \sqsubseteq D$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$, and it satisfies an equality $C \equiv D$ if $C^{\mathcal{I}} = D^{\mathcal{I}}$. Now given a set of axioms $\mathcal{T}$, an interpretation $\mathcal{I}$ satisfies $\mathcal{T}$ *iff* $\mathcal{I}$ satisfies each element of $\mathcal{T}$. If $\mathcal{I}$ satisfies an axiom in $\mathcal{T}$, then we say that it is a *model* of this axiom in $\mathcal{T}$. Then two axioms or two set of axioms are *equivalent* if they have the same models.

Definitions are used to describe complex concepts and made abstraction of them using a single name. An atomic concept on the left side of an equality *defines* the complex description explained on its right side.

A set of definitions $\mathcal{T}$ is called a *terminology* or a *TBox* if a symbolic name is defined only once. A terminology $\mathcal{T}$ contains a *cycle* *iff* there exists an atomic concept in $\mathcal{T}$ that uses itself [2]; otherwise $\mathcal{T}$ is called *acyclic*. An *acyclic* terminology $\mathcal{T}$ can be *expanded* iteratively through each definition in it, replacing each occurrence of a name on the right hand side with the concepts that it stands for. Now, we say that a terminology $\mathcal{T}$ is *definitorial* if it is *acyclic*, and we call to its semantics *descriptive semantics*. Those semantics that are motivated by the use of intuitively cyclic definitions are called *fixpoint semantics*. We will not consider fixpoint semantics in this paper.

2.3 Role Constructors

Binary relations between concepts are modeled by roles. If every role name is considered a role description or atomic role, and if R and S are roles descriptions, then $R \sqcap S$ (**intersection**), $R \sqcup S$ (**union**), $\neg R$ (**complement**), $R \circ S$ (**composition**), R^+ (**transitive closure**), and R^- (**inverse**) are also role descriptions. An interpretation $\mathcal{I}$ is adapted to the inverse role description as follows,

Inverse ($\mathcal{I}$ or $^{-1}$): $(R^-)^{\mathcal{I}} = \{(b,a) \in \Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}} \mid (a,b) \in R^{\mathcal{I}}\}$

Example 2 : For instance a hasParent role is obtained by applying the inverse role constructor to a given hasChild role. □

2.4 Properties for Reasoning

Given a terminology $\mathcal{T}$, if there is some interpretation of a concept that satisfies the axioms in $\mathcal{T}$ (a model of $\mathcal{T}$), then the concept denotes a nonempty set for the interpretation, furthermore this concept is known to be *satisfiable* w.r.t. $\mathcal{T}$. Otherwise it is called *unsatisfiable*. Formally,

Satisfiability [2]: A concept C is *satisfiable* w.r.t. $\mathcal{T}$ if there exists a model $\mathcal{I}$ of $\mathcal{T}$ such that $C^{\mathcal{I}}$ is nonempty. In such a case we say that $\mathcal{I}$ is a *model* of C.

Checking (un)satisfiability of concepts might be considered a key inference given that a number of other important inferences for concepts can be reduced to it. For instance, in order to check whether a domain model is correct, or to optimize concepts, we may want to know whether one concept is more general than another. This is called the *subsumption* problem. A concept C is *subsumed* by a concept D if in every model of $\mathcal{T}$, C is a subset of D.

Subsumption [2]: A concept C is *subsumed* by a concept D w.r.t. $\mathcal{T}$ if $C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. In such a case we write $C \sqsubseteq_{\mathcal{T}} D$ or $\mathcal{T} \models C \sqsubseteq D$.

A new kind of reasoning algorithms in DLs raised from the approach of considering satisfiability checking as the main inference. These algorithms are known as Tableaux Algorithms and can be understood as a specialized tableaux calculi.

Some special comments are dedicated to these kind of reasoners to be investigated as part of our future work on this research. Some of the latest DL systems based on satisfiability checking are KRIS [3], CRACK [4], FaCT [13], DLP[19], and RACE [10].

3 The Belief Dynamic Model

A **belief base** is a knowledge state represented through a set of sentences not necessarily closed under logical consequence. We also know that a **belief set** is a set of sentences of a determined language, closed under logical consequence. In general, a belief set is infinite being this the main reason of the impossibility to deal with this kind of sets in a computer. Instead, it is possible to characterize the properties that must satisfy each of the change operations on finite representations of a knowledge state.

The classic operations in the theory change are expansions, contractions, and revisions. An **Expansion** operation noted with "+", adds a new belief to the epistemic state, without guaranteeing its consistency after the operation. A **Contraction** operation, noted with "−", eliminates a belief α from the epistemic state and those beliefs that make possible its deduction or inference. The sentences to eliminate might represent the *minimal change* on the epistemic state. Finally, a **Revision** operation ("$*$") inserts sentences to the epistemic state, guaranteeing consistency (if it was consistent before the operation)[1] [**?**]. This means that a revision adds a new belief and perhaps it eliminates others to avoid inconsistencies. Other non-classical operations exists, like **Merge** operation [8][5] noted with " $\circ$ ", that fusions belief bases or sets assuring a consistent resultant epistemic state, and a **Consolidation** operation (" ! ") that restores consistency to the epistemic state [12].

