Toward Updates in Description Logics

Mathieu Roger, Ana Simonet, Michel Simonet

Laboratory TIMC-IMAG- Osiris Team
La Tronche Cedex
Mathieu.Roger@imag.fr, Ana.Simonet@imag.fr, Michel.Simonet@imag.fr

Abstract

The use of DL systems to add reasoning capabilities to database is now a major trend in convergence between knowledge base and database system but is still confronted with the issue of data update. DL systems provide fact additions and retractions but no real object update mechanisms. We present a semantics for update that favours attribute values to concept membership and deals with incomplete information. After relating our work to an existing previous one on update semantics we address implementation issues.

1 Introduction

Knowledge base systems are designed to express complex queries upon highly organised data whereas databases are meant to store and retrieve huge amounts of data in an efficient and secure way. Merging both approaches aims at combining their respective advantages and has been a hot topic for many years. Existing works differ both on database models (relational or object-oriented) and on knowledge base models (rule-based, constraint-based…). The current trend seems to be towards object orientation and the systematic use of Description Logics. A lot of systems implementing DL reasoners exist and, as time passes, they become more and more expressive and efficient, making them good candidates for combination with databases. Whereas translating schemes from database to DL has already been investigated [Borgida and Brachman 93; Bresciani 95], few work deal with updates. Practical DL systems manage a single ABox to which one can add or retract assertions. These capacities are very general but do not directly lead to database update. In this paper we consider the issue of introducing update semantics into a DL system, which is strongly related to view management in databases.
In classical database management, updates are limited to the assignment of values to attributes. Assigning "properties" to objects or to attributes is not common to databases whereas it is one of the main purposes of knowledge bases. Assigning an interval value to an attribute is not considered in databases, even less assigning an object to a view. In Description Logics, it is quite common to express properties of the form *Object* $\in$ *Concept*, which we call "concept assignment" in this paper. DL systems are able to check the consistency of the new assertion with respect to the previously asserted properties. However, these assertions are meant to complete an incompletely known object, not to modify it,

because assertions provided at a given moment cannot be contradicted by a further one. This is different in databases where updates are often meant to contradict existing assertions on data.

Problems may arise when assignment to a concept is performed on an existing object. If the new assertion only complements or refines the object, no question arises. Questions arise when the new concept contradicts the previous assertions made on the object. Should we forget all the previous assertions that contradict the new one? Should we try to find a subset of properties compatible with both the old and the new assertions? This kind of questions has been considered in knowledge base revision and there is no simple answer that could be applied to concept assignment in databases.

Assigning a concept C to an object o can be interpreted in two ways: 1) if the object o is compatible with C, then C’s properties are asserted on o; 2) modify (update) o in such a way that it becomes compatible with C. We have chosen the first interpretation, and we distinguish between attribute value and concept membership assignments. The former are provided by value assignment, e.g., o.age := 18, which assigns the value 18 to o.age, and the latter is provided by concept assignment, e.g., $o \in$ ADULT, that implies the property o.age ≥ 18. We impose that an attribute value can be modified only through explicit value assignment.

As in [Katsuno and Mendelzon 91] we consider two types of situations: those where we have new information about a world that has not changed (which they call *revision*), and those where the world has changed, that is *update*. Accordingly, we consider two types of transactions: *update transactions*, which are the standard transactions in databases, and *completion transactions*, which are meant not to update the values of the attributes of the objects. A transaction is divided into two parts: the first part deals with elementary updates and the second expresses constraints on the objects by means of concept assignment.

First we recall general issues on Description Logics, then we describe our proposition for an update semantics and we relate our work to existing one. Finally, we present implementation issues.

2 Description Logics

Description Logics provide a possible underlying data model for our work. DL languages are characterised by their concept constructors; here we focus on the language ALCQI that is well adapted to object-oriented models [Calvanese et al. 95]. The motivation for the choice of this particular language is that it contains most OO formalisms such as Entity Relationship or ODMG. In the Osiris Prototype we use the RACER [Haarslev and Moller 01] system to detect incoherent schemes, to check constraints and to classify instances.

