Test Case Generation using KBS Strategy

Laurence Vignollet
Laboratoire d'Intelligence Artificielle
ESIGEC - UNIVERSITE DE SAVOIE
73376 Le Bourget du Lac Cedex
FRANCE

Ruddy Lelouche
Département d'Informatique
UNIVERSITE LAVAL
Québec G1K 7P4
CANADA

Abstract

The system we have developed, SYCOJET, automatically builds test cases making use of specific expert knowledge. In our first version, it uses the internal structure of the knowledge base to be tested, and implicitly assumes that the inference engine derives all that can be derived from the knowledge base and the problem data. Here we are concerned with how to take into account, in a second version of SYCOJET, the actual inference engine control strategy. This control strategy, included in the inference engine, is not in general explicited in a form accessible to the computer. In this paper, we investigate to what extent the knowledge of the control strategy of the system being tested can be used to improve the "quality" of the test cases generated by SYCOJET.

1 Introduction

KBS-testing is a recent field of study [Morell 1988, Miller 1989, Ayel 1993, Gupta 1991]. Some execution monitoring tools, like WITNESS [VALID, 1991], provide information about the KBS running on a given problem. Other tools exist to help in the evaluation of the KBS results, such as CONTROLLER [VALID, 1991] and VORTEX [Cross *et al*, 1990].SYCOJET is a different type of tool: it automatically builds test cases for a KBS; these test cases are then used to study the behaviour of the KBS, its performance and many other validation topics. Other test case generators are described in [Zlatareva *et al*, 1993].

To build test cases, SYCOJET first analyses the KB itself. This analysis is based on building ATMS-like labels; these labels are an adaptation of De Kleer's [De Kleer, 1986].

SYCOJET uses specific expert knowledge in order to propose pertinent test cases. To reinforce the pertinence of the test itself, SYCOJET may use different coverage criteria, freely chosen by the user. When the KBS inference engine exhaustively derives everything that can be derived, the required level of the chosen criteria will be reached when all the selected test cases have been run with the KBS. Thus, when SYCOJET proposes a set of test cases, the level of the chosen criteria will be reached only with this hypothesis concerning KBS strategy.

If the KBS inference strategy can effectively be used in test case generation, the test cases will be of better "quality". Moreover, it will ensure that the required level of the chosen criteria will actually be reached when running the test cases with the KBS. In this paper, we propose two different approaches in order to account for the KBS strategy when selecting the test cases.

The first section presents the basic principles behind SYCOJET [Vignollet, 1991]. It shows why labels might be used to exhaustively "summarize" the deduction capabilities of a knowledge base (KB), and how these labels can be used in order to define test cases. Finally, section 2 demonstrates how some knowledge of the inference engine strategy can be useful to guide the construction of test cases respecting the assigned criteria, and presents two approaches to taking this strategy into account, depending on whether it is formally described and accessible or not.

2 Construction of test cases

In this paper, we present simplified label building for KBSs using a propositional formalism. However, the method can be generalized to KBSs using a first order logic formalism, as can be seen in [Ayel *et al*, 1993].

2.1 KBS formalism

Each KBS considered in this paper includes:
- a factual part describing the domain, or *domain fact base*,
- a factual part describing the problem that the KBS has to solve, or *problem fact base*,
- a *deductive part* describing the expertise required to solve a problem.

Definitions.
- A *fact* is a proposition.
- A fact is *a problem fact* if it may appear in a problem fact base.
- The deductive part of the KBS is a set of monotonic rules of the form

  IF *condition part* THEN *consequent part* [1]
  which can only assert new facts.
- A fact is *terminal* if it appears in a consequent part of a rule and if it does not appear in the condition part of any rule.

[1] The condition part is a conjunction of propositions, whereas the consequent part is a proposition (using a conjunction of propositions in the consequent part of the rule would not fundamentally change the method).

2.4.2 Test case building method

To build the test cases, the first step consists in constructing the deduction labels for each terminal fact of the KB as described above. In this way, we obtain all realistic problem fact bases of the KBS.