3.1 Kernel Contractions

The **Kernel Contraction** operator is applicable to belief bases and belief sets. It consist of a contraction operator capable of the selection and elimination of those beliefs in K that contribute to infer α.

Definition 3.1.1 - [11]: Let K be a set of sentences and α a sentence. The set $K^{\perp\!\!\!\perp}\alpha$, called *set of kernels* is the set of sets K' such that (1) $K' \subseteq K$, (2) $K' \vdash \alpha$, and (3) if $K'' \subset K'$ then $K'' \nvdash \alpha$. The set $K^{\perp\!\!\!\perp}\alpha$ is also called *set of α-kernels* and each one of its elements are called *α-kernel*.

For the success of a contraction operation, we need to eliminate, at least, an element of each α-kernel. The elements to be eliminated are selected by an **Incision Function**.

Definition 3.1.2 - [11]: Let K be a set of sentences and "σ" be an *incision function* for it such that for any sentence α it verifies, (1) $\sigma(K^{\perp\!\!\!\perp}\alpha) \subseteq \bigcup(K^{\perp\!\!\!\perp}\alpha)$ and (2) If $K' \in K^{\perp\!\!\!\perp}\alpha$ and $K' \neq \emptyset$ then $K' \cap \sigma(K^{\perp\!\!\!\perp}\alpha) \neq \emptyset$.

Once the incision function was applied, we must eliminate from K those sentences that the incision function selects, *i.e.*, the new belief base would consist of all those sentences that were not selected by σ.

Definition 3.1.3 - [11]: Let K be a set of sentences, α a sentence, and $K^{\perp\!\!\!\perp}\alpha$ the set of α-kernels of K. Let "σ" be an incision function for K. The operator " $-_\sigma$ ", called *kernel contraction determined by* "σ", is defined as, $K -_\sigma \alpha = K \backslash \sigma(K^{\perp\!\!\!\perp}\alpha)$.

Finally, an operator " $-$ " is a kernel contraction operator for K if and only if there exists an incision function "σ" such that $K - \alpha = K -_\sigma \alpha$ for all sentence α.

3.2 Merging Belief Bases

The union of two different belief bases may be inconsistent. Restoring this property to the resultant union may be thought in terms of a *deductively maximally consistent (d.m.c.)* subset of the union.

Definition 3.2.1 - Partial Meet Merge [8]: A *prima facie* candidate for the merge of two sets is any d.m.c. of their union. For each set K, a set X is a d.m.c. subset of K, if (1) $X \subseteq K$, (2) $X \nvdash \bot$, and (3) $\forall Y : X \subset Y \subseteq K$ implies $Y \vdash \bot$. It is easy to see that a set X is a d.m.c. subset of K just in case $X \in K \bot \{\bot\}$[2].

Fuhrmann defined in [8] a partial meet merge operation as a union of two bases, not necessarily closed under logic consequence, and a later consistency restoring applying a bottom contraction. Inspired on it we propose a merge operation over two bases, defined by means of the *Kernel Contraction* operator, and determined by an *Incision Function*, as follows.

Definition 3.2.2 - Merge: Let " $-$ " be a kernel contraction for the union of two belief bases $K_1 \cup K_2$, determined by an incision function "σ". Then the *Merge for Belief Bases* operator $\circ$ is defined as, $K_1 \circ K_2 = (K_1 \cup K_2) -_\sigma \bot$.

4 Terminology Integration

In order to reason about two presumably distributed and potentially inconsistent terminologies, a non monotonic operation for integration of both terminologies in a new consistent one is required. For achieving a formal definition of such an operator the following features might be desirable.

- A function that maps a concept name defined in one terminology to another concept name in the second terminology in order to recognize different concept names referring to a same concept of the real world.
- A theory change framework would be an interesting environment for revising beliefs and merging knowledge bases defined as part of the given terminologies.
- Conveniently, a concept defined in a given terminology will be **expanded** in order to generate its correspondent belief base in the theory change. This means that each concept description in it is expressed as a conjunction of basic concept descriptions in the same terminology.
- A translation method for expressing DL concepts as part of a first order logic language might be needed in order to apply the theory change operations.
- Contractions might retract minimal information. This means that an operation for merging terminologies should keep as much as possible knowledge in a consistency restoring process. This property reflects the minimal change requirement of the theory change.

4.1 Basic Definitions

Formal definitions might be interesting to specify reflecting some of the previous features to be the basis of a terminology integration operator.

Definition 4.1.1 - Concept Id.: Let $\mathcal{T}$ be a terminology composed of n distinct definitions, the i^{th} concept description D defined in the terminology $\mathcal{T}$ will be identified as $D_i^{\mathcal{T}}$, where $1 \leq i \leq n$.

Definition 4.1.2 - Names Mapping: Let ξ be a mapping function from a concept name defined in a terminology $\mathcal{T}_1$ to a different concept name defined in a terminology $\mathcal{T}_2$. Then $D_i^{\mathcal{T}_1} = \xi(D_j^{\mathcal{T}_2})$ indicates that $D_i^{\mathcal{T}_1}$ and $D_j^{\mathcal{T}_2}$ are name identifiers of a unique concept of the real world.

