Interactive refinement of a knowledge base

R. Djelouah[1, 2], B. Duval[1], S. Loiseau[1]

[1]LERIA, Université d'Angers, 2 bd Lavoisier, 49045 ANGERS Cedex 01 France
{djelouah, bd, loiseau}@info.univ-angers.fr
[2]ISAIP-ESAIP, 18 rue du 8 Mai 45, 49180 St Barthélémy France

Abstract

This paper presents a new method of interactive refinement of a knowledge base. The first step of our method is a validation stage which checks the consistency and the completeness of the knowledge base. Then, the second step is a refinement stage which aims to eliminate the anomalies (inconsistencies or incompleteness) that have been detected. The refinement stage is an iterative process performed in close interaction with the expert; the main step of this process consists of submitting to the expert a rule that must be repaired with a list of possible modifications for this rule. Our originality is that the expert has to take decisions about relevant modifications of a rule, which is an easy task for him because the rule represents an element of knowledge that he understands well. After each decision of the expert, our method updates the set of anomalies in order to apply criteria to select the following rule that needs to be repaired.

I. Introduction

A lot of research has been devoted to the subject of quality in the development of knowledge base systems (Preece 1997). A first type of work (Grogono et al. 1992) (O'Keefe and O'Leary 1993) (Preece 97) deals with the definition of criteria that can measure the quality of a knowledge base. For this purpose, two criteria have often been studied: consistency and completeness (Rousset 1988)(Bouali and Loiseau 1995) (Ayel and Bendou, 1996). Studying consistency means checking whether the knowledge base (KB) infers contradictory results and this test leads to the computation of sets of facts and knowledge that can produce a contradiction. Studying completeness means checking whether the KB covers the whole domain of study, and so finding an incompleteness consists of finding an example which is not covered by the base. Several formal characterisations of inconsistency and incompleteness have been proposed (Bouali and Loiseau 1998) (Buekenoogen et al. 2000).

A second type of works concerns the refinement of a KB (Craw and Sleeman, 1990)(Bouali and Loiseau, 1998) (Boswell and Craw. 2000). The refinement process relies on characterisations of inconsistency and incompleteness to propose modifications that suppress the problems that have been identified. The major problem with this kind of approach is that several possible refinements are computed to correct the KB and the choice between alternative modifications generally relies on heuristics without interacting efficiently with the expert (Boswell et al. 1996) (Bouali and Loiseau, 1998). Moreover, refinement often deals with inconsistency and incompleteness in independent ways. In fact, inconsistency is corrected by reducing the deductions that can be obtained from the KB, even though incompleteness is corrected by increasing the possible deductions.

This paper presents a new method that deals simultaneously with inconsistency and incompleteness and constructs a refinement of a rule base in a close interaction with the domain expert.

To have a fruitful interaction, we think that the system must question the user about understandable modifications of knowledge. So in our system, an interaction has the following form: we present to the user a rule that is involved in several anomalies (inconsistencies or incompleteness) and a set of elementary modifications of the rule that contribute to solve one or several anomalies. The user makes a choice between the different possible revision actions, i.e. modifications on the rule; then the set of anomalies and of possible refinements is updated according to this choice and the process is iterated until all the anomalies have been treated or until all the possible refinements have been considered.

So, the contributions of this paper are two fold: first, we treat inconsistency simultaneously with incompleteness, which avoids suggesting solutions that are good from only one point of view (consistency or completeness) but bad for another; second, the user directly controls the construction of refinement that guarantees the best solution to be achieved.

This paper is organized as follows. The first part presents our assumptions, how to compute characterizations of inconsistency and incompleteness, how to obtain revision actions, and finally a solution to compute local revision plans that make an anomaly disappear. The second part presents our approach to refine by interaction. It uses the local revision plans to suggest a rule to the user and to suggest modifications on it. This process is done iteratively.