As a second step, the test cases are chosen from this set of problem fact bases. This choice is guided by the quality criteria retained but we still have to determine the contribution of each deduction environment to a given coverage criterion.

Definitions.

• The *contribution of an environment* to a coverage criterion is the number of KB components to be covered according to that criterion that have not yet been counted in the set of test cases under development.

• The *contribution of a terminal fact* to a coverage criterion is the maximum contribution of all the deduction environments in its deduction label.

Each newly chosen test case will be one for which the environment from which it is extracted yields a contribution that is maximal.

Example. - Using the example in 2.2, if the quality criterion retained is rule coverage, the first deduction environment chosen will be <{*F1*, *F2*, *F3*, *F8*}, {R1, R2, R5}> because it maximizes the number of rules fired. The corresponding test case is the set of problem facts in this environment, i. e. the set {*F1*, *F2*, *F3*, *F8*}.

3. Using inference engine strategy: two approaches

There are many different possible strategies in the control mechanism of a KBS. J.P. Laurent in [Laurent 1984] gives a general presentation, and we shall certainly not deal with all of them here. However, we can start investigating this field in a general manner.

Since the assumption behind SYCOJET, i.e. that the KBS being tested exhaustively derives all that can be derived [Vignollet, 1991], does not hold in general, it might be of interest to investigate the situations where the actual strategy can be used to prune the search trees while building deduction environments. In other words, it might not be necessary to construct environments that will never be used by the actual system when it is run. However, this possibility is kept as a longer term approach, and the remainder of this paper deals exclusively with what can be done when exhaustively constructing the KB labels, as done in SYCOJET.

In order to take into account the inference engine strategy when deriving test cases from the computed labels, one can distinguish two different approaches. In the first (3.1), the strategy is known, formally expressed, and can be exploited; in such a case it may be taken into account in the choice of test cases, as is shown with a particular strategy. In the second (3.2), the strategy is not known, or cannot be automatically exploited, and we use a more empirical approach.

3.1 Known and exploitable strategy

The case wherein the inference engine strategy is known is intellectually more challenging, because it could possibly lead to some kind of automatic generation (i.e. "intelligent selection") of actual test cases. However, there are few chances indeed that it can actually be of general use, i.e. for all existing strategies.

Still, for particular inference engine strategies, strategy-specific knowledge can sometimes be used and lead to interesting results independent from the domain knowledge or from the validation knowledge itself. That allows us to go beyond the SYCOJET case where all possible deductible facts are derived. Such is the case, for example, when the conflicts are solved using weights on the deduction elements (e.g. rules) and the system stops as soon as the first terminal fact is reached, as we shall now demonstrate.

3.1.1 Strategy hypotheses

In order to simplify our discussion, we still assume that the KB deduction formalism is of the propositional logic type. Moreover, we now also assume that we deal with a KBS where:

• each rule is marked with some interest coefficient (or weight), and any conflict is solved by always firing the rule with the highest coefficient (note that the multiple maximum case is only theoretic, since either it is solved by executing some sub-procedure based on another condition, or the actual conflict resolution is implementation dependent[7]; we therefore assume that there is always only one highest coefficient rule[8]);

• the system stops as soon as a terminal fact has been reached.

Incidentally, let us remark that, under these assumptions, *some satisfaction criteria may prove to be unattainable.*

For example, if there are two rules in the KB with the same fact(s) as premise(s), the rule with the lower coefficient will never be fired. Therefore a test criterion specifying "execute every rule at least once" cannot be met. The same holds true if the rule with the higher coefficient has the form R1: SP1 $\rightarrow$ SC1 (where SP1 is the set of premise facts and SC1 is the set of derived facts), and the rule with the lower coefficient has the form R2: SP1 $\cup$ SF $\rightarrow$ SC2, for some non empty set of facts SF. With the test criterion above, in order to eliminate this problem, we would have to verify that the premise set of facts of any rule is never included in the premise set of facts of another rule with a lower coefficient.[9] (but this can be done automatically).