Definition 4.1.3 - Consistent Terminology Integration: Let $\triangleleft\triangleright$ be the operator for a *consistent terminology integration* $\mathcal{T}_1 \triangleleft\triangleright \mathcal{T}_2$. Such an operation will be composed of two consecutive sub-operations, *terminology unification* and *terminology consolidation*.

Intuitively, a **terminology unification** consist of copying every concept description from each terminology to a new one except for the application of a *non monotonic concept conjunction*. In such a case, those concepts defined in both terminologies that refers to a same concept in the real framework, identified by a *names mapping function* ξ, are consistently unified in a unique

[2]Fuhrmann's definition of the operator $\bot$ refers to a partial meet contraction defined by the use of a selection function.

new concept. The following sub-operation is defined as **terminology consolidation**. Here the idea is to restore consistency to the unified terminology $\mathfrak{T}$, capturing and solving inconsistencies that yields the unification process in concept descriptions that refers to a yet unified concept in its right hand side. A tentative unification operation was provided in [16] although this version did not consider a more complex case of study that we capture in this paper with the consolidation operation.

4.2 From Description Logics to First Order Logic Languages

DLs are not rule based languages but they may be translated into fragments of first order logics (FOL) in order to make an easier mapping to rule languages. By this, efficient logic programming based reasoners and deductive systems may be defined to make inference about the knowledge representation originally specified by a description language. Table 2, originally specified in [9], summarizes the translation rules above introduced.

DL	FOL
$C \equiv D$	$\forall x.C(x) \leftrightarrow D(x)$
$C \sqsubseteq D$	$\forall x.C(x) \rightarrow D(x)$
C	$C(x)$
$C \sqcap D$	$C(x) \wedge D(x)$
$\neg C$	$\neg C(x)$
$\exists R.C$	$\exists x.(R(y,x) \wedge C(x))$
$\forall R.C$	$\forall x.(R(y,x) \rightarrow C(x))$
$\geqslant nR$	$\exists y_1,...,y_n.(R(x,y_1) \wedge ... \wedge R(x,y_n)) \wedge (\bigwedge_{1 \leqslant i < n, i < j \leqslant n} y_i \neq y_j)$
$\leqslant nR$	$\forall y_1,...,y_{n+1}.(R(x,y_1) \wedge ... \wedge R(x,y_{n+1})) \rightarrow (\bigvee_{1 \leqslant i < (n+1), i < j \leqslant (n+1)} y_i = y_j)$

Table 2: DL - FOL equivalence.

The following definitions describe how our proposal manages DLs concepts descriptions as fragments in the theory change[3].

Definition 4.2.1 - KB of a Concept Description: Let $C \equiv C_1 \sqcap ... \sqcap C_n$ be an *expanded* concept description, then the set $K(C)$ will be the knowledge base for the concept description C such that $\phi_{C_1}, ..., \phi_{C_n} \in K(C)$, where ϕ_{C_i} is the first order logic translation of the concept C_i.

Observation 4.2.2: A DL conjunction $C_1 \sqcap C_2$ is interpreted in the theory change as $K(C_1) \cup K(C_2)$.
Proof: Let $C \equiv C_1 \sqcap C_2$ be a concept description, and let $C_1 \equiv A_1 \sqcap ... \sqcap A_n$, and $C_2 \equiv B_1 \sqcap ... \sqcap B_n$ be their respective atomic concept descriptions. Then the expanded concept description for C is $C \equiv A_1 \sqcap ... \sqcap A_n \sqcap B_1 \sqcap ... \sqcap B_n$. Finally using definition 4.2.1 follows that $K(C) = K(C_1) \cup K(C_2)$.□

4.3 Non Monotonic Concept Conjunction (⍃)

Let " ⍃ " be a *Non Monotonic Concept Conjunction* DL operator used as part of the terminology unification process and furthermore in the specification of the terminology consolidation, such that $D = D_1 \text{ ⍃ } D_2$ defines a new *satisfiable* concept description D from consistently unifying concepts D_1 and D_2 by the application of the operator ⍃ .

A consistent conjunction of two such a concept descriptions may intuitively be thought as a merge of the belief bases representing each concept. Afterwards, the resultant belief base should be translated back to the original description language in order to express the result as a new concept description. A translation method from DLs to belief bases in the theory change is necessary, then following the translation rules from a description language to a subset of first order logic rules in table 2, we may obtain the conversions of concept descriptions to belief bases in the theory change.

Definition 4.3.1 - Non Monotonic Concept Conjunction : Let D_1 and D_2 be two concept descriptions and let $K(D_1)$ and $K(D_2)$ be their correspondent belief bases (not necessarily closed under logical consequence). A *non monotonic concept conjunction* operation $D_1 \text{ ⍃ } D_2$, is interpreted as a merge operation for belief bases, such that $D_1 \text{ ⍃ } D_2 \Leftrightarrow K(D_1) \circ K(D_2)$.

Observation 4.3.2: If the conjunction $D_1 \sqcap D_2$ is satisfiable[4], then the non monotonic conjunction $D_1 \text{ ⍃ } D_2$ is equivalent to $D_1 \sqcap D_2$.
Proof: Suppose we have two concept descriptions D_1 and D_2, then by definition 4.2.1 their respective knowledge bases are $K(D_1)$ and $K(D_2)$.