II. Characterization of anomalies and local refinement

II.1 Assumptions

This work uses a formalism based on propositional logic. We consider that a KB is a rule base, noted *RB* made up of a set of revisable rules *rb* and a set of constraints *RC* that are considered as valid, non revisable knowledge. Our KB may contain negations. The KB designer also defines the set of input literals; these literals are the only literals that can occur in a input fact base, and they do not appear in the rule conclusions.
We also suppose that the KB designer provides a set of test cases, noted T_{test}. A test case, noted <*If* F_{test} *Then O*> describes a situation for which the designer surely knows the output *O* associated to the input fact base F_{test} . In fact, we suppose that a set of identified situations are available and can be used to define this valid knowledge. This is very often the case for example in medicine where the patients files provide data which allow the area of expertise to be described.

Example: this example will be followed all through this paper. We suppose that *RB* is composed of five revisable rules *r1, r2, r3, r4, r5* and one constraint *Cons1*. The set of input literals is {Shareholder, Bachelor, NoChild, Student, NoGrant, NoWage, Owner}. We consider also that T_{test} contains *T1, ... T5.*

r1 : If Shareholder **and** NoCharge **Then** Taxable
r2 : If Shareholder **and** Bachelor **and** NoChild **Then** NoCharge
r3 : If Student **and** NoGrant **Then** Helped
r4 : If NoWage **Then** Helped
r5 : If NoCharge **Then** ¬Helped
r6 : If Owner **Then** ¬Helped
Cons1 : If NoWage **and** Owner **and** NoChild **Then** ⊥ (⊥ means contradiction)
T1 : <**If** Shareholder **and** NoChild **Then** Taxable>
T2 : <**If** NoChild **Then** NoCharge>
T3 :<**If** Student **Then** Helped>
T4 :< **If** Owner **and** NoChild **Then** ¬Helped>
T5 :<**If** Shareholder **and** NoChild **Then** ¬Helped>

II.2 Characterization of inconsistency and incompleteness

The validation stage of our method consists of studying the inconsistency and the incompleteness of the rule base. In our previous works (Djelouah 2002), we have proposed formal definitions of consistency and completeness of a rule base. Our definition of consistency extends the definitions that have been previously proposed (Rousset 1988) (Cordier and Loiseau 1996) and uses the test cases to better determine all the possible inconsistent situations.

We have also proposed precise characterizations of inconsistency and incompleteness of a rule base. These characterizations are called *conflicts* and *deficiencies* and we recall below their definitions.

Definitions.
- A *conflict* is a couple <F_c, br_c>, composed of an input fact base F_c that satisfies the constraints, and of a set of revisable rules br_c, such that $F_c \cup br_c \cup T_{test} \cup RC$ enables to infer a contradiction.
- A *deficiency* is a test case <*If* F_{test} *Then O*> such that the output *O* cannot be deduced from *RB* and F_{test}.

The computation of conflicts and deficiencies is made with an adaptation of the algorithm ATMS (DeKleer 1986). The idea is to compute for each literal *L* all the input fact bases that enable to deduce *L* (our algorithm also provides the set of rules that is used to produce L from an input fact base). The conflicts correspond to the conditions of deduction of the contradiction, and following the principle of parsimony, we consider only the minimal conflicts. Deficiencies are obtained by comparing the conditions computed for the output literal *O* of a test case and the input fact base of the test.

Example. the rule base *RB*, together with T_{test}, gives five conflicts.
C1 :<{Student, Owner}, {r6}>. *C1* is a conflict because the facts {Student , Owner}, with the rule *r6* and the test *T3* lead to the deduction of *Helped* et *¬Helped*
C2 :<{Student, NoChild}, {r5}>
C3 :<{NoWage, NoChild}, {r4,r5}>.
C4 :<{NoWage, NoChild, Shareholder}, {r4}>,
C5 : <{NoWage, Owner}, {r6,r4}>
Four elements of T_{test} are deficiencies.
D1 = T1. In fact, with the facts *{Shareholder and NoChild}* from *T1* and the rule base, it is impossible to deduce the output of the test, *Taxable*.
The other deficiencies are *D2* = T2, *D3* = T3, *D4* = T5.
On the contrary, *T4* is a test proven by the rule base because the facts {*Owner, NoChild*} and the rule *r6* enable to deduce *¬Helped*.