3.1.2 A first result: potentially conflicting environments

Lemma. - Under the strategy hypotheses in 2.1.1, when executing the KBS, the problem facts of an environment E_x might lead to using another environment $E_y \neq E_x$ only if the problem facts of E_y are included in those of E_x.

---

7 Actually, if no sub-procedure exists, the first - or the last - rule with the highest coefficient will be fired, depending on the implementation.

8 Note that this assumption *cannot* be automatically verified.

9 The above conclusion shows that, in spite of the distinction traditionally made between verification and testing, the two aspects may be inter-related.

2.2 Example

This example illustrates our method used to build test cases with a propositional formalism. Let us assume that we have the following rule base:

R1: $F1, F2 \rightarrow F4$ [2]
R2: $F3, F4, F9 \rightarrow \underline{F6}$
R3: $F1, F3 \rightarrow F7$
R4: $F2, F7 \rightarrow \underline{F10}$
R5: $F8 \rightarrow F9$

where the problem facts are italicized and the terminal facts are underlined[3].

Then, for each terminal fact TF (F6 and F10), we build the problem fact bases which can lead to TF deduction. Furthermore to build test cases we need to associate to each problem fact base the rules used to deduce TF from it. The couple of the problem fact base and the rule list is called the deduction environment of TF. All the deduction environments of a terminal fact constitute its deduction label. For our example, we have:

Terminal fact	*Corresponding deduction label*
F6	{ <{*F1*, *F2*, *F3*, *F8*}, {R1, R2, R5}>, <{*F3*, *F4*, *F8*}, {R2, R5}> }
F10	{ <{*F1*, *F2*, *F3*}, {R3, R4}> }

If the user's objective is to use 60% of the rules when running test cases, we propose one test case which is: {*F1*, *F2*, *F3*, *F8*}. In this case, rules R1, R2 and R5 are supposed to be fired when the KBS runs. However, if the KBS inference engine uses a specific strategy, these rules might not be fired.

In the following sections we present the formalized method to build the deduction label of a terminal fact and to build test cases taking into account quality criteria.

2.3 Building ATMS-like labels

The objective of label building is to make explicit all the problems, i.e. all the problem fact bases, through which the KB terminal facts can be derived. Our building method is a variation of De Kleer's ATMS labels [De Kleer, 1986]; in fact, we adapt the ATMS concepts to our problem by defining the deduction environments and the deduction label of a terminal fact.

Definitions.

- A *deduction environment of a terminal fact* TF is a couple of the form <SPF, SR>, where SPF is a set of problem facts and SR a minimal set of rules used to derive TF from SPF[4].
- A set of problem facts is also called a *problem fact base*.
- The *deduction label of a terminal fact* TF is the set of all its deduction environments.

Each terminal fact may have several deduction environments; the set of these environments is then the deduction label of the terminal fact.

To construct the deduction label of a terminal fact TF, we have to build the search tree of TF, by backward chaining. We thus obtain all the *potential problem fact bases* allowing deduction of TF.

Each of these potential problem fact bases does not necessarily correspond to a *real* problem fact base. Thus, the set of all real problem fact bases is included in the set of all potential problem fact bases. In order to refine this last set, the domain expert is asked to define constraints to reinforce the realism of its elements, in order to build a set of *realistic* problems[5]. We will not develop this aspect here but the reader may refer to [Ayel & Vignollet 1992]. In the following, we suppose that the only deduction environments that are built are those containing a realistic problem fact base.

The next section shows how these deduction labels can be used to build pertinent test cases, i.e. test cases that detect a maximum number of errors, as implemented in SYCOJET.

2.4 The construction

The test cases are provided to verify correct system behaviour. Whenever the results obtained by the KBS do not fit the results expected by the expert, the presence of at least one error has been detected in the KB. Then the debugging task (under the expert's control) is responsible for locating and correcting this error (or these errors); this last task is not our topic here.