[3] We adopt a $\mathcal{ALCEN}$ description language, where the classical $\mathcal{AL}$ attribute language is extended by *complement*, *existential quantification*, and *number restriction* concept constructors.

[4] A conjunction $D_1 \sqcap D_2$ is satisfiable if there exists a model $\mathfrak{I}$ of $\mathfrak{T}$ such that $(D_1 \sqcap D_2)^{\mathfrak{I}}$ is non empty.

($\Rightarrow$) Consider the non monotonic concept conjunction $D_1 \boxdot D_2$, by definition 4.3.1 it may be though as a merge operation of their knowledge bases such that $K(D_1) \circ K(D_2)$. Using definition 3.2.2 (*merge*) we have $(K(D_1) \cup K(D_2)) -_\sigma \bot$. Then by definition 3.1.3 (*kernel contraction*) it is equivalent to $(K(D_1) \cup K(D_2)) \backslash \sigma((K(D_1) \cup K(D_2))^{\perp\!\!\!\perp} \bot)$.
Now suppose that (*by hypothesis*) $D_1 \sqcap D_2$ is satisfiable, then by definition 3.1.1 (*set of kernels*) it follows that there is no $K' \subseteq (K(D_1) \cup K(D_2))$ such that $K' \vdash \bot$. Thus, we have no beliefs to select from $(K(D_1) \cup K(D_2))$, and finally by observation 4.2.2 we have $D_1 \sqcap D_2$ (as we wanted to prove).
The opposite ($\Leftarrow$) direction can be similarly proved. □

The following algorithm describes the operation $D \equiv D_1 \boxdot D_2$ defined in definition 4.3.1 and optimizes it considering the observation 4.3.2.

Algorithm 1 Non Monotonic Concept Conjunction $D \equiv D_1 \boxdot D_2$

Input: Two expanded concepts D_1, D_2.
Output: D.
  if $D_1 \sqcap D_2$ is un-satisfiable **then**
    $D \Leftarrow K(D_1) \circ K(D_2)$.
  else
    $D \Leftarrow D_1 \sqcap D_2$
  end if

4.4 Merging Terminologies ($\triangleleft\!\triangleright$)

As previously specified, an operation $\mathcal{T} = \mathcal{T}_1 \triangleleft\!\triangleright \mathcal{T}_2$, consist of two sub-operations; *unification* of both terminologies $\mathcal{T}_1$ and $\mathcal{T}_2$ in a new one $\mathcal{T}$, and the respective *consolidation* of $\mathcal{T}$ for consistency restoring. In what follows we provide the correspondent algorithms for both sub-operations.

Algorithm 2 Terminology Unification

Input: Two terminologies $\mathcal{T}_1$ and $\mathcal{T}_2$.
Output: A unified terminology $\mathcal{T}$.
  for all D in $\mathcal{T}_1$ or $\mathcal{T}_2$ **do**
    if exists a mapping $\xi(D)$ for concept description D **then**
      $\mathcal{D}^{\mathcal{T}} \Leftarrow D \boxdot \xi(D)$ is a new concept in $\mathcal{T}$
    else
      $\mathcal{D}^{\mathcal{T}} \Leftarrow D$ is a new concept in $\mathcal{T}$
    end if
  end for

Observation 4.4.1: If no loss of knowledge is desired in a concept unification process – *i.e.*, that the conjunction of concept descriptions is unsatisfiable and the non monotonic conjunction will eliminate some beliefs from the epistemic state– then we should not consider these concepts as specifiers of a same concept of the real framework. So we should remove the mapping value in ξ that relate them.

Observation 4.4.2: Note that the consolidation algorithm adds new concept descriptions noted as $\mathcal{C}$ and $\mathcal{D}$ and these new definitions are considered in later iterations.

4.5 Worked Example

In this section we show how two different terminologies might be consistently merged in a new one following the previous definitions.

Algorithm 3 Terminology Consolidation

Input: The unified terminology $\mathcal{T}$.
Output: The consolidated terminology $\mathcal{T}$.
  for all $\mathcal{D}^{\mathcal{T}}$ **do**
    for all $C^{\mathcal{T}} \equiv D \sqcap C_{tail}$, where C_{tail} may be thought as $C_1 \sqcap C_2 \sqcap ... \sqcap C_n$ **do**
      if $\mathcal{D}^{\mathcal{T}} \sqcap C_{tail}$ is satisfiable **then**
        Replace the original definition $C^{\mathcal{T}} \equiv D \sqcap C_{tail}$ by $\mathcal{C}^{\mathcal{T}} \equiv \mathcal{D}^{\mathcal{T}} \sqcap C_{tail}$
      else
        Let S be $\sigma((K(\mathcal{D}^{\mathcal{T}}) \cup K(C_{tail}))^{\perp\!\!\!\perp} \bot)$ where the selection's scope is restricted to $K(\mathcal{D}^{\mathcal{T}})$.
        Generate $D_1^{\mathcal{T}} \Leftarrow K(\mathcal{D}^{\mathcal{T}}) \backslash S$ in $\mathcal{T}$.
        Redefine $\mathcal{D}^{\mathcal{T}} \equiv D_1^{\mathcal{T}} \sqcap S_{DL}$ in $\mathcal{T}$, where S_{DL} is the description language translation of S.
        Replace the original definition $C^{\mathcal{T}} \equiv D \sqcap C_{tail}$ by $\mathcal{C}^{\mathcal{T}} \equiv D_1^{\mathcal{T}} \sqcap C_{tail}$.
      end if
    end for
  end for