II.3 Local refinement of conflicts and deficiencies

The refinement task consists in modifying the rule base in order to make it consistent and complete. This task begins by a local refinement which searches the possible solutions to repair each detected anomaly, conflict or deficiency. To achieve this, we consider a set of refinement operators, which are elementary modifications of the rule base; these modifications are called here *revision actions.* For each conflict and each deficiency, we compute the different ways of repairing this anomaly by combining revision actions.

Definition. A *local refinement plan* for a conflict, as well as for a deficiency, is a set of revision actions which removes the conflict or the deficiency.

To remove a conflict, we can either forbid the input fact base of the conflict by adding a new constraint to the base, or specialize the rule base by removing rules or adding one or several premises to some rules. Our refinement method uses the following revision actions:

- *add_cons(FB)*, means adding the integrity constraint (If FB Then ⊥) to the constraints RC (FB is an input fact base of a conflict),

supp_rule(r), means removing the rule *r* from the rule base RB,

add_prem(r,L) means specializing the revisable rule *r* by adding the literal *L* to its premises.

So, for a conflict $C = <F_c, br_c>$, we propose the following local refinement plans:

- $P = \{add\_cons(F_C)\}$, if the fact base of the conflict F_C is not included in the input fact base of a test (which would prove it valid),

$P = \{supp\_rule(r)\}$ with *r* belonging to br_c

$P = \{add\_prem(r_1, L_1) +...+ add\_prem(r_k, L_k)\}$

Actually, we consider only plans which contain at most two revision actions. This assumption is justified because we suppose that the KB needs only minor repairs and so a close variant of it must be correct.

The refinement plans for the conflicts do not generate new conflicts because they specialize the rule base and reduce the set of possible deductions. Moreover, we retain only the plans that do not introduce new deficiencies. For that, we have proposed in our previous works conditions allowing to construct such plans (for example, a rule is not suppressed if it contributes to prove a test).

To remove a deficiency, we must generalize the rule base and for that, we use the following revision actions:

add_rule(r), consists in adding the rule *r* to *rb*

- *supp_prem(r, L),* consists in suppressing the literal *L* from the premises of *r*,

So, for a deficiency *D = T*, we propose the following refinement plans:

- $P = \{add\_rule(T)\}$, consists in adding to the expert rules a rule equal to the test *T*,
- $P = \{supp\_prem(r_1,p_1)+...+supp\_prem(r_k,p_k)\}$, consists in deleting the premises $P_1,..,P_k$ from the rules $r_1,...,r_k$ respectively, in order to make the test *T* proved by the modified base. Here also, we consider only the plans which contain at most two revision actions.

The refinement plans for the deficiencies do not generate new deficiencies because they generalize the rule base; the first type of plans which consists in adding a rule corresponding to a test cannot introduce new conflicts because the tests are taken into account to compute the conflicts. To verify if the second type of plans ($\{supp\_prem(r_1,p_1)+ ...+supp\_prem(r_k,p_k)\}$ does not introduce new conflicts, we must update some informations of ATMS.

III. Interaction and global refinement

III.1. Global refinement

Once we have computed a set of local refinement plans which are the different possibilities for removing the conflicts and the deficiencies, our objective is to simultaneously refine all the conflicts and all the deficiencies by suggesting a *global refinement plan.*

Definition. A *global refinement plan* is a set of revision actions that applied to the rule base removes all conflicts and all deficiencies.

The work presented in (Bouali and Loiseau 1995) suggests a method to refine a KB when only inconsistencies are taken into account. The authors first calculate for each conflict a set of local refinement plans, then they calculate the possible global refinements, which are fusions of local refinement plans. Such global refinements can be calculated with an algorithm inspired by the diagnostic method of a physical system proposed by Reiter (Reiter 1987).