2.4.1 Test quality criteria

As long as the KBS results are accepted by the expert, testing goes on, i.e. adequate test cases are fed into the system, until the latter is believed to have been sufficiently tested. This decision can be difficult to make and therefore, in order to help in decision-making, we propose using *quality criteria*.

In general, these quality criteria are *coverage criteria* such as:
- coverage of facts, like problem facts, terminal facts, etc.;
- coverage of rules, or of groups of rules;
- coverage of paths[6].

Note that coverage is not necessarily exhaustive, and that it may be of a mixed nature (e.g. coverage of 75% of the rules and 90% of the terminal facts). In any case, the retained criteria can then be used:
- to chose the test cases,
- to assess the quality of any test case, according to the components of the KB that will be used during the execution of that test case by the KBS.

---

2 Note that a derivable fact may also be a problem fact.

3 The definitions of these special facts are given in the next section.

4 Note that SPF has also to be sufficiently large to allow the deduction of TF, but such that withdrawing any facts from SPF would no longer allow the deduction of TF. In other words, it constitutes a minimal problem fact base.

5 The real problems are included in the realistic ones because we cannot be sure that the expert's constraints are sufficient to guaranty the realism of the askable fact bases. These constraints may greatly reduce the set of potential problem fact bases, but in general yield only a superset of real problems ; it is this superset that we call the set of realistic problems.

6 At this stage of our research, we consider only monotonic KBSs. That means that path coverage is the same as groups of rules coverage.

Proof. - The proof is trivial considering the completeness of the labels.

Example. - Using the same example as before (see 2.2), one can see from the labels only that:

• if facts F1, F2, F3 *only* are assumed to hold, the system will always use the deduction environment $E_{10} = <\{F1, F2, F3\}, \{R3, R4\}>$ and derive F10; in fact, F6 cannot be derived because F8 would be needed as a problem fact;

• if facts F1, F2, F3, *and F8* are assumed to hold, depending on the rule firing order, the system might either use the environment $E_6 = <\{F1, F2, F3, F8\}, \{R1, R2, R5\}>$ and derive F6, or use the environment $E_{10} = <\{F1, F2, F3\}, \{R3, R4\}>$ and derive F10.

The above lemma justifies the following definition:

Definition. - A deduction environment E_y is said to be *potentially conflicting with another one* $E_x \neq E_y$ whenever the problem facts of E_y are included in those of E_x.

Example. - In the above example, the environment $E_{10} = <\{F1, F2, F3\}, \{R3, R4\}>$ is potentially conflicting with the environment $E_6 = <\{F1, F2, F3, F8\}, \{R1, R2, R5\}>$.

Note that the concept of potentially conflicting environments is not symmetric, since the set inclusion relation is not.

The lemma can then be formulated as: the problem facts of an environment E_x might lead to using another environment $E_y \neq E_x$ only if E_y is potentially conflicting with E_x. Interestingly enough, it may be used in two different ways:

• with a terminal fact coverage criterion, if E_x and E_y are in the labels of two different terminal facts $TF_x \neq TF_y$, then an actual conflict might lead to deriving TF_y rather than TF_x (that is the case with the example above);

• with a rule coverage criterion, even if E_x and E_y are in the same label (i.e. they lead to the same terminal fact), the derivation path of E_y might be followed rather than that of E_x.

However, *the condition is only a necessary but not a sufficient one*; even if it holds, the system might still use E_x as expected, which is a relatively weak result. Can we strengthen it, taking the interest coefficients into account ? Fortunately we can, and the result which we arrive at is the following:

Theorem. - Let $E_x = <SPF_x, SR_x>$ be an environment and $E_y = <SPF_y, SR_y>$ be another one potentially conflicting with E_x. Let m_x and m_y be the minimum of the coefficients of the rules in SR_x and in SR_y respectively. Then, under the strategy hypotheses 3.11, executing the KBS with SPF_x will use E_y rather than E_x iff $m_y > m_x$.

