Revision of Rule Bases

Fatma Bouali, Stéphane Loiseau, Marie-Christine Rousset

L.R.I - U.R.A C.N.R.S, University of Paris-Sud
Building 490
91405 Orsay Cedex
email: {bouali, loiseau, mcr}@lri.fr

ABSTRACT: Building a correct knowledge base (KB) is an incremental and difficult process, made of successive verification and revision steps. *Verification* steps aim at ensuring that a KB is correct. *Revision* steps aim at modifying the knowledge base in order to eliminate the anomalies that the verification steps have pointed out. In this paper, we propose a method for the revision of an incorrect rule base, based on characterizing the possible causes of the detected inconsistencies. The possible actions of revision that are considered are adding integrity constraints, deleting rules, and more interestingly refining existing rules by adding new conjuncts

KEYWORDS: inconsistency, conflict, debugging, refinement, reparation

1 Introduction

Many works deal with verification of a Rule Base (RB) [?]. Several of them focus on consistency checking [?, ?, ?, ?]. A RB is said consistent if for any valid input, it does not enable entailing a contradictory output. In this paper, we consider the definition of rule base consistency given in [?] and which improves the previous ones [?, ?, ?]. This definition considers a set of test cases (given by a reliable source) in addition to the set of usually considered integrity constraints to describe valid inputs and contradictory outputs. A given test case is composed of an input facts set and an output fact such that this output fact is known to logically hold from the input fact set in the RB system domain.

Checking the consistency of a rule base is not an end in itself. At the border between verification and revision is the *characterization* of rule base inconsistencies. Such characterization consists in a set of proofs of the RB inconsistencies. Those proofs are called *conflicts* and are the basic information to the revision process. The first contribution of this paper is to present how to extend a conflict definition [?, ?, ?] to the setting of our RB consistency definition [?]. A conflict can be seen as a minimal set of rules, eventually associated to an input fact set, that is a sufficient condition to prove an inconsistency.

Once a rule base has been characterized inconsistent, it has to be revised in order to correct it (i.e, to eliminate the conflicts). The aim is to find a *revision plan* to correct the rule base while preserving as much as possible of the previous modeling work. We consider that a revision plan is composed of the following elementary revision *actions*: deleting a rule, modifying a rule, adding an integrity constraint. In particular, deleting an existing rule may restore the consistency of the rule base from a logical point of view but it is not necessarily a desirable action of revision because it is a loss of expertise. Modifying a rule by adding conjuncts to its antecedent is often a more appropriate action: it can restore the logical consistency of the rule base in a less brutal way than deleting this rule. Adding an integrity constraint is also a possible action to restore the consistency. Most of previous works deal only with adding of integrity constraints or deleting rules actions[?, ?]. The major contribution of this paper is to propose a solution to restore the RB consistency which also deals with rules modifications and preserves as much as possible the expertise. First, this solution considers the adding of a cunjunct to a rule if it can prevent the rule to be used to proove an inconsistency (such an action eliminates a conflict). Second, any revision plan must preserve the true positive test cases [1].

Our aim was not to design a tool for automatically revising a rule base. We consider that the revision plan to perform has eventually to be decided by the expert. The third contribution of this paper is to present a heuristic to sort those plans such that the best one is presented first to the expert.

The paper is organized as follows. In section 2, we introduce our working hypotheses, we recall our rule base consistency definition [?] and we present the way to characterize the inconsistencies of a rule base. In section 3 , we describe the revision actions we deal with to restore a rule base consistency. We define revision plans w.r.t a rule base which restore the consistency of a rule base and preserve the true positive test cases. We present an algorithm which computes such revision plans. Finally, we present an heuristic to sort the different revision plans.

2 Rule Base Consistency

2.1 Working Hypotheses

We consider knowledge bases having an *attribute operator value* syntax. A rule base $\mathcal{R}$ is a set of rules of the form: $l_1 \wedge \ldots \wedge l_n \Rightarrow l$ where the literals l_i are triples of the form $A_i\ op_i\ v_i$, such that A_i is the name of an attribute, v_i is a value and op_i is one operator among $\leq$, $<$, $\geq$, $>$, $=$ or $\neq$. A literal l appearing as a consequent of a rule can be either of the form $B\ op\ v$, or the proposition $\perp$. The rules having $\perp$ as consequent are *integrity constraints* expressing incompatibilities between literals. We distinguish the subset $\mathcal{R}_{sure}$ of rules which have been stated as reliable and cannot be modified. In the example, the names of the rules that belong to $\mathcal{R}_{sure}$ begin with a capital letter.