In our approach, the computation of a global refinement plan cannot be undertaken directly by union of local plans because we have to restore both consistency and completeness. The local refinement plans for conflicts and for deficiencies may contain actions which cannot be combined.

So our proposition is to build a global refinement plan by a close interaction with the user.

The interactive tool, proposed in (Cordier and Loiseau 1996) to refine a KB, computes global refinement plans to refine all the inconsistencies of a rule base and then presents these plans to the user. The major difficulty with this method is that it is difficult for the user to choose between alternative global plans because it is difficult for the user to apprehend the signification of a global plan.

So our proposition is that the user decisions must concern elements that are easily understandable, and that is why the interaction in our method involves only rules and elementary actions on rules.

III.2. Interactive refinement of the base

The first step of our refinement system is to deal with the constraints of the domain. We have seen that some conflicts can be resolved if some integrity constraints are added to the KB. Deciding whether a constraint is relevant or not for the domain is generally a task that the user performs easily. That is why our search for a global refinement plan first submits to the user the set of integrity constraints that could contribute to solve the conflicts. For each constraint, the user decides if it must be added or not to the set of constraints RC.

Example. The five conflicts of our example bring up the following constraints as suggestions to the user
add-cons(student, Owner), add_cons(student, NoChild), add_cons(NoWage, NoChild), add_cons(NoWage, NoChild, Shareholder), add_cons(NoWage,Owner)

After this first interaction, a certain number of conflicts could have been resolved by the addition of constraints and therefore we continue the process of refinement by considering the unsolved conflicts and all the deficiencies as well as the associated plans that we have computed .
The set of conflicts, deficiencies and local refinement plans are represented by a three level graph. The first level of the graph is composed of vertices, that represent the conflicts and deficiencies that must be repaired. The second level contains all the local refinement plans that have been computed for these conflicts and deficiencies, and the third level contains revisable rules that occur in the local plans. A conflict or a deficiency is connected to the local plans that solve it and a local plan is connected to the rules that it contains (see Fig 1).
A global refinement plan of the rule base thus consists of a set *S* of local plans (level 2 nodes), which contains only compatible actions and such that each conflict and each deficiency is connected to an element of *S*.
The basis of our method is to interact with the user on rules. We want to present to the user a rule, considered as incorrect, as well as the revision actions that we suggest for that rule. So, according to the information contained in the graph, we have to determine which rule is a good candidate for modification.
To examine this graph, we proceed in two stages: A '*descending'* stage and an '*ascending'* stage. (see Fig 1). In the first stage, we sort the local refinement plans; and from this classification, we determine a rule which is an interesting candidate for presentation to the user. Intuitively, the order on the plans favours those plans that resolve the greatest number of anomalies and that only require slight modifications of the KB. We then choose a rule that often occurs in the preferred plans. The second stage, that we could qualify as ascending, takes into account the modifications chosen by the user and therefore updates the graph (set of local refinement plans as well as the conflict and deficiency sets). We now describe these two phases in more details.

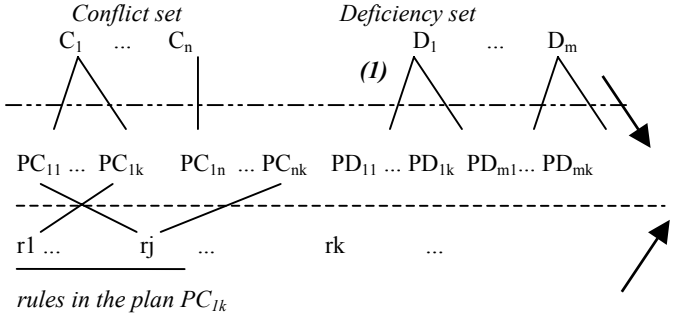

Fig.1

Determination of the element of interaction. The main idea of this phase of examination of the graph is to localise the most interesting plans so as to define the rule that we will suggest to the user. For that, we sort the refinement plans according to several criteria.
In our approach, we define three criteria of classification that we combine to give each plan a value called *plan interest* that determines the final classification. Our objective is to eliminate the greatest number of conflicts and deficiencies and also to modify the base as little as possible. The first two criteria estimate a plan from a quantitative point of *view*: *cr1* is equal to the number of inconsistencies and deficiencies resolved by the plan; *cr2* depends on the number of revision actions in the plan, it is set to 1 if the plan contains only one action and to 0 otherwise. The third criterion is a qualitative evaluation of the modification proposed by this plan; in fact, to stay as close as possible to the original knowledge, it is preferable to modify the rules rather than to add or subtract rules from the base; so *cr3* is equal to 1 if the plan contains only modifications of rule and to 0 if it contains additions or suppressions of rule.
We define, as well, an order of importance (Vincke 1992) of criteria *cr1, cr*2, and *cr3* by giving a weight to each criterion. Sorting the different plans is made by calculating the *interest* of each plan, which corresponds to a weighted sum of the values of the criteria.
The weights of the criteria can be given by the user. In our system, we have defined default weights, which correspond to a lexicographic order on *cr1* first, then *cr2* and finally *cr3*.

Definition. The interest of a plan *P* is defined by :
$Interest(P) = \sum_{i=1..3} K_i\ cr_i(P)$
where K_i represents the weight of the criterion and $cr_i(P)$ the value of criterion cr_i for the plan P.

Example. For the conflicts C1, C2, C3, C4 and C5 we have the following sets of refinement plans :
C1 **: {**{add_prem(NoChild, r6)**}**,
C2 **:{**{supp_rule(r5)}, {add_prem(Owner,r5)}**},**
C3 **:** **{**{supp_rule(r5)}, {supp_rule(r4)}, {add_prem(Owner, r5)}, {add_prem(Owner, r4}**}** **,**
C4 **:{** {supp_rule(r4)} **},** pour C5 **:{**{supp_rule(r4)}, {add_prem(NoChild, r6}, {add_prem(NoChild, r4}**}**
For the deficiencies D1, D2, D3 et D4 we have the following sets of refinement plans :
D1 **: {**{supp_prem(Bachelor, r2)}, {add_rule(T1)}**},**
D2 **:{**{add_rule(T2)},{supp_prem(Bachelor,r2)+ supp_prem(Shareholder, r2)}**},**
D3 **:{** {add_rule(T3)}, {supp_prem(NoGrant, r3)}**}**,
D4 **:{**{add_rule(T5), {supp_prem(Bachelor, r2)}**}**
The plan *P = {supp_rule(r4)}* has the greatest value of the interest ; in fact, *P* has the greatest value for the criteria *cr1* because *P* refines three conflicts.
Sorting the refinement plans allows the definition of a set called *maximum interest class*, which contains all the plans having the maximal value of interest. From this set, we determine which rule will be presented to the user (if

several rules have the same number of maximal occurrences in this class, we choose arbitrarily one of them). The idea of this heuristic is that if a rule has multiple occurrences in the interesting plans, it is natural to think that a modification of the knowledge contained in this rule is pertinent. Once the rule is determined, the interaction with the user consists of presenting this rule as well as the set of revision actions appearing in the different previously calculated plans.

Example. The maximum interest class contains only the plan $P = \{\{supp\_rule(r4)\}\}$; the candidate rule is therefore the rule *r4*. We present to the user this rule and the revision actions for this rule that occur in the different local plans.

Updating the graph after an interaction. When we submit a candidate rule and a set of possible revision actions for this rule, the user must choose between *apply a proposed action* or *reject a proposed action*. Then our system must take into account the user's choices to update the set of conflicts and deficiencies remaining and therefore the set of refinement plans to be considered.

If the user applies a revision action *RevAction* to a rule , two cases are possible:
- If a plan contains only this revision action, the conflict, respectively the deficiency, associated to this plan is repaired and the other refinement plans for this conflict can now be ignored.
- If a plan contains two revision actions one of which is the action *RevAction*, we suggest immediately to the user this associated action in order to apply the plan and to resolve the conflict or the deficiency.

Example. If the user applies the action *supp_rule(r4),* the conflicts C3, C4, C4, C5 are solved and consequently they are removed from our graph as well as the refinement plans connected to them.

If a user rejects a revision action on a rule, all the plans containing this action are removed from the set of possible local plans.

At the end of this ascending phase that updates the graph, if all the conflicts and deficiencies have not yet been treated and if there are still refinement plans for these anomalies, we re-evaluate the three criteria for these plans and establish a new classification which allows us to determine which candidate rule will be proposed to the user in the following iteration.
This iterative refinement goes on until all the conflicts and deficiencies are resolved or until the set of remaining local refinement plans is empty.

III.3 The tool ICC

We implement our interactive method of refinement with a Java program called *ICC* (Interaction for Consistency and Completeness). In ICC, first a user interface presents the different constraints that can be added to the rule base. Each constraint is associated with a justification presenting the different conflicts that one can treat using this action. A second menu then proposes to modify the expert rules of the base. Each candidate rule is presented with a set of revision actions. To guide the user in his choices, every action can be linked to a window giving details of the conflicts or deficiencies treated by this action.

The ICC system was tested on a rule base available on the Irvine site (Black and Merz 1998). This example called Moral Reasoner Domain consists of a Horn clause theory simulating the reasoning of a person with guilt feelings, and a set of positive and negative examples of the concept « guilty ». We transformed this Horn theory by using real negations. The rule base thus obtained has 32 rules and a depth of deduction of 7. The number of rules is not high, but still gives 218 different ways of proving the output « guilty ». We had voluntarily introduced several errors in order to make the base incomplete and inconsistent and we had selected amongst the positive examples 8 examples that we used as a test-set. The refinement of the rule base thus produced allowed us to verify that with an example that we did not know, the interactive method effectively guides the user in his choices. Now we have to supply a KB user to see how he evaluates the interface we are proposing.

IV. Related work

One of the major systems of refinement that uses a set of tests is SEEK2 (Ginsberg 1988). This system deals with inconsistency and incompleteness and uses statistical data to suggest a set of refinements. The major limits of this system are the particular formalism of the rule representation, and the partial characterisation of inconsistency and incompleteness. Its advantage is that it offers two types of refinement: one is automatic and the other interactive. The interactive mode proposes refinements of rules, that the user can accept or not.

More recently, (Cordier and Loiseau, 1996) proposed the system XGT that has common points with ICC. XGT only deals with inconsistencies; for each inconsistency, a set of local refinements are proposed and these local modifications are then combined to construct a global refinement of the KB. These different global solutions are suggested to the user. The problem is that this method proposes to the user a large number of plans, among which it is difficult to choose the most relevant.

KRUST (Craw and Sleeman 1990)(Craw and Sleeman 1995) refines propositional KB systems, where the inference engine uses rule ordering as the conflict resolution strategy. Negation is not allowed in this representation. KRUST refinement operators consists of the usual generalization and specialization operators on rules (nearly the sames as in ICC) and in addition KRUST

also uses operators that change the rule priorities for the inference engine (by modifying the rule position in the KB). For each misclassified example, KRUST determines the possible causes of this problem and then construct a list of possible refinements. After a filtering step that eliminate some of these refinements (for example, conflicting modifications of the same rule are not possible), KRUST implements all the remaining refinements, which leads to a set of possible refined KBs. All these alternative refined KB are tested over the entire set of tests and the KB with the best performance is finally proposed to the user. This phase of choice consumes a lot of processing time. More recently, KRUST has been extended in a generic toolkit (Boswell and Craw 2000) that can deal with different shells by using generic representation for the rules and for the reasoning process.

(Carbonara and Sleeman 1999) presents the system STALKER that is very simlar to KRUST because it proposes many alternative refinements for each misclassified example. The differences between the two systems are that STALKER uses induction to propose new refinements and it uses a Truth Maintenance System for the testing phase, which makes it more efficient.

It must be noted that, in these two systems that do not use negation, the faults in the KB are revealed by the set of tests, but the notion of global consistency of the rules, which is central in ICC, is not considered.

V. Conclusion

This paper presents a new method which detects and repairs the inconsistency and the incompleteness of a rule base. The inconsistency and the incompleteness are respectively revealed by the conflicts and the deficiencies. First, our method computes all the conflicts and all the deficiencies of the rule base; then, for each conflict or deficiency, it computes a set of revision actions. The application of a revision action to the rule base contributes to eliminating a conflict or a deficiency. Secondly, our method proposes an interactive and iterative solution to restore consistency and completeness. At each iteration, our refinement method offers to the user a rule for a repair and a set of possible revision actions on this rule. To localize the rule to be refined, the sets of revision actions are classified by using heuristics criteria. These criteria express two priorities: to eliminate the greatest number of conflicts and deficiencies, and to modify the initial base as little as possible. The iterative process of our method ends when all conflicts and deficiencies are resolved or when all revision actions are proposed to the user.

Our method presents two major advantages on preceding approaches: first, we treat inconsistency simultaneously with incompleteness which avoids proposing some bad solutions, second, the user controls directly the construction of the refined KB.

References

Ayel, M. and Bendou, A. 1996. Validation of rule bases containing constraints. ECAI 96, Workshop on Validation, Verification and Refinement of KBS, p. 96-101.

Black, C.L. Merz, CJ. 1998. UCI Repository of machine learning databases Irvine, CA: University of California, Departement of Information and Computer Science, [http://www.ics.uci.edu/~mlearn/MLRepository.html].

Bouali, F. and Loiseau, S. 1995. Rule base diagnosis for debugging : the KB-DIAGO2 system. EUROVAV 95, p.225-239.

Bouali, F. and Loiseau, S. 1998 Révision d'une base de règles. RFIA 98, p. 225-232.

Boswell, R. and Craw, S. 2000. Experiences with generic refinement toolkit. EKAW 2000, p 249-256.

Boswell, R. and Craw, S. 1996. Refinement of a product formulation expert system. In Proceedings of the ECAI 96, p 74-79.

Carbonara, L. and Sleeman, D. 1999. Effective and Efficient Knowledge Base Refinement. Machine Learning, Vol 37, p143-181.

Cordier, M.O. and Loiseau, S. 1996. Validation of first-order rule base systems. Computational Intelligence System vol.12(4), p.520-540.

Craw, S. and Sleeman, D. 1990. Automating the refinement of knowledge-based systems. ECAI 90, p. 167-172.

De Kleer, J. 1986. An assumption-based truth maintenance system. AI 28, p. 127-224.

Djelouah, R. and Duval, B. and Loiseau, S. 2002. Validation and reparation of knowledge bases. ISMIS 02, Lectures Notes in Artificial Intelligence. Vol. 2366. Springer. p. 312-320.

Buekenoogen, M. and Gerrits, R. and Spreeuwenberg, S. 2000. VALENS : A knowledge based tool to validate and verify on aion knowledge base. ECAI 2000, p731-735.

Ginsberg, A. Automatic refinement of expert system knowledge bases. 1988. London.. Pitmann Publishing.

Grogono, P., Batarekh, A., Preece, A., Shinghal, R. and Suen C. 1992. Expert System Evaluation Techniques: A selected Bibliography. Expert Systems, p.227-239.

O'Keefe, R.M. and O'Leary, D.E. 1993. Expert system verification and validation: a survey and tutorial. Artificial Intelligence Review vol 7, p. 3-42.

Preece, A. 1997. Evaluation of verification tools for knowledge-based systems. International Journal Human-Computer Studies 47, p. 629-658.

Reiter, R. 1987. A theory of diagnosis from first principles. AI vol 32, p. 57-95.

Rousset, M.C. 1988. On the consistency of knowledge bases: the Covadis System. ECAI 88, p. 79-84.

Vincke, P. 1992. Multicriteria decision aid. Wiley and sons New York 92.