Proof. - If $SPF_y \subseteq SPF_x$, SPF_x contains a set of problem facts needed to use the environment E_y, executing the system with SPF_x will successively fire rules from SR_x and rules from SR_y (and possibly some other rules coming from other deduction environments) in a given order, according to the conflict resolution strategy given above (i.e. whenever several rules can be fired, the one with the highest coefficient is fired first).

Let us first assume that $m_y > m_x$, and prove that E_y is used rather than E_x, i. e. that the rules in SR_y are fired first and that assuming the opposite leads to a contradiction. Let us assume that the system uses E_x, i. e. that the rules in SR_x are fired first. Let $R_{mx} \in SR_x$ be a rule with coefficient m_x (there must be one by definition). R_{mx} is on the derivation path made up of the rules in SR_x (using the environment E_x). When R_{mx} is to be fired, all the rules $R_j \in SR_y$ (needed to use the environment E_y) have been fired, since they all have a coefficient $c_j \geq m_y > m_x$. Therefore E_y has been used and the system has stopped without firing R_{mx}, which contradicts the fact that it uses E_x and fires all the rules in SR_x.

Conversely, suppose that the KBS uses E_y rather than E_x, i.e. fires all the rules in SR_y before firing all the rules in SR_x. We could similarly show that the hypothesis $m_x > m_y$ leads to a contradiction, but (for a change) we are to prove directly that $m_y > m_x$. Indeed, there exists a firing order in SR_x alone when using E_x, and a firing order in SR_y alone when using E_y. When the system has used E_y, the next rule $R_i \in SR_x$ which would have been fired if the system had not stopped has a coefficient $c_i \geq m_x$ (by definition of m_x). If it has not been fired, it is because all other conflicting rules $R_j \in SR_y$ had a coefficient $c_j > c_i$ (because of the conflict resolution strategy; remember that the case $c_j = c_i$ has been excluded as purely theoretic). The minimum m_y of these c_j also satisfies $m_y > c_i$. Therefore $m_y > c_i \geq m_x$, and the proof is complete. ♦

Notation. - Since the theorem above shows the importance of the minimal coefficient of the rules in the deduction path associated with any deduction environment, it might be desirable, when constructing the deduction labels, to compute this minimal coefficient along with the set of problem facts and the set of rules of the environment, and integrate it as part of the deduction environment. Thus, under the strategy hypotheses above, we shall write

$E = <SPF, SR, m>$

when we want to make this minimal coefficient explicit and associate it with the environment E.

Example. - With the same example (see 2.2 and 3.1.2), assume that we are to run the system on the set $\{F1, F2, F3, F8\}$, i.e. the problem facts corresponding to $E_6 = <\{F1, F2, F3, F8\}, \{R1, R2, R5\}>$.

First suppose that the rule interest coefficients are:

R1: 10; R2: 15; R3: 20; R4: 12; R5: 30.

Note that the rule R5, which uses the fact F8 as a premise (F8 is the discriminating fact between the sets of problem facts of E_6 and E_{10}), has the highest interest coefficient. However, the system will successfully fire the rules R5, R3, R4 in that order, and stop after deriving the terminal fact F10, i.e. after using E_{10}.

The reasons are that:

1º E_{10} is potentially conflicting with E_6,

2º $m_{10} = \min(20, 12) > m_6 = \min(10, 15, 30)$,

and, by the theorem above, E_{10} is used rather than E_6 (the conflict is effective). Thus $\{F1, F2, F3, F8\}$ should not be proposed as a test case since it would not yield its "theoretical" contribution to the quality criteria (rule coverage for instance). Instead, one should look for another environment, i.e. the one giving the best theoretical contribution among the environments not already selected or marked as unattainable.

Now suppose that the coefficient of R4 is 8 rather than 12, leaving the other coefficients unchanged. Then the rules will be fired in the order R5, R3, R1, R2, and the system stops after reaching F6, i.e. after using E_6. Here:

1º E_{10} is still potentially conflicting with E_6,

2º but $m_{10} = \min(20, 8) < m_6 = \min(10, 15, 30)$,

and, by the theorem above, E_6 is used as expected and not E_{10} (the conflict is not effective).