$Bird$	$\equiv$	$Animal \sqcap Bipedal \sqcap Oviparous \sqcap$ $hasFeathers \sqcap = 2hasWings$
$Mammal$	$\equiv$	$Animal \sqcap \forall giveBirth.LiveBirth$
$Oviparous$	$\equiv$	$Animal \sqcap \forall giveBirth.Egg$
$Bipedal$	$\equiv$	$= 2hasFoot$
$LiveBirth$	$\equiv$	$hasHeartBeat \sqcap$ $hasVoluntaryMovement$
Egg	$\equiv$	$hasHeartBeat \sqcap$ $\neg hasVoluntaryMovement$

Table 3: A terminology $\mathcal{T}_1$ (TBox) with concepts about animals.

$Platypus$	$\equiv$	$Aquatic \sqcap Monotreme$
$Monotreme$	$\equiv$	$Mammal \sqcap Oviparous$
$Mammal$	$\equiv$	$Animal \sqcap$ $\geqslant 2hasMammaryGlands$
$Oviparous$	$\equiv$	$Animal \sqcap \geqslant 1layEggs$

Table 4: A terminology $\mathcal{T}_2$ (TBox) with concepts about animals.

The terminology expressed in table 4 shows among other definitions, some main characteristics of *mammals* and *oviparous* animals, and particularly defines *monotremes* to be a conjunction of both animal classes. When we try to merge terminologies $\mathcal{T}_1$ and $\mathcal{T}_2$ we find that the concept descriptions for *mammals* and *oviparous* in $\mathcal{T}_1$ yield the following contradiction,

$$hasVoluntaryMovement \sqcap \neg hasVoluntaryMovement$$

4.5.1 Method Application

In what follows we will develop the integration of both terminologies following the previous $\triangleleft\triangleright$ operation definition in order to see more clearly how the consistency problem is solved. Let consider the following mapping instances,

- $Mammal^{\mathcal{T}_1} = \xi(Mammal^{\mathcal{T}_2})$
- $Oviparous^{\mathcal{T}_1} = \xi(Oviparous^{\mathcal{T}_2})$.

Then, the unified terminology $\mathcal{T} = \mathcal{T}_1 \triangleleft\triangleright \mathcal{T}_2$ will have the following concept descriptions,

$Bird$	$\equiv$	$Bird^{\mathcal{T}_1}$
$Platypus$	$\equiv$	$Platypus^{\mathcal{T}_2}$
$Monotreme$	$\equiv$	$Monotreme^{\mathcal{T}_2}$
$Mammal$	$\equiv$	$Mammal^{\mathcal{T}_1} \boxdot Mammal^{\mathcal{T}_2}$
$Oviparous$	$\equiv$	$Oviparous^{\mathcal{T}_1} \boxdot Oviparous^{\mathcal{T}_2}$
$Bipedal$	$\equiv$	$Bipedal^{\mathcal{T}_1}$
$LiveBirth$	$\equiv$	$LiveBirth^{\mathcal{T}_1}$
Egg	$\equiv$	$Egg^{\mathcal{T}_1}$

Table 5: The unified terminology $\mathcal{T}$ from the originals $\mathcal{T}_1$ and $\mathcal{T}_2$.

Merging belief bases $K(Mammal^{\mathcal{T}_1})$ and $K(Mammal^{\mathcal{T}_2})$ does not arise any inconsistency.

$K(Mammal^{\mathcal{T}})$

$Animal(X)$
$hasMammaryGlands(X, y_1)$
$hasMammaryGlands(X, y_2)$
$giveBirth(Y, X) \rightarrow hasHeartBeat(X) \wedge hasVoluntaryMovement(X)$

The resultant unified concept will be translated from the previous belief base to the correspondent description language as,

$$Mammal^{\mathcal{T}} \equiv Animal \sqcap \geqslant 2hasMammaryGlands \sqcap \forall giveBirth.(hasHeartBeat \sqcap hasVoluntaryMovement).$$

A similar situation occurs with the belief base for concept description $Oviparous$, and its correspondent translation to DLs,

$$Oviparous^{\mathcal{T}} \equiv Animal \sqcap \geqslant 1layEggs \sqcap \forall giveBirth.(hasHeartBeat \sqcap \neg hasVoluntaryMovement).$$

$K(Oviparous^{\mathcal{T}})$

$Animal(X)$
$layEggs(X, y_1)$
$giveBirth(Y, X) \rightarrow hasHeartBeat(X) \wedge \neg hasVoluntaryMovement(X)$

We developed so far the terminology unification, first sub-operation of the terminology integration previously defined. The following step is the application of the terminology consolidation in order to verify and restore consistency to the resultant unified terminology. Here, the concept description for $Monotreme$ yields the inconsistency.

$$Monotreme^{\mathcal{T}} \equiv Animal \sqcap \geqslant 2 hasMammaryGlands \sqcap \forall giveBirth.(hasHeartBeat \sqcap hasVoluntaryMovement) \sqcap Animal \sqcap \geqslant 1 layEggs \sqcap \forall giveBirth.(hasHeartBeat \sqcap \neg hasVoluntaryMovement)$$

Clearly, this concept is unsatisfiable, so from the $\perp$-kernels set

$$(K(Mammal^{\mathcal{T}}) \cup K(Oviparous^{\mathcal{T}}))^{\perp\!\!\!\perp} \perp$$

an appropriate[5] incision function σ would select rules only from $K(Mammal^{\mathcal{T}})$ in order to avoid the inconsistency in this concept. In such a case the correspondent sentence would be

$$giveBirth(Y, X) \rightarrow hasHeartBeat(X) \wedge hasVoluntaryMovement(X)$$

that comes from the DL concept

$$\forall giveBirth. (hasHeartBeat \sqcap hasVoluntaryMovement)$$

so a new concept $Mammal_1^{\mathcal{T}}$ is defined to be part of the terminology $\mathcal{T}$ without considering the previous selection, such that

$$Mammal_1^{\mathcal{T}} \equiv Animal \sqcap \geqslant 2 hasMammaryGlands$$

Then the definitive concept for the definition of $Mammal$ in $\mathcal{T}$ would be,

$$Mammal^{\mathcal{T}} \equiv Mammal_1^{\mathcal{T}} \sqcap \forall giveBirth.(hasHeartBeat \sqcap hasVoluntaryMovement)$$

Finally, following the consolidation algorithm, the concept description for $Monotreme$ in terminology $\mathcal{T}$ would be,

$$Monotreme^{\mathcal{T}} \equiv Mammal_1^{\mathcal{T}} \sqcap Oviparous^{\mathcal{T}}$$

Note that following the proposed method for terminology integration we do not only eliminate the inconsistency when merging both terminologies, but also keep all information as part of the resultant terminology, by identifying and splitting the problematic concept in two interrelated, revisiting the hierarchy technique of the object oriented paradigm.

5 Conclusions, Related and Future Work

A union operation of terminologies probably yields contradictions on concept descriptions and further inconsistency in the resultant terminology. The use of a belief revision framework to define terminologies in order to meet a consistent merge operation is proposed and generates a new *Non-monotonic Description Logics* model as a powerful theory to be applied on future Semantic Web researches.

Ontology Change [6] expresses the necessity of modifying the knowledge described in ontologies responding to different given interests. For instance, ontology changes may arise due to some change originated in the world being modeled, or on users' needs; and/or due to previously unknown knowledge, or bugs found after design steps.

Many authors are nowadays focusing their efforts in some closely related areas under terms like, ontology translation, evolution, integration and merging. For instance, in [6] an AGM compatible contraction operator for DLs is described, and based on it a DL revision operator is defined in [20], from the generalized AGM postulates. These two works are some of the newest results obtained so far by considering theory change and DLs as an hybrid framework for belief revision.

Our proposed approach means an interesting alternative in the area, whose most notorious difference relays in the fact of the focalization on *non-prioritized change operations* where the AGM postulates are not fully met. From a naming convention point of view, although most of our work might be thought as *Ontology Merging*, we enclose it in the so called *Ontology Integration*, following the terminology specified in [6], due to some slight overlapping that we consider exists between these terms.

[5] The reader is invited to refer the conclusions of this work to understand this selection.

$Bird$	$\equiv$	$Animal \sqcap Bipedal \sqcap Oviparous \sqcap hasFeathers \sqcap = 2 hasWings$
$Platypus$	$\equiv$	$Aquatic \sqcap Monotreme$
$Monotreme$	$\equiv$	$Mammal_1 \sqcap Oviparous$
$Mammal_1$	$\equiv$	$Animal \sqcap \geqslant 2 hasMammaryGlands$
$Mammal$	$\equiv$	$Mammal_1 \sqcap \forall giveBirth.LiveBirth$
$Oviparous$	$\equiv$	$Animal \sqcap \forall giveBirth.Egg \sqcap \geqslant 1 layEggs$
$Bipedal$	$\equiv$	$= 2 hasFoot$
$LiveBirth$	$\equiv$	$hasHeartBeat \sqcap hasVoluntaryMovement$
Egg	$\equiv$	$hasHeartBeat \sqcap \neg hasVoluntaryMovement$

Table 6: The resultant terminology $\mathcal{T} = \mathcal{T}_1 \triangleleft\triangleright \mathcal{T}_2$ with concepts about animals.

5.1 Incision Function and Confidence Levels

The *incision function* (σ) deserves a special discussion due to its responsibility on the information management of knowledge bases which makes to keep or restore consistency regarding the selections it does. For instance, several merge operators have been proposed in the theory change bibliography, but in this work we proposed a different one based on a kernel contraction determined by an incision function. This is due to our intentions to reduce every belief revision and merge operation to an "intelligent" definition of the incision function, so placing it on top of our future investigations.

Another situation is given by concept descriptions to be consistently merged "retracting" those simple concepts descriptions that presumably generate the inconsistency in the new concept description. This selection is the one being made by the incision function, but here always might be more than one possible election to avoid inconsistency. More formally, $C \Leftarrow K(C_1) \cup K(C_2) -_\sigma \bot$ might be solved selecting knowledges from any of the two involved concept descriptions. Here is where *Epistemic Entrenchment*[6] methods arise, in order to properly select, w.r.t. the local environment, the most convenient knowledge and to give only one possible concept description as the operation resultant.

Motivated by the latter examples, a deeper investigation on *Epistemic Entrenchment* methods could be useful to semi-automate the well functioning of a reasonable incision function (σ) to cut the α-kernels obtained by the use of merge operations. This means that the incision function (σ) will select those sub-concepts with less plausibility or epistemic entrenchment to be cut off the resulting definition.

To achieve this, we will investigate *confidence levels* on terminologies to be incorporated by the ontology integration requester depending on his confidence about the origin where the ontology came from (author). Furthermore, it would be interesting also to consider confidence levels on concept descriptions incorporated by the ontology designer. By this, we intend to have some extra information at the time of evaluating which is the most relevant knowledge to be held after an integration operation.

5.2 Pure DL's Reasoning Services and Tableaux Algorithms

Another goal in our research is to formalize the relying theory change definitions into a pure DL's flavor. Part of this work has being done in [6], where the generalized postulates have being defined and their pertinent analysis developed. But still we have some formalizations taken from the theory change that have not being generalized.

By this we want to avoid conversions from DL's into fragments of FOL, reasoning directly on description languages knowledge specification. In order to achieve this shortcut for reasoning services, we might firstly specify definitions like α-kernels thinking only on DLs, this means that we should take care about DL reasoners that might be used to give proofs of DL knowledge bases.

Relevant inference problems usually are reduced to the consistency problem for ABoxes, provided

[6]The *Epistemic Entrenchment* method specifies a way to measure the level of importance of a sentence α to belong to the epistemic state.

that the DL at hand allows for conjunction and negation. However, for those description languages of DL systems that do not allow for negation, subsumption of concepts can be computed by so-called *structural subsumption algorithms*, *i.e.*, algorithms that compare the syntactic structure of (possibly normalized) concept descriptions.

Although usually very efficient, they are only complete for rather simple languages with little expressivity. In particular, DLs with (full) negation and disjunction cannot be handled by structural subsumption algorithms. For such languages, so-called *tableau-based algorithms* have turned out to be very useful.

Tableaux algorithms are nowadays probably the most important reasoning algorithms used in the area. A distinctive feature of this reasoning service is the way it reasons about incomplete information, inferring new beliefs by combining assumed ones (from the generated models) and effective knowledge (from the base).

All this happens while proving clauses' satisfiability, and indeed, while contracting a sentence from the base. By this, we have a completely different way of reasoning about knowledge due to a multiple generation of base extensions. This all is what motivates the definition of a model contraction operation and its several components.

Let give an example borrowed from [7] in order to understand more precisely the extensions obtained during a reasoning process. Let Σ be a knowledge base such that,

$$\Sigma = \left\{ \begin{array}{l} FRIEND(john, susan) \\ FRIEND(john, andrea) \\ LOVES(susan, andrea) \\ LOVES(andrea, bill) \\ Female(susan) \\ \neg Female(bill) \end{array} \right\}$$

Now we want to know if is there some not $Female$ loving a $Female$ who is $FRIEND$ of $john$. This is a query $\Sigma \models^{?} \alpha$ such that α is $\exists FRIEND.(Female \sqcap (\exists LOVES.\neg Female))(john)$. Following the given tableau specifications in [2], note that we have two different possibilities in order to achieve satisfiability of α, this is, two interpretations (models) named $\mathcal{M}_1$ and $\mathcal{M}_2$ satisfying the query $\Sigma \models \alpha$, where $\neg Female(andrea) \subseteq \mathcal{M}_1$ and $Female(andrea) \subseteq \mathcal{M}_2$.

Let analyze the yielding situation with respect to the α-kernels for each model.

1. Considering $\mathcal{M}_1$ there is only one proof set K' for α, this is $\Sigma^{\perp\!\!\!\perp}\alpha = \{FRIEND(john, susan), Female(susan), LOVES(susan, andrea), \neg Female(andrea)\}$
2. Considering $\mathcal{M}_2$ there is also only one proof set K' for α, $\Sigma^{\perp\!\!\!\perp}\alpha = \{FRIEND(john, susan), Female(andrea), LOVES(andrea, bill), \neg Female(bill)\}$

Now if we verify the restrictions given in definition 3.1.1 for α-kernels, we realize that the first restriction $K' \subseteq \Sigma$ is not verified due to the assumption taken during the reasoning service, *i.e.*, the beliefs adopted from each model $\mathcal{M}_1$ and $\mathcal{M}_2$ that are not part of the explicit knowledge base Σ. In order to facilitate further reference to this beliefs, let call $K'_{\mathcal{M}_1} = \{\neg Female(andrea)\}$ and $K'_{\mathcal{M}_2} = \{Female(andrea)\}$.

Note that we now have redefined the K' proof set as $K' = K'_{\Sigma} \cup K'_{\mathcal{M}}$, where K'_{Σ} is the proof subset that is part of the explicit KB Σ, and $K'_{\mathcal{M}}$ that extends it consistently, is the set of beliefs assumed in $\mathcal{M}$, *i.e.*, that are outside the KB Σ.

A formal definition of this new proposal and some other extensions of the theory change seen in this paper is explained in [17]. A complete definition of the new proposed theory for model based contractions on description languages may be referred in [14].

Acknowledgements

"Inteligencia Artificial" is a periodic publication distributed by the *"Asociación Española para la Inteligencia Artificial (AEPIA)"*.

References

[1] Carlos Alchourrón, Peter Gärdenfors, and David Makinson. *On the Logic of Theory Change: Partial Meet Contraction and Revision Functions. The Journal of Symbolic Logic*, 50:510–530, 1985.

[2] F. Baader and W. Nutt. Basic Description Logics. *In the Description Logic Handbook, Cambridge University Press*, pages 47–100, 2002.

[3] Franz Baader and Bernhard Hollunder. A Terminological Knowledge Representation System with Complete Inference Algorithms. In *Proceedings of the First International Workshop on Processing Declarative Knowledge*, volume 572, pages 67–85, Kaiserslautern (Germany), 1991. Springer–Verlag.

[4] P. Bresciani, E. Franconi, and S. Tessaris. Implementing and Testing Expressive Description Logics: Preliminary Report. *In Proc. of the 1995 Description Logic Workshop (DL'95)*, pages 131–139, 1995.

[5] M. Falappa, G. KernIsberner, and G. Simari. Explanations, Belief Revision and Defeasible Reasoning. *Artificial Intelligence Journal*, 141(1-2):1–28, 2002.

[6] Giorgos Flouris. On Belief Change and Ontology Evolution. *Doctoral Dissertation, Department of Computer Science, University of Crete*, February 2006.

[7] Enrico Franconi. Propositional Description Logics. *From his course Description Logics dictated during argentinean springtime at Universidad Nacional del Sur, Bahía Blanca*, 2006.

[8] André Fuhrmann. An Essay on Contraction. *The European Association for Logic, Language and Information (FOLLI), ISBN (Paperback): 1575860546, ISBN (Cloth): 1575860554*, 1996.

[9] Benjamin Grosof, Ian Horrocks, Raphael Volz, and Stefan Decker. Description Logic Programs: Combining Logic Programs with Description Logic. *In Proceedings 12th International Conference on the World Wide Web (WWW-2003), Budapest, Hungary*, May 20-23, 2003.

[10] Volker Haarslev and Ralf Moller. RACER System Description. *In Proc. of the Int. Joint Conf. on Automated Reasoning (IJCAR 2001)*, 2083:701–705, 2001.

[11] S. O. Hansson. Kernel Contraction. *The Journal of Symbolic Logic*, 59:845–859, 1994.

[12] S. O. Hansson. Semi-Revision. *Journal of Applied Non-Classical Logic*, 7:151–175, 1997.

[13] Ian Horrocks. Using an Expressive Description Logic: FaCT or Fiction? *In Proc. of the 6th Int. Conf. on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 636–647, 1998.

[14] M. Moguillansky and M. Falappa. Model Based Contractions on DLs. Computing science technical report, Universidad Nacional del Sur (UNS), Departamento de Cs. e Ing. de Computación (DCIC), Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA), Bahía Blanca, Argentina, 2007. To be submitted.

[15] M. Moguillansky and M. Falappa. Tableau Calculi for Revision of Description Logics Knowledge Bases. Computing science technical report, Universidad Nacional del Sur (UNS), Departamento de Cs. e Ing. de Computación (DCIC), Laboratorio de Investigación y Desarrollo en Inteligencia Artificial (LIDIA), Bahía Blanca, Argentina, 2007. To be posted.

[16] M. Moguillansky and M. Falappa. On the Use of Belief Revision to Merge Description Logic Terminologies. *WICC'2006, Universidad de Moron*, pages 57–61, ISBN 978–950–9474–34–5, June 2006.

[17] M. Moguillansky and M. Falappa. Tableau Calculi for Description Logics Revision. *WICC 2007 IX Workshop de Investigadores en Ciencias de la Computación, Universidad Nacional de la Patagonia San Juan Bosco, Trelew*, pages 160–164, ISBN 978–950–763–075–0, May 2007.

[18] M. Moguillansky and M. Falappa. Towards A Non Monotonic Description Logics Model. *XII Congreso Argentino de Ciencias de la Computación, CACIC'2006, Universidad Nacional de San Luis*, pages 1354–1365, ISBN 950–609–050–5, October 2006.

[19] Peter F. Patel-Schneider. Dlp. *In Proc. of the 1999 Description Logic Workshop (DL'99)*, pages 9–13, 1999.

[20] Guilin Qi, Weiru Liu, and David Bell. Knowledge Base Revision in Description Logics. *In Proceedings of 10th European Conference on Logics in Artificial Intelligence (JELIA'06), Springer Verlag*, pages 386–398, September 2006.

[21] Manfred Schmidt-Schauß and Gert Smolka. Attributive Concept Descriptions with Complements. *Artificial Intelligence*, 48(1):1–26, 1991.