2.1 The TBox

A **TBox** (Term Box) is a triplet <C,R,A> where C is a set of concept names, R a set of role names and A a set of axioms. Informally, concepts are named sets of objects, roles are binary relationships between concepts, and axioms express constraints between concepts and roles. Given C and R, complex concepts can be constructed according to Chart 1. Notion of role is

very similar to those of attribute, an attribute is a role seen from one side. In the following we will use both terms.
An **interpretation** I is given by a set of entities (objects) Δ^I and an interpretation function $\cdot^I$ which maps each concept from C to a subset of Δ^I, and each role from R to a subset of $\Delta^I \times \Delta^I$. Complex concepts are interpreted through $\cdot^I$ accordingly to Chart 1.
Axioms are inclusion constraints between concepts (concepts from C or complex concepts), and then a **valid interpretation** is an interpretation that satisfies every axiom from A, i.e., if $C1 \subseteq C2$ is an axiom from A and I a valid interpretation then $C1^I \subseteq C2^I$.
Intuitively, a Tbox may be safely considered as a Database scheme and an interpretation as a possible state for this scheme where there is only complete information.

2.2 The ABox

An **ABox** (Assertion Box) is a finite set of assertions of the following forms: $o \in C$, $(o1,o2) \in R$, closed(o,R)

Constructor	Syntax	Semantics
Attribute Typing	$\forall$ R.A	$\{x \in \Delta^I \mid \forall y \in \Delta^I: (x,y) \in R^I \Rightarrow y \in A^I\}$
Intersection	$U \cap V$	$U^I \cap V^I$
Negation	$\neg U$	$\{x \in \Delta^I \mid x \notin U^I\}$
Union (U)	$U \cup V$	$U^I \cup V^I$
Qualified Cardinality Constraints (Q)	$(\leq n\ R\ C)$	$\{x \in \Delta^I \mid \#\{y \in C^I, (x,y) \in R^I\} \leq n\}$
	$(\geq n\ R\ C)$	$\{x \in \Delta^I \mid \#\{y \in C^I, (x,y) \in R^I\} \geq n\}$
Inverse Role (I)	R^{-1}	$\{(x,y) \mid (y,x) \in R^I\}$

Chart 1. Syntax and semantics for the language ALCQI.

The first form of assertions states that the object o belongs to the concept C, the second form that o1 and o2 are related through a role R and the third expresses that all the objects linked to o through role R are given explicitly by second form axioms in the Abox; it allows to distinguish between “Mark and Jack are some of Mike's brothers” and “Mark and Jack are all of Mike's brothers”. An ABox can be considered as a database state. ABox reasoning consists in determining if a given ABox is **consistent** with a TBox.

An ABox is said to be **complete** if all roles are closed for all objects and if for each named concept C and each object o either $o \in C$ or $o \in \neg C$. In other words, in a complete ABox there is no incomplete information. A complete ABox is equivalent to an interpretation. An ABox A is said to be **more precise** than an ABox B if $B \subseteq A$ (considering A and B as sets of axioms). A more precise ABox contains more information.
An ABox A is **valid for a TBox** T if and only if there exists a complete ABox I that is more precise than A and such that I is equivalent to a valid interpretation for T.
Therefore, although it is not explicitly mentioned, DL systems naturally deal with incomplete information.

2.3 Instance Classification

Description Logics systems provide an instance classification mechanism by allowing to ask if a given object belongs to a given concept; it is expressed in RACER with the *(instance-of? o C)* command that returns **true** if the object is proven to belong to C and **nil** otherwise.
When dealing with incomplete data, the negation of **true** is not **false**; there are three possibilities.

1. o belongs to C can be proven
2. o belongs to ¬C can be proven
3. none of them can be proven

Given an ABox A, **instance classification** for an object o can be performed by asking the following two questions for every concept from C: `(instance-of? o C)` and `(instance-of? o ¬C)`. If none answers true, membership of o to C is unknown.

3 A model for transaction updates

3.1 Context

Specifying an update requires saying what has changed in the world and what has not changed. There are often much more unchanged facts than changed ones, so specifying an update can need a lot of words. This issue is known as the *frame problem* [MacCarthy and Hayes 69]. We solve this issue by restricting to independence of roles and imposing explicit role updates, which is the standard proposition used both in programming languages and in databases.

As usual when dealing with state evolution, we assume that a state keeps as many of its constituting properties as possible. This is known as the law of inertia [Przymusinski and Turner 97]. For example, painting an object does not change its shape.
We impose that all role updates are explicitly expressed by the user, by means of elementary updates. This means that concept assignments does not change the value of the roles. In effect, concepts are defined by properties on role values.

3.2 States

We consider a countable infinite set of objects O, and a given TBox S=<**C**,**R**,**A**>, also referred as the **scheme**.
The **initial state** of a universe is the empty ABox. We consider that there is a **discrete time**, the **date** is a natural number which indicates how many transactions or completions occurred since the initial state. We indicate states by their date, e.g., a state indexed by the date n refers the state after n successive transactions/completions occurred starting from the initial state.
In order to capture information evolution and explain the global semantics, we build, for each date, a TBox with concepts, roles and axioms corresponding to S. For each date j we consider a TBox S(j) such that:

- for each r in **R** and concept c in **C** there are respectively a role r(j) and a concept c(j) in S(j)
- S(j) contains each axiom from **A** where a role r (respectively concept c) is replaced by r(j) (respectively c(j))

Intuitively c(j) represents the state of the concept $c \in \mathbf{C}$ at the date j.

Example. Let us consider the following initial schema S :
PERSON $\subseteq$ $\forall$friend. PERSON $\cap$ $\forall$cars. CAR
CAR $\subseteq$ $\forall$cars^{-1}. PERSON $\cap$ $\forall$colour. (STRING1$\cup$STRING2) $\cap$ (= 1 colour)
STRING1 $\subseteq$ $\neg$STRING2
BLACK-CAR = CAR $\cap$ $\forall$colour. STRING1

Thus S(j) is the following schema:
PERSON(j) $\subseteq$ $\forall$friend(j). PERSON(j)$\cap$ $\forall$cars(j). CAR(j)
CAR(j) $\subseteq$ $\forall$cars^{-1}(j). PERSON(j) $\cap$ $\forall$colour(j). (STRING1(j)$\cup$STRING2(j)) $\cap$ (= 1 colour(j))
STRING1(j) $\subseteq$ $\neg$STRING2(j)
BLACK-CAR(j) = CAR(j) $\cap$ $\forall$colour(j). STRING1(j)

A **role state at date n**, RS(n), is a set of constraints of the following form :

- (o1,o2) $\in$ role(n)
- closed(o, role(n))

A **membership state at date n**, MS(n), is a set of constraints of the following form :

- $\in$ C(n)

The idea is to separate role values from concept membership. On one hand, role values are explicitly changed and those that are not modified retain their value from state to state (law of inertia). On the other hand, concept membership may change according to role value changes. The purpose and meaning of this indexation is to describe what was true in a past state and collect all the assessed information into one single global state in which reasoning can be achieved. If one says : "in 1945, the president was Mr X1" and "in 1973, the president was Mr X2", there is no contradiction. Labelling a fact by the time it is true (or assessed to be true) allows to construct a coherent logical state (in fact it depends of the facts but the incoherency is not due to the process of modification in itself). We could not find any related work using this idea that we believe fundamental for modelling deduction in case of updates.

Given a set of ABox axioms X over the TBox S(n), we define **X$\uparrow$** as the set of axioms from X where each occurrence of n is replaced by n+1.

Example. If X = {(o1,o2) $\in$ role(n), o1 $\in$ c(n)} then X$\uparrow$ = {(o1,o2) $\in$ role(n+1), o1 $\in$ c(n+1)}.

3.3 Elementary updates and completions

An **elementary update** on a state is one of the following operations expressed through an object-oriented syntax:

1. Adding a relationship between two objects **o1.role.add(o2)**
2. Removing a relationship between two objects **o1.role.remove(o2)**
   It is possible only if (o1,o2) $\in$ role(n) is an axiom of the state
3. Modifying an object **o.role := value**

4. Modifying an object **o.role in value**
5. Deleting an object **o.delete**
   It is only possible on a state where o is an object

Here **role** can be either a member of **R** or **role** = r^{-1} with r in **R**.

Given an attribute state RS labelled by n and an elementary update u, we define **RS◊u** as the result of the operation u on state RS :

1. RS◊ **o1.role.add(o2)** = RS $\cup$ {(o1,o2) $\in$ role(n)}
2. RS◊ **o1.role.remove(o2)** = RS - {(o1,o2) $\in$ role(n)}
3. RS◊ **o1.role:=value** = RS - {(o1,x) $\in$ role(n) | "(o1,x) $\in$ role(n)" $\in$ RS} $\cup$ {(o1,o2) $\in$ role(n) | o2$\in$value} $\cup$ {closed(o1,role)}
4. RS◊ **o.role in value** = RS - {(o1,x) $\in$ role(n) | "(o1,x) $\in$ role(n)" $\in$ RS} $\cup$ {(o1,o2) $\in$ role(n) | o2$\in$value} - {closed(o1,role)}
5. RS◊ **o.delete** = RS – {(o,x) $\in$ role(n) | "(o,x) $\in$ role(n)" $\in$ RS}

We say that **an elementary update u modifies the role r for an object o** iff there exists an object x such that "(o,x) $\in$ r" $\in$ RS and "(o,x) $\in$ r" $\notin$ RS◊u, or "(o,x) $\in$ r" $\notin$ RS and "(o,x) $\in$ r" $\in$ RS◊u.

An elementary completion on a state is one of the following operations expressed through an object-oriented syntax:

1. Adding a relationship between two objects **o1.role.add(o2)**
2. Closing an attribute value **o.role.close**

Given an attribute state RS labelled by n and an elementary completion c, we define **RS◊c** as the result of operation c on state RS :

1. RS◊ **o1.role.add(o2)** = RS $\cup$ {(o1,o2) $\in$ role(n)}
   It is only possible if closed(o1,role) $\notin$ RS and closed(o2,role^{-1}) $\notin$ RS
2. RS◊ **o.role.close = RS** $\cup$ {closed(o,role)}

Remark. Because of inverse attributes, in the case where role = r^{-1}, we re-write "(x,y) $\in$ role(n)" by "(y,x) $\in$ r (n)" in all above formulas.

3.4 Transactions

A **transaction** is a sequence of elementary updates {U_1, …,U_k} constrained through **object constraints** {C1, …, Cm} that are axioms of the form *object∈concept* and that are asserted to be true at the end of the transaction.

The **global theory** for the initial state is the empty set. Now let us consider the recursive case. First we have to define the **partial theory**, PT(n+1), for the state at date n+1 **after transaction T**=<{U_1,…,U_k},{C1,…Cm}> executed on state n as the union of :

- The global theory for state at date n
- RS(n+1) = (RS(n)$\uparrow$)◊U_1◊ ... ◊U_k
- {C1, ..., Cm}
- axioms o.r(n+1) = o.r(n) for objects whose attribute r is not modified

The partial theory after a transaction is used to check the validity of the transaction: a **transaction is valid** if the partial theory after the transaction is coherent with regards to the union of S(0), ..., S(n). The transaction is aborted if it is not valid.

Example. Let us consider the scheme from example 1. The global theory for the initial state is {}. First we apply to that state the transaction T1 = <{o1.cars:={o2,o3}, o2.colour = black},{o1∈PERSON, o2∈CAR, o3∈CAR, black∈STRING1}>.
The **partial theory for the state at date 1** is :

> {(o1,o2)∈cars(1), (o1,o3)∈cars(1), closed(o1,cars(1)), (o2,black)∈colour(1), closed(o2,colour(1)), o1∈PERSON(1),o2∈CAR(1), o3∈CAR(1), black∈STRING1(1)}

→ It is a coherent set of constraints so the transaction is valid.

In the partial theory we have lost all the concept membership axioms from the state n. In order to recover as many of these axioms as possible we define :

$$PM(n+1) = \{ X \subseteq MS(n) \mid X\uparrow \text{ is coherent with } PT(n+1)\}$$

Intuitively, an element from PM(n+1) represents a set of concept membership axioms from the sate at date n that are still compatible (if asserted to be true at time n+1) with the partial theory of the state at date n+1. Elements from PM(n+1) are partially ordered by inclusion.
PM represents the previous true axioms that could be safely added to the new state and we call an element of PM (j) a **set of plausible previous membership axioms for state j**. We consider that for a valid transaction to state i+1, the axioms from the intersection of the maximum elements of PM (i+1) are true in the state i+1. This allows recovering axioms on objects that have not been modified, and recover membership to unaffected concepts for modified objects. This is a criterion of minimal change as in [Fagin et al. 83].

We systematically classify each object o, i.e., for each named concept C there is either an axiom o∈C or o∈¬C, or none if neither can be proven. We keep the object membership to each named concept for each object. Systematic instance classification is not usual in DL nor in DB. The purpose of this feature is the maintenance of persistent views, which is one of the aims of the Osiris system [Simonet et al. 94; Roger et al. 01]. This classification is performed through the third point of the following definition.

If the partial theory is valid then we can define the **global theory**, GT(n+1), **for the state at date n+1** as the union of :

1. The partial theory for the state n+1
2. The intersection of all maximal elements from PM(n+1)
3. For each object o and each concept C, "o∈C" if it is compatible with the two preceding sets of constraints, or "o∈¬C", or nothing if none are compatible

Example. In continuation of the preceding example, the **global state** at date 1 is :

> {(o1,o2)∈cars(1), (o1,o3)∈cars(1), closed(o1,cars(1)), (o2,black)∈colour(1), closed(o2,colour(1)), o1∈PERSON(1),o2∈CAR(1), o3∈CAR(1), black∈STRING1(1), o2∈BLACK-CAR(1)}

Now we apply the transaction T2=<{o2.delete},{ o1∈PERSON, o3∈CAR, black∈STRING1}>. This implies the following **partial state** :

> GT(1) ∪ {(o1,o3)∈cars(2), closed(o1,cars(2)), o1∈PERSON(2), o3∈CAR(2), black∈STRING1(2), o1.friends(2)=o1.friends(1), o3.colour(2)=o3.colour(1), o3.cars^{-1}(2) = o3.cars^{-1}(1)}

→ this is a coherent set of axioms so the transaction is valid. The **global theory** for the state at date 2 is :

> GT(1) ∪ {(o1,o3)∈cars(2), closed(o1,cars(2)), o1∈PERSON(2), o3∈CAR(2), black∈STRING1(2), o1.friends(2)=o1.friends(1), o3.colour(2)=o3.colour(1), o3.cars^{-1}(2) = o3.cars^{-1}(1)}

3.5 Completions

A **completion** is a sequence of elementary updates {U_1, …,U_k} constrained through **object constraints** {C1, …, Cm} that are axioms of the form *object∈concept* and that are asserted to be true at the end of the transaction.

The **partial theory**, PT(n+1), for the state at date n+1 **after a completion C**=<{c_1, ..., c_k}, },{CM_1,…CMm}> executed on a state n is the union of :

1. The global theory for state at date n
2. RS(n+1) = (RS(n)↑)◊c_1◊ ... ◊c_k
3. {CM_1, ..., CMm} where each concept C is replaced by C(n+1)
4. MS(n)↑
5. axioms o.r(n+1) = o.r(n) for each object o and attribute r such that o.r is not modified

A completion is **valid** if the global theory after the completion is coherent.
The **global theory**, GT(n+1), for the state at date n+1 **after a valid completion C** is the union of :

1. the partial theory for the state at date n+1 after the completion C
2. all membership axioms that can be deduced from this partial theory

Example. Let us apply the following completion to the state at time 2 of our previous example, C=<{o3.colour.add(red), closed(o3,colour)},{red∈STRING2}>.

The **partial theory** is :

> GT(2) ∪ {{(o1,o3)∈cars(3), closed(o1,cars(3)), (o3,red)∈colour(3), closed(o3,colour(3)), red∈STRING2(3), o1∈PERSON(3), o3∈CAR(3), black∈STRING1(3), o1.friends(3)=o1.friends(2), o3.cars^{-1}(3) = o3.cars^{-1}(2), o1.cars(3)=o1.cars(3)}

→ it is coherent
The **global theory** is:

> PT(3) ∪ {o3∈¬BLACK-CAR}

4 Comparison with previous work

Here we compare to Possible Model Approach for knowledge base update from [Katsuno and Mendelzon 91]. First we need to relate terms from this work to ours. Here **a knowledge base** corresponds to a state in conjunction with the scheme, so a **model** is a model for the conjunction of the axioms from the state and from the scheme. **Mod(Φ)** denotes the set of all the models of a set of axioms Φ. Given a pre-order $\leq$ on models and a set of models M, **Min(M,≤)** denotes the set of minimal elements for $\leq$ in M.

This comparison relies on defining a pre-order on models. Given two models I and A we define $O_{Attr(I)}(A)$ (respectively $O_{Mem(I)}(A)$) as the set of objects o such that o has the same attribute values (respectively membership values) in A as in I. Given a model I we define the partial pre-order on models $\leq_I$ as follows: A $\leq_I$ B if A and B satisfy one of these conditions:

- $O_{Attr(I)}(B)$ is a strict subset of $O_{Attr(I)}(A)$
- $O_{Attr(I)}(B)=O_{Attr(I)}(A)$ and $O_{Mem(I)}(B)$ is a strict subset of $O_{Mem(I)}(A)$
- $O_{Attr(I)}(B)=O_{Attr(I)}(A)$ and $O_{Mem(I)}(B)=O_{Mem(I)}(A)$ and A has lesser objects than B

This pre-order is **faithful**, i.e., for J≠I we have I $<_I$ J. It compares models according to attribute values first, then to membership and finally to minimal addition of objects. Lemma 1 states that minimal models of the change according to this pre-order give the model of the changed state.

Lemma. If we consider a state Φ and a transaction μ, if the application of transaction μ on state Φ succeeds, then the resulting state Φ◊μ (with respect to semantics expressed in 3.3) is such that:

$$Mod(\Phi\lozenge\mu) = \cup_{I \in Mod(\Phi)} Min(Mod(\mu), \leq_I) \qquad (1)$$

Katzuno and Mendelzon showed that update operators satisfying (1) for a faithful pre-order $\leq_I$ are exactly those that satisfy axioms (U1)-(U9) (that are the correspondents for update of AGM axioms for revision). Note that Katzuno and Mendelzon worked on a propositional logic model, whereas we are working on an object model. Work from [Simonet et al. 94; Roger et al. 01] suggests that correspondences could be done between DL and propositional logic.

5 Implementation issues

In this section we describe how computation of a new state is done. Complexity is in calculating the intersection of maximal unchanged facts (or union of minimal changed facts). If made in a simple way this requires trying all possibilities. So we focus here on possible optimisations.

5.1 Optimisations

The first optimisation is to concentrate only on objects both updated and recursively connected to updated objects. Concerning concept membership, the objects that are not modified or that are not recursively related to any modified object must not change classification: they are part of all maximal plausible previous membership axioms.

The second idea is to derive facts on unknown attribute values. Membership constraints are used to derive facts on unknown attribute values for objects. Each membership constraint is put in normal form, i.e., it must not contain any named concepts (these names are replaced by their definition) or any known attribute constraint (they are replaced by their truth value):

Consider the TBox C1 = ∀A.C2 ∩ ∀B.C1 and state

```
(o1,o2) in A; (o1,o3) in A
closed(o1,A); closed(o2,A); closed(o3,A)
o1 in C1; o2 in C2; o3 in C2
```

the normal forms for the membership constraints are

```
o1 in ∀B.C1
```

If a transaction does not update `o1.B` then `o1 in ∀B.C1` is still true in the updated state. If a transaction updates an unknown value, then all occurrences of that attribute for normal form membership axioms concerning the object are marked as "obsolete". When an axiom in normal form has its entire attribute marked as "obsolete" then it is not useful anymore and may be forgotten. Axioms in normal form are a kind of value constraints on unknown attributes.