Again note the generality of the above lemma and theorem: either can be indistinctly applied for rule coverage or for terminal fact coverage, as has been shown above for the lemma.

Also note that, in the above theorem, nothing is asserted in a situation where several environments E_y potentially conflict with a given environment E_x. This is why the corollary below deals precisely with this point.

3.1.3 Using this result for choosing test cases.

As explained in 2.4, all test cases are chosen inside the set of realistic problem fact bases. At each choice point, i.e. whenever SYCOJET is to add a new test case to the set under construction, it selects the deduction environment with the maximal contribution. As already pointed out, this choice is made assuming that the KBS derives all that it can from a problem fact base.

However, when a test case is run by a KBS using a non exhaustive deduction strategy, the system might use a different environment, and therefore take a different derivation path than the one expected from the chosen environment, or reach a different terminal fact than the one corresponding to the chosen environment, etc. Since this may happen at each choice point, the overall result might show a large gap between the criteria level actually reached in executing the set of test cases with the KBS and the SYCOJET-predicted level for the same criteria.

Indeed, when choosing a test case, we must know whether the environment it comes from will be the one actually used when running it on the system. Thus at each stage of test case building, we have to choose the environment *which will actually be used* and with the highest possible contribution.

If the KBS uses rule weights and stops at the first terminal fact encountered (strategy hypotheses 3.1.1), the lemma in 3.12 shows that, if SYCOJET selects an environment E, a degradation can only come from an environment potentially conflicting with E. Moreover, the theorem in 3.1.2 gives a necessary and sufficient condition for a potentially conflicting environment to yield an actual conflict, i.e. to prevent E from being actually used when running the system.

To be sure that a selected environment E would be effectively used in a running session, we have to find all environments actually conflicting with E. The following corollary to the theorem in 3.12 allows to decide if the selected environment E should be used.

Corollary. - Let E = <SPF, SR, m> be a deduction environment. Let

$$CES(E) = \{E_i = \langle SPF_i, SR_i, m_i \rangle \mid SPF_i \subseteq SPF\}$$

be the set of all deduction environments potentially conflicting with E. Then, under the strategy hypotheses 3.11, executing the KBS with SPF as problem facts will use SR iff, for every $E_i \in CES(E)$, $m_i < m$.

Proof. - If the system running SPF actually uses E, then every environment E_i potentially conflicting with E is not in actual conflict, and therefore $m_i < m$ by the theorem in 3.1.2.

Conversely, if for every $E_i = \langle SPF_i, SR_i, m_i \rangle$ potentially conflicting with E, $m_i < m$ holds, then by the theorem in 3.12 none of the E_i will be used. Since no other environment is potentially conflicting with E, E will be used. ♦

This corollary shows that if, for any deduction environment E = <SPF, SR, m> selected by SYCOJET, there exists one deduction environment $E_i = \langle SPF_i, SR_i, m_i \rangle$ potentially conflicting with E and such that $m_i > m$, then we should not use SPF as the next test case to be included in the set of test cases under construction. In other words, the existence of such an E_i implies that E, which has been selected because of its potential contribution to the retained quality criteria, cannot actually yield this "theoretical" contribution to these criteria when running the system.

3.2 The strategy is not known, or not exploitable

The second approach deals with the situations in which nothing particular is known about the KBS inference strategy. In those situations, even though the inference engine strategy cannot be exploited as such, we still have some quality criteria (see 2.4.1) for defining a set of "good", or of "the best", test cases, with respect to these criteria. We then may define two procedures as follows.

The first procedure is an algorithm defined over the existing KB:

1º Compute the deduction labels as shown in section 1.

2º Define the criteria satisfaction indices (e.g. the various coverage counters) according to the retained test criteria and initialize them.