A fact base is a set of literals. An *input* fact base contains only literals involving input attributes. The *input attributes* are those attributes that do not appear in the consequents of the rules [2].

The semantics of the rules and of the facts is the standard model-based logical semantics. The attributes can be seen as functions of arity 0, the values can be seen as constants. An *interpretation* I contains a non-empty domain $\mathcal{O}^I$ in which the operators $\leq$, $<$, $\geq$, $>$, $=$ or $\neq$ are well defined. It assigns

[1] Intuitively, a test case is said *true positive* if its output fact is logically satisfied from its input fact set and RB

[2] We consider that those attributes are determined once for all and that they do not change over the possible modifications of the rule base

an element $A^I \in \mathcal{O}^I$ to every attribute A, and an element $v^I \in \mathcal{O}^I$ to every value v. An interpretation I is a model of a literal $A\ op\ v$ iff $A^I\ op\ v^I$ is true in $\mathcal{O}^I$. An interpretation I is a model of r if, whenever every literal of the antecedent of r is true, then its consequent is true. Finally, an interpretation is a model of $\mathcal{R}$ if it is a model of every rule. Given a rule base $\mathcal{R}$ and an input fact base $\mathcal{F}$, the corresponding output is the set of facts that can be entailed from $\mathcal{R} \cup \mathcal{F}$.

A test case is a pair $< \mathcal{F}_{test},\ O >$ where $\mathcal{F}_{test}$ is an input fact base, and O is a non input fact (= output). Each test case $< \mathcal{F}_{test},\ O >$ represents a logical association, which is known to hold on the domain, between an input fact base ($\mathcal{F}_{test}$) and a distinguished output fact (O). Every test case $< \{f_1, \dots, f_n\},\ O >$ can then be expressed as a rule $f_1 \wedge \dots \wedge f_n \Rightarrow O$, which represents a positive constraint which should be satisfied. Given a set of test cases, $\mathcal{R}_{test}$ is the set of rules which corresponds to it. A test case which is satisfied by the KB is said true positive.

Definition 1: *A test case $< \mathcal{F}_{test},\ O >$ is true positive w.r.t a rule base $\mathcal{R}$ if the expected output literal $O = A\ op\ v$ is more specific than the one that can be reached from $\mathcal{R}$ and the input $\mathcal{F}_{test}$, i.e: $\mathcal{R} \cup \mathcal{F}_{test} \models A\ op'\ v'$ and $A\ op\ v \models A\ op'\ v'$.*

Example 1: We use the following illustrative example throughout the paper. Consider a domain where we want to determine time that people can spend as voluntary helpers (resp. at sport). The attributes $WageEarner$, $Manager$, $Priest$, $HouseWife$, are input attributes which distinguish different kinds of people. The attributes $WorkHours$ and $ChildrenNumber$ are numerical input attributes which denote the number of hours spent at work and the number of children. The boolean attribute $FreeTime$ describes whether people have free time or not. The attributes $HelpHours$ (resp. $SportHours$) denotes the number of hours spent as a voluntaree helper (resp. at sport) . The following rules $\mathcal{R}$ describe our domain.

$r_1 : WageEarner = true \wedge Manager = true \Rightarrow FreeTime = false$
$r_2 : FreeTime = false \Rightarrow HelpHours \leq 5$
$r_3 : Priest = true \Rightarrow HelpHours \geq 15$
$r_4 : HouseWife = true \Rightarrow HelpHours \geq 8$
$r_5 : FreeTime = false \Rightarrow SportHours \leq 2$
$R_6 : HouseWife = true \wedge WageEarner = true \Rightarrow \perp$

$\mathcal{R}_{sure}$ contains the single constraint R_6 stating that a house wife cannot be a wage-earner. Consider that we are given the following test cases $\mathcal{T}$.

$T_1 :< \{WageEarner = true, Manager = true, WorkHours \geq 40\}, HelpHours = 4 >$
$T_2 :< \{HouseWife = true, ChildrenNumber = 3\}, HelpHours = 3 >$
$T_3 :< \{Priest = true\}, HelpHours = 20 >$
$T_4 :< \{WageEarner = true, Manager = true, WorkHours \leq 20\}, SportHours = 4 >$
$T_5 :< \{WageEarner = true, Manager = true, WorkHours \geq 40\}, SportHours = 1 >$
T_1 and T_3 and T_5 are true positive w.r.t $\mathcal{R}$ while T_2 and T_4 are not.

2.2 Rule Base Consistency Definition

Intuitively, a rule base is consistent if it does not entail any contradiction from any valid input fact base. In our approach the validity of a fact base is characterized by its non contradiction w.r.t some integrity constraints that are known to hold in the domain. Some of those constraints are explicit. Some other constraints can be infered from the explicit ones and the subset of rules which can be considered as representing a reliable knowledge, i.e $\mathcal{R}_{sure}$ together with $\mathcal{R}_{test}$.

Definition 2: *Let $\mathcal{R}_{sure}$ be the subset of the rule base composed of rules that are stated as reliable, and let $\mathcal{R}_{test}$ be the set of rules corresponding to a set of test cases T. An input fact base $\mathcal{F}$ is contradictory if $\mathcal{R}_{sure} \cup \mathcal{R}_{test} \cup \mathcal{F} \models \perp$*

In order to determine the set of minimal contradictory fact bases for a non recursive rule base $\mathcal{R}$, an extended ATMS-like [?] algorithm can be used [?].

Our rule base consistency definition is in the same line as previous ones [?, ?, ?, ?] but it extends them in two ways. First, it relies on the more constrained notion of contradictory input fact bases. Second, it takes into account a set of test cases. A rule base is then consistent if it does not enable to entail a contradiction from any non contradictory input fact base and from any test case.

Definition 3: *Let $\mathcal{R}$ be a rule base, $\mathcal{T}$ a set of test cases $< \mathcal{F}_{test}, O >$. $\mathcal{R}$ is consistent if*
- *For every $< \mathcal{F}_{test}, O > \in \mathcal{T}$, $\mathcal{R} \cup \mathcal{F}_{test} \cup O \not\models \bot$*
- *For every* non contradictory *input fact base $\mathcal{F}$, $\mathcal{R} \cup \mathcal{F} \not\models \bot$*

2.3 Characterization of rule base inconsistencies

For revision purposes, it is important to be able to characterize the possible causes of the rule base inconsistency. Those possible causes are either subsets of non sure rules ($\notin \mathcal{R}_{sure}$) which are responsible for the entailment of $\bot$ from an input test case, or subsets of rules together with sets of non contradictory input facts from which $\bot$ can be entailed. They respectively correspond to the notions of *strong conflicts* and *weak conflicts.*

Definition 4: *Let $\mathcal{R}$ be a rule base , including a subset $\mathcal{R}_{sure}$ composed of rules having $\bot$ as consequent (and possibly other reliable rules). Let $\mathcal{T}$ be a set of test cases, composed of pairs $< \mathcal{F}_{test}, O >$.*
- *A strong conflict is a subset $\mathcal{R}_{s_{conflict}}$ of non sure rules such that there exists $< \mathcal{F}_{test}, O > \in \mathcal{T}$ s.t $\mathcal{R}_{s_{conflict}} \cup \mathcal{R}_{sure} \cup \mathcal{F}_{test} \cup O \models \bot$. $\mathcal{R}_{s_{conflict}}$ is then said revealed by $< \mathcal{F}_{test}, O >$*
- *A weak conflict is a pair $< \mathcal{F}, \mathcal{R}_{w_{conflict}} >$, where $\mathcal{R}_{w_{conflict}}$ is a subset of non sure rules such that $\mathcal{F}$ is a non contradictory input fact base such that $\mathcal{R}_{w_{conflict}} \cup \mathcal{R}_{sure} \cup \mathcal{F} \models \bot$.*

Theorem 1: *A rule base is consistent iff there is no (weak or strong) conflict.*

Detecting conflicts is an informative basis for the revision process. A strong conflict $\mathcal{R}_{s_{conflict}}$ reveals a real anomaly in the rule base because it is a proof that a contradiction is entailed from a valid input ($\mathcal{F}_{test}$): some of the rules of $\mathcal{R}_{s_{conflict}}$ have to be modified or deleted. A weak conflict $< \mathcal{F}, \mathcal{R}_{w_{conflict}} >$ reveals a potential anomaly of the rule base. It might be the case that $\mathcal{F}$, which is the input fact base causing the rules to infer $\bot$ is not a valid input. In that case, the rules in $\mathcal{R}_{w_{conflict}}$ might be correct. Adding an input constraint, making $\mathcal{F}$ contradictory, would eliminate the conflict. Otherwise, there is a real anomaly, some of the rules of $\mathcal{R}_{w_{conflict}}$ have to be modified or deleted.

We compute *minimal* conflicts which represent minimal proofs of inconsistencies. A strong conflict $\mathcal{R}_{s_{conflict}}$ is minimal if and only if it does not exist a proper subset of it which is a strong conflict. A weak conflict $< \mathcal{F}, \mathcal{R}_{w_{conflict}} >$ is minimal if and only if it does not exist another weak conflict $< \mathcal{F}', \mathcal{R}'_{w_{conflict}} >$ such that $\mathcal{R}'_{w_{conflict}} \subseteq \mathcal{R}_{w_{conflict}}$ and $\mathcal{F} \models \mathcal{F}'$.

Determining the minimal strong conflicts can be done by applying a standard inference procedure on the rules together with each $\mathcal{F}_{test}$, while keeping the trace of the non sure rules involved in the deduction of $\bot$.

Determining the minimal weak conflicts can be done by an extended ATMS algorithm which computes the label of $\bot$, and by keeping the trace of the non sure rules involved in the computation of the label.

Example 2: For our example, there is one minimal weak conflict:
$C_1 =< \{r_1, r_2, r_3\}, \{WageEarner = true, Manager = true, Priest = true >$.
From the literals $WageEarner = true$ and $Manager = true$, and the rules r_1 and r_2, $HelpHours \leq 5$ can be entailed. ¿From $Priest = true$ and r_3, it can be entailed $HelpHours \geq 15$, which is unsatisfiable together with $HelpHours \leq 5$.
There are two minimal strong conflicts: $C_2 = \{r_4\}$ revealed by T_2. $C_3 = \{r_1, r_5\}$ revealed by T_4.

3 Revision of an Inconsistent Rule Base

Once a rule base has been characterized inconsistent, it has to be revised in order to correct it (i.e, to eliminate the conflicts). The aim is to correct the rule base while preserving as much as possible of the previous modeling work. After presenting the revision actions (3.1) we deal with, we show how to compute revision plans composed of such actions in two steps. The first step reviews each minimal conflict and determines a set of revision plans where each revision plan enables to eliminate it (3.2). The second step composes the set of revision plans obtained at the first step in order to determine a set of revision plans such that each of them enables to eliminate all the conflicts (3.3). Finally, we present (3.4) a solution to sort these revision plans.

3.1 Revision Actions and Revision Plans

We consider three types of revision actions. First, the *modification* of a rule r, by adding a conjunct p to its antecedent: it will be denoted $mod(r,p)$. Second, the *deletion* of a rule r: it will be denoted $del(r)$. Third, the *adding* of an integrity constraint: it will be denoted $add_{IC}(p_1,\ldots,p_n)$ where $p_1,\ldots,p_n$ denote the conjuncts composing the body of the constraint.

A set of revision actions will be called a *revision plan*. Given a revision plan $\mathcal{P}$, we will denote $[\mathcal{P}](\mathcal{R})$ the set of rules obtained from $\mathcal{R}$ by applying the actions of $\mathcal{P}$.

Given a conflict C, there are different possible revision plans $\mathcal{P}_i$ which can eliminate it (i.e, C is not a conflict for $[\mathcal{P}_i](\mathcal{R})$). Our approach consists in computing minimal plans that eliminate the existing conflicts while guaranteeing that they do not add new conflicts and that they are conservative w.r.t the set of given test cases.

Definition 5: *Let $\mathcal{R}$ be a rule base, $\mathcal{T}$ a set of test cases and $\mathcal{P}$ a revision plan. $\mathcal{P}$ is conservative w.r.t $\mathcal{R}$ and $\mathcal{T}$ if every test case in $\mathcal{T}$, which is true positive for $\mathcal{R}$, is true positive for $[\mathcal{P}](\mathcal{R})$.*

In order to guarantee that the plans that we compute are conservative and do not add any additional conflicts, we have to choose carefully the rules to which our revision actions apply. For this purpose, the non sure rules are categorized in three types : *removable* rules are those which are not necessary for satisfying any true positive test case, *modifiable* rules are those whose the modification (i.e, adding a conjunct to their antecedent) maintains all the true positive test cases. The other rules are said *unchangeable.*

Definition 6: *Let $\mathcal{R}$ be a rule base and $\mathcal{T}$ a set of test cases.*
- *A non sure rule r is removable w.r.t a true positive test case T iff T remains true positive in $[\{del(r)\}](\mathcal{R})$.*
A non sure rule r is removable w.r.t $\mathcal{T}$ if it is removable w.r.t every true positive test case of $\mathcal{T}$.
- *A non sure rule r is modifiable by p w.r.t a true positive T if r is not removable w.r.t T and T is true positive for $[\{mod(r,p)\}](\mathcal{R})$.*
A non sure rule r is modifiable by p w.r.t $\mathcal{T}$ if r is not removable w.r.t $\mathcal{T}$ and r is modifiable by p or removable w.r.t every true positive test case of $\mathcal{T}$.
- *A non sure rule r is unchangeable w.r.t $\mathcal{T}$ otherwise.*

Example 3: In our example, r_2 (resp. r_5) is modifiable by $WorkHours \geq 40$ w.r.t our set of test cases: it only interfers with the test case T_1 (resp.T_5) ; adding in its antecedents the conjunct $WorkHours \geq 40$ which appear in the input fact base of T_1 (resp.T_5) is a revision action which guarantees that T_1 (resp.T_5) remains true positive. r_1 is also modifiable by $WorkHours \geq 40$ w.r.t our set of test cases: it interfer with the test cases T_1 and T_5 ; adding in its antecedents the conjunct $WorkHours \geq 40$ which appear in the input fact base of T_1 and T_5 is a revision action which guarantees that T_1 and T_5 remain true positive. Rule r_3 is unchangeable w.r.t $\mathcal{T}$ because removing it or adding a conjunct to its antecedent would make T_3 not true positive anymore. Rule r_4 is removable: it only interfers with test case T_2, which is not true positive anyway.

3.2 Revision Plans w.r.t a conflict

Performing a revision plan $\mathcal{P}$ w.r.t to a minimal conflict C on $\mathcal{R}$ should eliminate C and should not introduce new conflicts.

Definition 7: *Let $\mathcal{R}$ be a rule base,$\mathcal{T}$ a set of test cases, and C a minimal conflict of $\mathcal{R}$. P is a revision plan w.r.t C iff P is a minimal set of actions such that:*
(1) P eliminates C: C is not a conflict of $[P](\mathcal{R})$
(2) P is conservative w.r.t $\mathcal{R}$ and $\mathcal{T}$
(3) P does not add any new minimal conflict: let $\mathcal{CS}$ be the set of minimal conflicts of $\mathcal{R}$ and $\mathcal{T}$, $\mathcal{CS}_P$ the set of minimal conflicts of $[P](\mathcal{R})$ and $\mathcal{T}$: $\mathcal{CS}_P \subset \mathcal{CS}$

Given a minimal conflict, a set of rules $\mathcal{R}$ and a set of test cases $\mathcal{T}$, the algorithm **revise**$(C,\mathcal{R},\mathcal{T})$ (cf. Appendix **??**) computes a set of revision plans that eliminate it. We restrict the literals that can be involved in a modification of rules to belong to the test cases. With respect to this restriction, we show that the result of the algorithm **revise**$(C,\mathcal{R},\mathcal{T})$ is the set of revision plans w.r.t C. Theorem **??** gives the core theoretic result, which guarantees that our method restores the global consistency in a conservative and minimal way.

The most subtle and interesting point in our revision algorithm is the last case which deals with modification of rules involved in a weak conflict. The idea is to add to the antecedents of those rules new conjuncts which, altogether and together with the input facts $\mathcal{F}$ revealing the weak conflict, compose a contradictory input fact base. This point is illustrated in example **??** which shows the result of the algorithm $revise(C_1, \mathcal{R}, \mathcal{T})$, when applied to the weak conflict C_1 (which is the single weak conflict that was obtained previously).

Example 4:
$C_1 =< \{r_1, r_2, r_3\}, \{WageEarner = true, Manager = true, Priest = true\} >$. We have already seen that r_1 and r_2 were modifiable (r_3 is unchangeable). The set $\mathcal{L}_m$ of input literals appearing in the test cases for which the rules of the conflict are modifiable is $\{WorkHours \geq 40\}$. The only subset of $\mathcal{L}_m$ leading to a contradictory fact base with {WageEarner=true, Manager=true, Priest=true} is $\mathcal{L}_m$ itself. Therefore, the revision plans corresponding to modifications of rules that eliminate the conflict C_1 are: $\{mod(r1, WorkHours \geq 40)\}$; $\{mod(r2, WorkHours \geq 40)\}$

The algorithm provides another plan corresponding to adding an input constraint:
$\{add_{IC}(WageEarner = true, Manager = true, Priest = true)\}$.
The single revision plan w.r.t the strong conflict $C_2 = \{r_4\}$ is: $\{(del(r_4)\}$
The possible revision plans w.r.t the strong conflict $C_3 = \{r_1, r_5\}$ are:

$\{mod(r_1, WorkHours \geq 40)\}$
$\{mod(r_5, WorkHours \geq 40)\}$

Theorem 2: *Let $\mathcal{R}$ be a set of rules, $\mathcal{T}$ a set of test cases, and C a minimal (strong or weak) conflict. Let $PossiblePlans$ be the set of revision plans returned by the algorithm $revise(C, \mathcal{R}, \mathcal{T})$. Considering only modifications of rules by literals belonging to test cases, $PossiblePlans$ is the set of revision plans w.r.t C.*

3.3 Revision plans w.r.t a rule base

A revision plan P w.r.t a rule base is a set of actions that enable to restore the rule base consistency in a minimal and conservative way. Such a revision plan should be minimal in the sense that each proper subset of P should not restore the consistency.

Definition 8: *Let $\mathcal{R}$ be a rule base, $\mathcal{T}$ a set of test cases and P a revision plan. P is a revision plan w.r.t $\mathcal{R}$ iff:*

1. *P is conservative w.r.t. $\mathcal{T}$*
2. *$[P](\mathcal{R})$ is consistent*
3. *P is a minimal set verifying both (1) and (2).*

A solution which copes with all the minimal conflicts of a rule base $\mathcal{R}$ and hence restores the rule base consistency consists in performing on $\mathcal{R}$ a revision plan P such that for each minimal conflict C of $\mathcal{R}$, P includes a revision plan w.r.t C.

Theorem 3: *Let $\mathcal{R}$ be a rule base, $\mathcal{T}$ a set of test cases, $\mathcal{C} = \{C_1 \dots C_n\}$ the set of minimal conflicts of $\mathcal{R}$ and $\mathcal{P}_{C_i}$ the set of revision plan w.r.t C_i $(i = 1 \dots n)$. The sets $\mathcal{H} = \bigcup_{j=1}^{m} H_j$ where $H_j \in \bigcup_{i=1}^{n} \mathcal{P}_{C_i}$ verifying the following properties:*

1. *$\forall C_i \in \mathcal{C}, \exists H_k \in \{H_1 \dots H_m\} / H_k$ includes a revision plan w.r.t C_i*
2. *$\forall \mathcal{H}' \subset \mathcal{H}$, (1) is not verified*

are revision plans w.r.t $\mathcal{R}$[3]

The algorithm which computes the sets of repair actions of a rule base (cf. Appendix **??**) is an extension of Reiter's algorihm [?] which computes the minimal hitting sets of a collection of sets $\mathcal{S}$. A minimal hitting set of $\mathcal{S}$ has a common element with each set of $\mathcal{S}$. In our case, a revision plan $\mathcal{H}$ includes an element H (which is a set) of each $\mathcal{P}_{C_i}$.

[3] The property of conservatism of $\mathcal{H}$ w.r.t $\mathcal{R}$ and $\mathcal{T}$ must be also verified if $\mathcal{R}$ presents some redundancies

Example 5:

$\mathcal{P}_{C_1} = \{\{mod(r_1, WorkHours \geq 40)\}, \{mod(r_2, WorkHours \geq 40)\},$
$\{add_{IC}(WageEarner = true, Manager = true, Priest = true)\}\}$
$\mathcal{P}_{C_2} = \{\{del(r_4)\}\}$
$\mathcal{P}_{C_3} = \{\{mod(r_1, WorkHours \geq 40)\}, \{mod(r_5, WorkHours \geq 40)\}\}$
Therefore, the revision plans w.r.t the rule base $\mathcal{R}$ corresponding to extended hitting sets of $\{\mathcal{P}_{C_1}, \mathcal{P}_{C_2}, \mathcal{P}_{C_3}\}$ are:
$\{mod(r_1, WorkHours \geq 40), del(r_4)\}$
$\{mod(r_2, WorkHours \geq 40), mod(r_5, WorkHours \geq 40), del(r_4)\}$
$\{\{add_{IC}(WageEarner = true, Manager = true, Priest = true)\}, mod(r_5, WorkHours \geq 40), del(r_4)\}$

3.4 Sorting the revision plans w.r.t a rule base

In this section, we introduce an heuristic criterion which will be used to partially order the previously computed revision plans w.r.t a rule base. We consider that rules modifications should be privileged and the highest number of test cases that suggest such modification is a measure of the quality of a revision plan. For each rule modification action $A = mod(p, r)$, is associated a coefficient which represents the number of test cases that suggest to perform such an action: $n_A = cardinal(\{T \in \mathcal{T}/r$ is modifiable by p w.r.t $T\})$. The other actions have the coefficient 0. The coefficient n_P associated to a revision plan $\mathcal{P}$ is obtained by summing the coefficients of the actions it contains and dividing this sum by its cardinal (i.e, $n_P = \frac{\sum_{A \in P} n_A}{cardinal(P)}$). The revision plan with a highest coefficient is the best one and it will be supplied first to the rule base designer.

Example 6: The coefficients of the differents adding actions are:
$mod(r_1, WorkHours \geq 40) : 2$ (the test cases T_1 and T_5 suggest such modification)
$mod(r_2, WorkHours \geq 40) : 1$
$mod(r_5, WorkHours \geq 40) : 1$
Hence the coefficients of the previous revision plans are:

$\{mod(r_1, WorkHours \geq 40), del(r_4)\}$: 1
$\{mod(r_2, WorkHours \geq 40), mod(r_5, WorkHours \geq 40), del(r_4)\} : 0.66$
$\{\{add_{IC}(WageEarner = true, Manager = true, Priest = true)\},$
$mod(r_5, WorkHours \geq 40), del(r_4)\} : 0.33$

4 Conclusion

We described a new approach which takes into account a set of test cases to characterize revision plans for rule bases. A revision plan enables to recover the rule base consistency and is conservative in the sense that each true positif test case remains true positif after performing this revision plan on the rule base. We proposed algorithms for generating revision plans. It is novel for a revision system to generate revision plans containing modifications of rules that cope with not only strong conflicts but also with weak conflicts. This issue has not been addressed so far.

Our algorithms have been implemented in the KBDR system. Though the problems that are addressed are inherently complex [**?**, **?**] our method has been provably feasible on medium size knowledge bases. In particular, the KBDR system has been used to verify and revise a real medical rule base designed by the French marine army (160 rules, with a depth of 10). Two conflicts were found and four revision plans have been computed and ordered. The session took 30 seconds on a workstation and the rule base easily revised. The main source of inefficiency of our method is the number of conflicts, which can lead to an explosion of revision plans. For large scale problems, it will undoubtely be necessary either to employ heuristic strategies in order to filter revision plans, or to restrict the use of our complete method to revision steps that are final in the verification/revision process.

The main point that distinguishes our approach w.r.t most of the existing works on KB revision [**?**, **?**, **?**, **?**] is that they fit in with a testing approach only. They just consider strong conflicts. Most of them are inspired by a learning approach and aim at covering the most possible test cases. In such approaches, it is not guaranteed that the revision actions preserve the global consistency of the KB. The preliminary work reported in [?] is similar in spirit to our approach. To eliminate a conflict, a solution proposed is to add the negation of a literal to a rule of a conflict when another rule in this conflict has this literal in its condition part.

Our current research aims first to extend our algorithms to non recursive Horn rule bases, and second to merge our approach with learning works in order to maximise the number of true positive test cases.

A Computing revision plans for a conflict

procedure revise$(C, \mathcal{R}, \mathcal{T})$
/* C is a (weak or strong) conflict, $\mathcal{R}$ is a set of rules, $\mathcal{T}$ is a set of test cases */
/* The procedure computes a set of revision plans in order to eliminate C in a conservative way */
Let $\mathcal{L}$ be the set of input literals appearing in the test cases of $\mathcal{T}$
Possible-Plans := $\emptyset$
Let $\mathcal{R}_{conflict}$ be the set of rules involved in the conflict C
for every removable $r \in \mathcal{R}_{conflict}$ **let** $P = \{\{del(r)\}\}$
Possible-Plans := Possible-Plans $\cup$ P
/* deleting a removable rule is a revision action which is appropriate
for deleting a strong or weak conflict */
if C is a strong conflict $\mathcal{R}_{conflict}$:
for every rule $r \in \mathcal{R}_{conflict}$ and every input literal $p \in \mathcal{L}$
such that r is *modifiable* by p:
if for every test case $< \mathcal{F}_{test}, \mathcal{O} >$ revealing C $\mathcal{F}_{test} \not\models p$, **let** $P = \{\{mod(r,p)\}\}$
Possible-Plans := Possible-Plans $\cup$ P
/* disabling one rule of a strong conflict to be triggered from
the test cases revealing it is a revision action which removes it */
if C is a weak conflict $< \mathcal{F}, \mathcal{R}_{conflict} >$:
if $\forall < \mathcal{F}_{test}, O > \in \mathcal{T}$, $\mathcal{F}_{test} \not\models \mathcal{F}$, **let** $P = \{\{add_{IC}(\mathcal{F})\}\}$
Possible-Plans := Possible-Plans $\cup$ P
/* adding a constraint stating that the fact base associated with a weak conflict
is contradictory is a revision action which removes it */
let $\mathcal{L}_m = \{p \in \mathcal{L} \mid \exists r \in \mathcal{R}_{conflict}$ s.t r is modifiable by $p\}$
for every subset $\mathcal{S} = \{p_1, \ldots, p_n\}$ of $\mathcal{L}_m$ such that
$\mathcal{S} \cup \mathcal{F}$ is a minimal contradictory input fact base:
for every $p_i \in \mathcal{S}$, **let** $P_i = \{mod(r, p_i) \mid r \in \mathcal{R}_{conflict}$, modifiable by $p_i\}$
let $P = P_1 \times \ldots \times P_n$
/* P is the cartesian product of $P_1, \ldots, P_n$ */
Possible-Plans := Possible-Plans $\cup$ P
return Possible-Plans

B Computing revision plans for a set of conflicts

procedure hitting-set($\mathcal{P}$)
/* $\mathcal{P} = \{\mathcal{P}_{C_1} \dots \mathcal{P}_{C_n}\}$ is a collection of sets of revision plans ordered w.r.t "$\prec$"
$\mathcal{P}_{C_i} \prec \mathcal{P}_{C_j}$ iff $\exists \{add_{IC}(\bigwedge_{F \in \mathcal{F}_i} F \Rightarrow \bot)\} \in \mathcal{P}_{C_i}$ and $\exists \{add_{IC}(\bigwedge_{F \in \mathcal{F}_j} F \Rightarrow \bot)\} \in \mathcal{P}_{C_j}$
such that $\mathcal{F}_j \models \mathcal{F}_i$ (notice that $\mathcal{P}_{C_i}$ contains atmost one add_{IC} action)
The procedure constructs an edge labeled and node labeled extendend HS-tree $Tree$
and returns a set of revision plans (special leaves of $Tree$)
in order to eliminate all the conflicts in a conservative way. */
Possible-Plans := $\emptyset$
Label the root of $Tree$ by the first element of $\mathcal{P}$
for each node N_i of $Tree$ labeled by an element $\mathcal{P}_{C_l}$ of $\mathcal{P}$ (i.e. a set of revision plan)
  for each element P of $\mathcal{P}_{C_l}$ (i.e. a revision plan)
    create a node N_j
    create an edge labeled by P joining N_i to N_j
    let $H_{IC}(N_j)$ the union of the labels of the edges on the path from the root
    to the node N_j
    which are constitued only of integrity constraints adding actions
    let $H(N_j)$ the union of the other labels of the edges on the path from the root
    to the node N_j
    if $\exists \mathcal{P}_{C_k} \in \mathcal{P} / \mathcal{P}_{C_k} \cap H(N_j) = \emptyset$ and $(\forall \{add_{IC}(p_1 \dots p_n)\} \in \mathcal{P}_{C_k},$
    $\nexists add_{IC}(p'_1 \dots p'_m) \in H_{IC}(N_j) / \bigwedge p_1 \dots p_n \models \bigwedge p'_1 \dots p'_m)$
      label the node N_j with $\mathcal{P}_{C_k}$
    else
      if $\exists N_k$ with *'V'* label $\in Tree /\ H(N_k) \subseteq H(N_j)$ and $\forall add_{IC}(p_1 \dots p_n) \in H_{IC}(N_j)$
      $\exists add_{IC}(p'_1 \dots p'_m) \in H_{IC}(N_k) /\ \bigwedge p_1 \dots p_n \models \bigwedge p'_1 \dots p'_m$
        close the node N_j /* it can not lead to a minimal revision plan*/
      else
        label the node N_j with *'V'* /* N_j is a special leave * /
        PossiblePlans := PossiblePlan $\cup \{H(Nj) \cup H_{IC}(N_j)\}$
return Possible-Plans