5.2 Update Propagations and Instance Re-Classification

During a transaction, all modified instances are marked with a tag "to be classified". In order to be classified these instances need classification information on related instances, so these related instances are marked with "to be used", and they are also marked "to be classified" if they logically depend on an object also marked "to be classified".

At the end of the transaction, considering each marked object:

- each marked object is assigned a unique name; built from its oid, for example.
- if it is marked "to be classified" then it is translated into ABox assertions according to its attribute values.
- if it is marked "to be used" and not marked "to be classified", then, as the object did not change classification during the transaction, it must not be classified and its previous classification is still valid and will be used. The object is translated into axioms of the form *object belongs to class* or *object does not belong to class*, one for each known concept membership

Then the object constraints are asserted in the Abox. Thus the DL system is able to determine coherence. Classification of "to be classified" objects consists in evaluating queries as described in 2.3 for each named concept and for each object.

6 Conclusion

DLs and DBs are two kinds of system bound to merge in order to respectively increase reasoning power in DB (instance classification, query optimisation) and the amount of data efficiently processed in DL. But this convergence is hindered by the distance between

database and DL representation models. Thus, works on semantic query optimisation rely on persistent view maintenance (through defined concepts), but the lack of such a mechanism in most databases renders them unusable through this approach. DB and DL coupling requires both a scheme and an update translation; whereas scheme translation has already been studied, update semantics for DL used in a database context is still lacking.
We have suggested a semantics that allows post-conditions on transactions and constraint checking, which is an important feature for database issue. It also deals with incomplete information. This semantics is compatible with the axioms proposed by Katsuno and Mendelzon and the mainstream in database and programming languages, i.e., attributes only change through explicit assignments.
This solution uses a DL system to make inferences and checks but also relies on optimisations not directly translatable in DL, which suggests using a DL system as a component in a more complex architecture. We are conducting implementation for this solution in the context of the Osiris system, and we use RACER as the DL sub-system. Our work is still in progress and is limited both in a theoretical and a practical way. We are investigating the reduction of restrictions on concept assignments. Moreover, only tests may validate the proposed optimised mechanisms. This work is a first step toward a finer integration between description logics and databases.

7 References

[Borgida and Brachman 93] Alex Borgida and Ronald Brachman, Loading Data into Description Reasoners, SIGMOD Record,vol 22, N° 2, 1993
[Bresciani 95] Paolo Bresciani, Querying Database from Description Logics, KRDB'95
[Calvanese et al. 95] Diego Calvanese, Guiseppe De Giacomo and Maurizio Lenzerini, Structured Objects: Modeling and Reasoning, DOOD'95
[Fagin et al. 83] R. Fagin, J Ullman and M. Vardi, On the semantics of Updates in Databases, 2nd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems, Atlanta, 1983
[Haarslev and Moller 01] Volker Haarslev and Ralf Moller, RACER System Description, IJCAR’01
[Katsuno and Mendelzon 91] Hirofumi Katsuno and Alberto Mendelzon, On the Difference Between Updating a Knowledge Base and Revising it, KR'91
[MacCarthy and Hayes 69] J. M. MacCarthy and P. J. Hayes, Some Philosophical Problems from the Standpoint of Artificial Intelligence, Machine Intelligence, Vol. 4, 1969
[Przymusinski and Turner 97] Teodor C. Przymusinski and Hudson Turner, Update by Means of Inference Rules, JLP 30(2), pp 125-143, 1997
[Roger et al. 01] Mathieu Roger, Ana Simonet et Michel Simonet, Object Space Partitioning in a DL-like Database and Knowledge Base Management System, DEXA 2001
[Simonet et al. 94] Ana Simonet and Michel Simonet, Objects with Views and Constraints: From Databases to Knowledge bases, OOIS’94