3º WHILE the criteria satisfaction indices are not satisfied AND there are pertinent environments available in the labels:

- Choose a pertinent environment in the computed deduction labels, using the underlying test criteria as a guide for this choice (using the SYCOJET mechanism for choosing a test case).
- Define a test case from this environment, and execute it on the KBS.

• IF the results are correct, i. e. agree with the expert's opinion or conclusion, THEN:

- Analyze the execution traces to update the criteria satisfaction indices.

- Mark any environment(s) in the labels which become unnecessary (i.e. executing the corresponding test would not improve the criteria satisfaction indices).

ELSE {the expert disagree with the obtained results}

End the procedure {there is something wrong in the KB; procedure 2 must be performed next}.

• END WHILE.

4º IF the criteria satisfaction indices are satisfied THEN

The KBS has been successfully tested {the satisfaction criteria are met}.

ELSE {there is no pertinent environment left in the labels}

Testing the KBS is impossible with respect to the defined test criteria.

END of first procedure (over the existing KB).

The second procedure takes place whenever the existing KB proves erroneous somewhere, i.e. when a test case uncovers an erroneous behavior of the KBS. It consists in correcting the KB with the help of the domain expert. This correction cannot be formalized without the adequate domain knowledge, and is therefore outside the scope of this article. Once the KB has been modified, the first procedure above must be run again on the KBS with the new KB.

Conclusion

In the area of test case generation for knowledge-based systems, SYCOJET is an automatic test case building system which has given significant results on several KBSs[10]. The method it uses in this test case construction consists first in building the deduction label of each terminal fact of the knowledge base. These labels provide the set of all potential problem fact bases. Then the actual test cases are extracted from this set, according to the coverage criteria chosen by the user. SYCOJET is operational for KBSs using a first order formalism [Ayel *et al*, 1993]. However, it makes the hypothesis that the inference engine of the KBS deduces all it can. If that is not the case, the criteria level obtained in executing the set of test cases proposed by SYCOJET might be significantly lower than the SYCOJET-predicted level for the same criteria. Thus, taking into account the inference engine strategy will lead to better test case generation.

In this paper, we have proposed two approaches which take the inference engine strategy into account differently.

In the first approach, we suppose that the strategy is known and that it is used to help choose the test cases. This approach cannot actually concern all existing expressible strategies, but we have shown that it can work for some particular strategies. Indeed, we have established some results for a KBS which uses rule weights and stops at the first terminal fact encountered. First, we have introduced the notion of a deduction environment that potentially conflicts with another. We have exhibited and proven a lemma which shows that if an environment E is selected, a degradation can only come from an environment potentially conflicting with E. Moreover, we have established a theorem which gives a necessary and sufficient condition for a potentially conflicting environment to yield an actual conflict, i.e. to prevent E from being used when running the system. These results can lead to more intelligent test case selection.

The second approach to the choice of test cases can always be taken, in particular when the strategy is not explicitly known, or when it is not exploitable. It is a more pragmatic one: a test case will be chosen according to the level of the required criteria already reached by the execution of previous test cases.

The first approach is intellectually more satisfactory but very limited in scope. Our study has shown that, even for a KBS with a well defined but simple strategy, it requires a specific, fairly complex method.

One potentially interesting path for further research could be to examine some other strategies, and investigate whether strategy-specific procedures can be established. Another one would be to try to adequately use the strategy-specific knowledge to prune the search trees used in deduction label construction.

References

[Ayel *et al*, 1993] M. Ayel and L. Vignollet "SYCOJET and SACCO, two tools for verifying expert systems". *International Journal of Expert Systems: Research and Applications* (special issue on validation, to be published).

[Cross *et al*, 1990] S. Cross & M. Grisoni "VORTEX: a methodology for producing validated real-time expert systems". *Conf. Knowledge-Based Systems, Applications for guidance and control*, NATO Advisory Group for Aerospace Research and Development, Proc. Nº 474, pp. 27.1-27.11.

[Gupta, 1991] Uma G. Gupta, ed. *Validating and Verifying Knowledge-Based Systems*. IEEE Computer Society Press (Los Alamitos, