Object knowledge base revision

Isabelle Crampé and **Jérôme Euzenat**[*]

Abstract. A revision framework for object-based knowledge representation languages is presented. It is defined by adapting logical revision to objects and characterised both semantically and syntactically. The syntactic analysis of revision shows that it can be easily interpreted in terms of object structures (e.g. moving classes or enlarging domains). This is the source of the implementation and it enables users to be involved in the revision process.

There has been comprehensive work done on revision in logical formalisms. However, due to several factors (complexity of revision and decision procedures, needs not well-stated [7] and mismatch between logical formalisms and implemented knowledge representation systems), little have been implemented. Moreover, very few studies (with the notable exception of [11]) have been devoted to revision in object formalisms.

The present work attempts to transfer the results of logical revision to the field of object-based knowledge representation systems. Moreover, its goal is to come up with an implemented system in the context of error correction during knowledge base editing.

Two noticeable points are thus taken into account:

- the revision must take advantage of the structure of object formalisms (the developed operator is grounded on a language which enables a localised syntactic characterisation of inconsistency and a natural interpretation of minimal revisions in terms of objects);
- the user can be interactively asked about the possible revisions.

The paper begins by formalising a basic object-based language (§1) in order to formally characterise the kind of revision which is being performed. Then we introduce a syntactic manipulation of the knowledge bases required in order to transpose usual revision notions such as contraction and minimality (§2). In section 3, revision is defined and minimality criterion is introduced. They are found to closely follow the object-based structure. In the last section, we finally present the implementation of revision and its embedding in an actual knowledge representation system.

Proof of propositions, extensions and extensive examples can be found in [3].

1. OBJECT REPRESENTATION LANGUAGE

The definition of a language corresponding to object-based notions has two purposes: formally defining what objects mean and allowing an easier transposition from logical revision to object models. Although the language presented here might seem simple, it covers in fact a large part of object representation systems.

1.1 Language

As expected, the representation language contains objects and classes. They have attributes which are typed in the classes and valued in the objects. The values of attributes can be either objects or basic values (integers in the examples). Moreover, there is a notion of sub-class relation between classes and of attachment of an object to a class. The grammar of the language is presented below.

Definition 1 (Grammar).

<KB> ::= {<assertion>*}
<assertion> ::= <spec> | <attachment>
| <class_slot> | <inst_slot>
<spec> ::= <class_name> ≤ <class_name>
<attachment> ::= <inst_name> ∈ <class_name>
<class_slot> ::= <class_name>.<slot_name>⊆<domain>
| <class_name>.<slot_name>⊆<class_name>
<inst_slot> ::= <inst_name>.<slot_name> = <value>
| <inst_name>.<slot_name> = <inst_name>
<class_name>|<inst_name>|<slot_name> ::= <identifier>
<identifier> are alphanumeric strings. <value> and <domain> are integer and intervals, respectively.

Moreover, there are meta-syntactic constraints:

- A knowledge base is a finite set of assertions;
- The graph of the ≤ relation, called specialisation, is acyclic;
- The graph of the ⊆ relation, called slot restriction, (composed with ≤) is acyclic.

It can be noted that classes and instances can be introduced more than once: multi-specialisation (several super-classes for a class) and multi-instanciation (object attached to several classes) are allowed. Figure 1 graphically presents such a knowledge base.

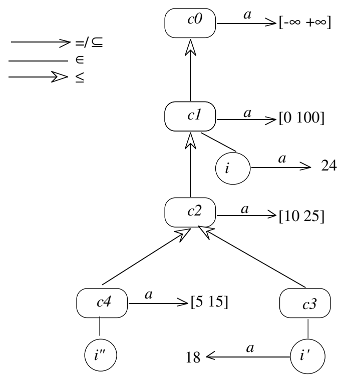

Figure 1: $B=\{c_1 \leq c_0 ; c_2 \leq c_1 ; c_3 \leq c_2 ; c_4 \leq c_2 ; i \in c_1 ; i' \in c_3 ; i'' \in c_4 ; c_0.a \subseteq [-\infty +\infty] ; c_1.a \subseteq [0\ 100] ; c_2.a \subseteq [10\ 25] ; c_4.a \subseteq [5\ 15] ; i.a=24 ; i'.a=18\}$.

1.2 Semantics

The semantics of the language is given by reference to a domain of interpretation. The objects are interpreted as the elements of this domain while classes are subsets of it. The specialisation relation is interpreted as subset and attachment as set membership.

This semantics is not that of terminological logics [11] because the language presented here is purely descriptive (there is no definition of the classes as necessary and sufficient

[*] INRIA Rhône-Alpes
655 avenue de l'Europe, 38330 Montbonnot Saint-Martin (France)
{Isabelle.Crampe,Jerome.Euzenat}@inrialpes.fr
Corrected version: 31/08/1998

ECAI 98. *13th European Conference on Artificial Intelligence*
Edited by Henri Prade
Published in 1998 by John Wiley & Sons, Ltd.

conditions) and not all formulas that can be assigned a truth statement are reducible to subsumption. Moreover, the language has been designed to correspond to those many people work with every day. Another semantics that comes to mind is that of frame logics [10]. It is not used here because our semantics of specialisation is more precise (it is strictly identified with set-inclusion and not with an arbitrary order relation).

Definition 2 (Interpretation).
Let B be a knowledge base (with O, C, A as set of object, class and attribute names respectively and τ as the set of possible values) and D a domain. An interpretation is a pair $<D,I>$, in which I is called interpretation function:

I: $O \rightarrow D$ injective

$C \rightarrow 2^D$

$A \rightarrow (D \rightarrow \tau \cup D)$ $I(a)$ is total

Injectivity of I on O ensures the unique name assumption for objects.

Definition 3 (Assertion satisfaction).
An interpretation $<D,I>$ satisfies an assertion δ (written $\models_{<D,I>} \delta$) if

$\models_{<D,I>} c \leq c'$	iff	$I(c) \subseteq I(c')$
$\models_{<D,I>} c.a \subseteq d$	iff	$I(a \mid c) \subseteq I(d)$
$\models_{<D,I>} o \in c$	iff	$I(o) \in I(c)$
$\models_{<D,I>} o.a = v$	iff	$I(a)(I(o)) = I(v)$
$\models_{<D,I>} c.a \subseteq c'$	iff	$I(a \mid c) \subseteq I(c')$
$\models_{<D,I>} o.a = o'$	iff	$I(a)(I(o)) = I(o')$

$I(a|c)$ is the domain of the interpretation of slot a when the codomain is restricted to c.

As usual, a knowledge base *model* is an interpretation which satisfies all assertions, an assertion is a *consequence* of a knowledge base if it is satisfied in every models, and a knowledge base is *consistent* if it has at least one model (and inconsistent if it has none).

1.3 Deduction

Modelling deduction is useful for revision since inconsistency is sometimes raised by inferring in the knowledge base and since the set of deduction rules can help to revise through an abductive process.

A deduction procedure is provided as a set of inference rules specific to object representation systems. Some rules ($\leq$-reflexivity) are technical rules (enabling to obtain completeness), and others are well-known ($\leq$-transitivity, inheritance). $\in$-inference is an original rule (suggested in [4]) providing a kind of type-inference not provided by many systems but required by the above semantics.

Definition 4 (Inference rules).
$\forall$ c, c', c'', $c_i \in C$, $o \in O$, $a \in A$, $v \in V$ and d, d', $d_i \in T$:

(1) ($\leq$-reflexivity) $$\frac{}{c \leq c}$$

(2) ($\leq$-transitivity) $$\frac{c \leq c' \quad c' \leq c''}{c \leq c''}$$

(3) ($\in$-closure) $$\frac{c \leq c' \quad o \in c}{o \in c'}$$

(4) (d-intersection) $$\frac{c.a \subseteq d \quad c.a \subseteq d'}{c.a \subseteq d \wedge d'}$$

(5) (d-inheritance) $$\frac{c.a \subseteq d \quad c' \leq c}{c'.a \subseteq d}$$

(5') (c-inheritance) $$\frac{c.a \subseteq c'' \quad c' \leq c}{c'.a \subseteq c''}$$

(6) (d-closure) $$\frac{c.a \subseteq d}{c.a \subseteq d'} \quad d <: d'$$

(6') (c-closure) $$\frac{c.a \subseteq c' \quad c' \leq c''}{c.a \subseteq c''}$$

(7) (o-valuation) $$\frac{\{c_i.a \subseteq d_i \quad o \in c_i\}}{o.a = v} \quad \{v\} = \bigwedge_i d_i$$

(8) ($\in$-inference) $$\frac{c.a \subseteq c' \quad o \in c \quad o.a = o'}{o' \in c'}$$

The *closure* (Cn(B)) of a knowledge base B by the rules is defined by the set of assertions obtainable by applying the rules and *deducibility* by membership of an assertion to the closure.

Definition 5 (Deduction).
An assertion δ is deducible from a knowledge base B (written $B \vdash \delta$), iff $\delta \in$ Cn(B).

1.4 Completeness

As expected the deduction is sound and complete with regard to the semantics.

Proposition 1 (Soundness).
Let B be a knowledge base and δ an assertion,

if $B \vdash \delta$ then $B \models \delta$.

Proposition 2 (Partial completeness).
Let B be a *consistent* knowledge base and δ an assertion,

if $B \models \delta$ then $B \vdash \delta$.

It is noteworthy that the completeness property is only partial because it requires the consistency of the knowledge base. As usual, when a knowledge base is inconsistent (i.e. has no model) every assertion is true in all models (since there is none). In the present language, total completeness could be achieved (with the help of results below) by adding inference rules enabling to infer anything from evidence of inconsistency.

However, retaining partial consistency allows to keep the inconsistency local (at its source) instead of spreading it to the whole base. This enables the revision process to start from a relevant position.

2. NORMAL FORM AND SYNTACTIC TREATMENT

While a theoretical presentation of an object representation language has been provided, actual implementations are very different. A normalisation process is usually applied to the knowledge base enabling to compact it and to retrieve knowledge efficiently.

A normal form for knowledge bases is presented here because it allows to define syntactic treatments on the knowledge bases used in revision. A constructive characterisation of the normal form is given before syntactically characterising inconsistency.

2.1 Normal form

The definition of the normal form is first provided axiomatically through three properties. Basically, it is a form equivalent to the initial knowledge base (1 and 2) in which the deletions are effective (3, i.e. if an assertion is deleted from the base, it is not deducible from it anymore). This property is invaluable when revising because it discards some non minimal revisions.

Definition 6 (Normal form).
The normal form of a base B, written NF(B), is such that:
(1) $B \vdash \mathrm{NF}(B)$,
(2) $\mathrm{NF}(B) \vdash B$,
(3) for each δ of NF(B), $\mathrm{NF}(B)-\{\delta\} \nvdash \delta$.

Proposition 3 states that the normal form exists, is unique and can be obtained by a simple procedure. It roughly consists of applying the deduction rules backward, starting with the knowledge base closure. Of course, the implementation does not start with the deductive closure but rather mixes inference and computation of the normal form.

Proposition 3 (Characterisation of normal form).
Let B be a base, the normal form of B is computed by:

$$
\begin{aligned}
\mathrm{NF}(B) = [\mathrm{Cn}(B)-\{(c\leq c)\in \mathrm{Cn}(B)\}] \\
&- \{(c\leq c'') ; (c\leq c'), (c'\leq c'')\in \mathrm{Cn}(B)\} \\
&- \{(o\in c) ; (o\in c'), (c'\leq c)\in \mathrm{Cn}(B)\} \\
&- \{(c.a\subseteq d) ; (c.a\subseteq d')\in \mathrm{Cn}(B) \text{ et } d'<:d\} \\
&- \{(c.a\subseteq c') ; (c.a\subseteq c''), (c''\leq c')\in \mathrm{Cn}(B)\} \\
&- \{(o.a=v) ; (c_i.a\subseteq d_i)\in \mathrm{Cn}(B) \text{ et } \{v\} = \textstyle\bigwedge_i d_i \} \\
&- \{(c.a\subseteq d) ; \{c_i.a\subseteq d_i, c\leq c_i\}\in \mathrm{Cn}(B), \textstyle\bigwedge_i d_i = d \} \\
&- \{(c.a\subseteq c') ; (c''.a\subseteq c'), (c\leq c'')\in \mathrm{Cn}(B)\} \\
&- \{(o\in c) ; (o'\in c'), (c'.a\subseteq c), (o'.a=o)\in \mathrm{Cn}(B)\}
\end{aligned}
$$

2.2 Inconsistency

Among the advantages of the normal form is the opportunity to syntactically isolate the source of inconsistency. This, combined with the partial completeness property above, allows for the immediate localisation of inconsistency and the triggering of abductive repairing mechanisms.

Proposition 4 (Syntactic inconsistency).
A knowledge base B is inconsistent iff one of the following propositions is true:
(1) $(o.a=x)\in B$, $(o.a=x')\in B$ with $x\neq x'$.
(2) $B\vdash o\in c$, $B\vdash c.a \subseteq d$, $(o.a=v)\in B$ with $\neg v{:}d$
(3) there exists I, such that for each $i\in I$ $B\vdash o\in c_i$, $(c_i.a_1.a_2.\ldots a_n\subseteq^* d_i)\in B$ and $\bigwedge_{i\in I} d_i=\emptyset$

with $c.a_1.a_2.\ldots a_n\subseteq^* d$ standing for "there exist $c_1,\ldots c_n$, $c'_1,\ldots c'_n$ such that for each $1\leq j\leq n$ $(c_j\leq c'_j)\in B$ and $(c'_j.a_j\subseteq c'_{j+1})\in B$ (with $c=c_1$ and $(c'_n.a_n\subseteq d) \in B$)".

Proposition 5 (Complexity).
The complexity of obtaining the normal form and detecting inconsistency is polynomial.

3. REVISION

The definition of revision on the object representation language is provided below (§3.1). Minimality is defined and can be characterised both semantically and syntactically (§3.2). Moreover, this minimality has a natural interpretation in the context of object languages (§3.3). This allows to apply it straightforwardly to actual knowledge bases (§4).

3.1 Revised knowledge bases

Here are classically introduced the notions of contracted base (a consistent subset of a knowledge base inconsistent with a particular assertion) and revised base (a contracted base to which the problematic assertion has been added).

Definition 7 (Contracted knowledge base).
Let B be a consistent base and δ an assertion such that $B\cup\{\delta\}$ is inconsistent. B' is a contracted base of (B,δ) iff $\mathrm{Cn}(B')\subset\mathrm{Cn}(B)$, (that is $B' \not\models B$ and $B\models B'$) and $B'\cup\{\delta\}$ is consistent.

Definition 8 (Revised knowledge base).
Let B be a consistent base and δ an assertion such that $B\cup\{\delta\}$ is inconsistent. B' is a revised base of (B,δ) iff $B'=B''\cup\{\delta\}$ with B'' a contracted base of (B,δ)

3.2 Minimality

Among the various revised knowledge bases, a partial order can be provided enabling to select the closest revisions to the initial knowledge base. This order is based on consequence (or entrenchment).

Definition 9 (Order between contracted knowledge bases).
Let B' and B'' be contracted knowledge bases of (B,δ), $B'\propto B''$ iff any of the following equivalent propositions is satisfied:
(1) $B'\models B''$
(2) $\mathrm{Cn}(B'')\subseteq\mathrm{Cn}(B')$

Thus the notion of minimality is strictly based on the conservation of the maximum of deductions (or the limitation of new models).

Definition 10 (Minimal contraction).
B' is a minimal contraction of (B,δ) iff B' is a contracted knowledge base of (B,δ) and there is no $B''\neq B'$ such that B'' is a contracted knowledge base of (B,δ) and $B''\propto B'$.

In order to manipulate contracted knowledge bases, we need to write them as finite assertion sets, namely using normal form. The normal form of a contracted knowledge base of (B,δ) B' can always be written as $B'=\mathrm{NF}(B')= (\mathrm{NF}(B)- B'_-)\cup B'_+$ with the following constraints:

$$B'_- \cap B'_+ =\emptyset,\ B'_- \subseteq \mathrm{NF}(B) \text{ and } B'_+ \subseteq \mathrm{Cn}(B)$$

This corresponds to the normal form of B from which a set B'_- is deleted and another set B'_+, deducible from B, is added.

The minimality can be characterised semantically and syntactically.

Proposition 6 (Semantic characterisation).
B' is a minimal contracted knowledge base of (B,δ) iff $\forall\gamma$ such that $B\models\gamma$ and $B'\cup\{\delta\}\not\models\gamma$ then $B'\cup\{\delta\}\cup\{\gamma\}$ is inconsistent.

Proposition 7 (Syntactic characterisation).
Let B' and B'' contracted knowledge bases of (B,δ), then B' is smaller than B'' $(B'\propto B'')$ iff $B'_- \subseteq B''_-$ and $B' \vdash B''_+$.

The two characterisations are equivalent.

3.3 Object interpretation

One strength of the resulting revision is that it can be interpreted in terms of objects instead of logics. In this particular case, object terms can be substituted for logic ones. In fact, the revision of an object-based knowledge base is a set of the following modifications:
- moving an object upward in the class hierarchy;
- deleting a value of an object slot;

- enlarging a class slot restriction (by enlarging the domain or using a more general class);
- moving a class upward in the class hierarchy.

Minimality itself has a behaviour that translates into objects.

Proposition 8 (Object interpretation of minimality).
B' is a minimal contracted knowledge base of (B,δ) iff:

(1) B' is a contracted knowledge base of (B,δ).

(2) $\forall\ B''$ a contracted knowledge base of (B,δ), if $B''_- \subseteq B'_-$ and $B'_+ \subseteq B''_+$, then $B'=B''$.

(3) $\forall (c \leq c') \in B'_+$, $\forall k \neq c'$, if $B' \vdash k \leq c'$ and $(c \leq k) \in \mathrm{Cn}(B)$, then $(c \leq k)$ cannot be added in $B' \cup \{\delta\}$.

(4) $\forall (o \in c) \in B'_+$, $\forall c' \neq c$, if $B' \vdash c' \leq c$ and $(o \in c') \in \mathrm{Cn}(B)$, then $(o \in c')$ cannot be added in $B' \cup \{\delta\}$.

(5) $\forall\ (c.a \sqsubseteq d) \in B'_+$, $\forall (c',d') \neq (c,d)$, if $d' \leq: d$, $B' \vdash c \leq c'$ and $(c'.a \sqsubseteq d') \in \mathrm{Cn}(B)$, then $(c'.a \sqsubseteq d')$ cannot be added in $B' \cup \{\delta\}$.

(6) $\forall\ (c.a \sqsubseteq k) \in B'_+$, $\forall (c',k') \neq (c,d)$, $k,k' \in C$, if $B' \vdash k' \leq k$, $B' \vdash c \leq c'$ and $(c'.a \sqsubseteq k') \in \mathrm{Cn}(B)$, then $(c'.a \sqsubseteq k')$ cannot be added in $B' \cup \{\delta\}$.

(7) $\forall (o \in c) \in B'_+$, $\forall c' \neq c, o', a$ if $(c'.a \sqsubseteq c) \in \mathrm{Cn}(B)$, $(o' \in c') \in \mathrm{Cn}(B)$ and $(o'.a = o) \in \mathrm{Cn}(B)$, then one of the three assertions cannot be added in $B' \cup \{\delta\}$.

(8) $\forall (o.a = v) \in B'_+$, for each minimal assertion set $\{c_i.a \sqsubseteq d_{ij}, o \in c_i\} \in \mathrm{Cn}(B)$ such that $\{v\} = \wedge d_{ij}$, then one of the assertions cannot be added in $B' \cup \{\delta\}$.

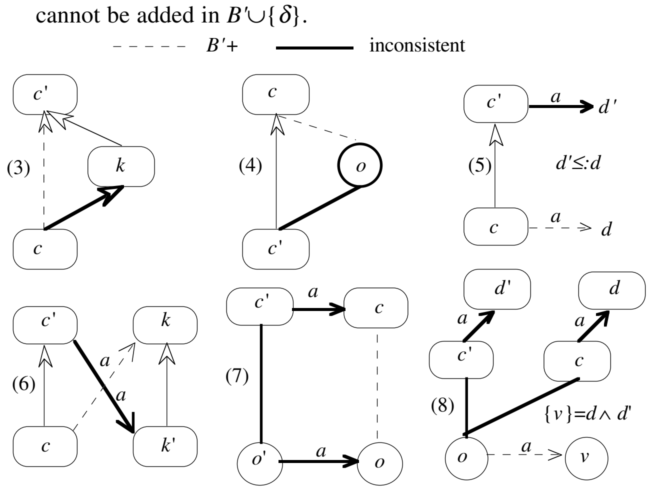

Figure 2. Examples of the different cases of the proposition 8.

Proposition 8 basically states that the notion of minimality can be interpreted on specialisation hierarchies as it is shown in figure 2. Its interpretation is that: if an assertion (3: specialisation relation, 4,7: class membership, 5,6: slot restriction, 8: slot valuation) is added (B'_+) in a minimal knowledge base, some other assertion (3: specialisation relation, 4: class membership, 5,6: slot restriction, 7: slot valuation, class membership or slot restriction, 8: class membership or slot restriction) which was necessary to its inference in the initial knowledge base (B) has to be discarded in order to avoid inconsistency (B'_-).

4. Working out revision

The final goal of this work is to build a practical revision system for an actual object-based system. The results provided so far (because they are always syntactically characterised on the normal form) allow to implement this revision.

Below, the context in which revision is to be applied is provided (§4.1). This context is important in the feasibility of the system. Some not self-explained implementation details are then provided (§4.2).

4.1 Framework

The revision operation is used in the context of a shared knowledge base. It aims at helping users to sort out the conflicts which can result from a modification they attempt. Revision is thus triggered when the users interact with the system.

Since there can exist many different minimal revised bases, the task of choosing the appropriate one is left to the user (while there are some general strategies such as not modifying terminological knowledge or — conversely — not modifying assertional knowledge). In order to be synthetic, the choices are presented to the user in a levelled way (following the abduction processed by the algorithm).

Thus, the revision protocol is the following:

1) the user attempts a modification;
2) the system detects an inconsistency, points it out to the user and proposes some assertions to withdraw (including the modification);
3) the user selects the assertion to withdraw; this might be sufficient or require more iterations in order to select from the various ways to withdraw the assertion.

Detecting an inconsistency related to a new assertion is very efficient (polynomial) due to the results of §2.2. However, the finding of all the revisions remains intractable in this system. Taking advantage of the user in order to choose between the various alternatives to explore enables to obtain any revised base in polynomial time (because only the abduction branch which leads to it is explored and testing consistency is efficient). However, if the users decide to explore several branches the revision process can be very long.

4.2 Implementation

The described system has been implemented on top of the Troeps object-based knowledge representation system [13]. It traps all the errors raised by Troeps when the system is operated and, if the error is revisable (e.g. it is not a spelling error), the revision system deals with the problem.

When inconsistency is detected, the assertions involved are precisely known. For each kind of inconsistency, we have listed the possible global strategies (e.g. move a class, move an instance under the most specific class possible). When a particular inconsistency arises, the adapted strategy is applied to the concerned assertions. The choices made by the user are ordered such that the first modifications made cannot be influenced by the following ones, since for example assertions on class hierarchy are modified before ones on instances. So that, the minimality of the global solution is generally ensured by the minimality of every steps. There are a few exceptions when a choice has an impact on previous identified modifications for which an a posteriori verification is needed, especially to verify if no assertion has been unnecessarily removed.

Interaction with the user is handled through HTTP (as is the rest of the system). At any moment the user can decide to abandon the current track and to come back to a previous choice.

This is implemented by stacking the alternatives (of course this can take polynomial space).

5. DISCUSSION

The apparent simplicity of the language proposed is a point deserving discussion. In fact, the language presented here covers a large part of the needs in object knowledge bases whose purpose is not to solve any possible problem but rather to organise knowledge and answer to some simple queries (e.g. [5]). In the examples, values are integers and domains are intervals, but this is a restriction which is not required by the work presented. The implementation takes advantage of an independent type system [2] that provides the required operations (emptiness test, intersection, generalisation). However, the expressiveness of the language could be enhanced in other ways (e.g. by adding constructors such as list or set). A more dramatic and required improvement would be to consider cyclic references among objects and classes.

Another important point is complexity. As a matter of fact, the complexity of the revision process exists but it is transferred to the users. The rationale behind this is that users are those who know the revision to be done and that it cannot be achieved automatically: the knowledge base does not contain the knowledge required for taking the decision [7]. In the tests we carried out, the compromise seems satisfactory. Although, the users could be overloaded with the number of solutions they have to consider. We have tried to overcome this problem through the design of the user interface and are currently working on an argumentative framework allowing several users to collaborate in the management of the revision.

6. RELATED WORKS

The revision obtained satisfies the basic AGM postulates (the first six ones) of [1] and a maximal consistent set of Hansson constraints [8]. It can be characterised as a maxi-choice operator [1].

The main related piece of work concerns revision in description logics [11,12]. However, the complexity of the decision problem in description logics did not allow the design of operational systems. The presented system is more alike feature algebras, however, we are not aware of any attempts to revise in this context.

Recently, Norman Foo [6] proposed methods to revise conceptual graph bases, others have considered using scripts for repairing errors in problem solving methods [9] and numerous other works have studied reorganisation of object databases (with the help of invariant properties and rules for restoring them). These studies give an idea on the methods for revising a base but do not explicitly consider the semantics of the knowledge representation formalism.

Theodoratos [14] studied the same problem as ours. His language is more powerful than the one used here in the expression of the links between classes, but slots are not considered. From a logical modelling of the class definition language, he defined two classical revision procedures, showed the co-NP-completeness of the general problem and several polynomial restrictions interesting for the language presented here.

7. CONCLUSION

We have presented a framework for interactive knowledge revision in object-based knowledge representation systems. It applies the results of automatic revision in logic to interactive revision in object knowledge representation systems. The work takes advantage of the structure of objects in order to constrain the space (the inconsistency does not spread, the source of inconsistencies can be syntactically tracked and the revisions are naturally ordered) and to interact with the users in an object style.

The complete treatment presented here has also been successfully applied to incoherent classes (classes whose interpretation is the empty set in all models) [3]. This has not been presented due to restricted space.

There are three lines of research that can be developed from this point:

- Extending the language capabilities (this covers cyclic references, type constructors — set, list, etc. — and perspectives and bridges — Troeps features [13]).
- Introducing the notion of preference in the implementation (obvious choices are protecting the « conceptual scheme » and protecting the data).
- Exporting that model to typed object-oriented programming languages (for providing the possible repairs to the compiling errors in such languages).

ACKNOWLEDGEMENTS

We thank Roland Ducournau for helpful comments on a previous version of the article.

REFERENCES

References [2, 3, 5, 13] are available at:
`http://www.inrialpes.fr/sherpa/publi.html`

[1] C. Alchourrón, E. Gärdenfors, P. Makinson. On the logic of theory change: partial meet contraction and revision functions. *Journal of symbolic logic*. 50(2):510-530, 1985

[2] C. Capponi. Type extensibility of a knowledge representation system with powersets. *Lecture notes in computer science* 1325:338-347, 1997

[3] I. Crampé. Révision interactive dans une base de connaissance à objets. Thèse d'informatique, université Joseph-Fourier, Grenoble (FR), 1997

[4] R. Ducournau. Les incertitudes de la classification incertaine. Actes 3es journées « langages et modèles à objets », Leysin (CH), pp183-200, 1996

[5] J. Euzenat, C. Chemla, B. Jacq. A knowledge base for *D. melanogaster* gene interactions involved in pattern formation. Proc. 5th ISMB, Halkidiki (GR), pp108-119, 1997

[6] N. Foo. Ontology revision. *Lecture notes in computer science* 954:16-31, 1995

[7] N. Friedman, J. Halpern. Belief revision: a critique. Proc. 5th KR conference, Cambridge (MA US), pp421-431, 1996

[8] S. Hansson. A test battery for rational database updating. *Artificial intelligence* 82:341-352, 1996

[9] Y. Gil, M. Tallis, A script-based approach to modifying knowledge bases, Proc. 14th AAAI, Providence (RI US), pp377-384, 1997

[10] M. Kifer, G. Lausen, J. Wu. Logical foundations of object-oriented and frame-based languages. *Journal of the ACM* 42(4):741-843, 1995

[11] B. Nebel. Reasoning and revision in hybrid representation systems. *Lecture notes in computer science* 422, 1990

[12] B. Nebel. Syntax-based approaches to belief revision. in P. Gärdenfors (ed.). Belief revision. Cambridge tracts in theoretical computer science 29. Cambridge university press, Cambridge (UK), pp52-88, 1992

[13] Projet Sherpa. Tropes 1.0 reference manual. Internal report, INRIA Rhône-Alpes, Grenoble (FR), June 1995 (rev. Troeps 1.2 reference manual, May 1998, 142p.), 85p.

[14] D. Theodoratos. Updating object-oriented schema structures viewed as logical theories. Technical report 12, ERCIM, Rocquencourt (FR), 1995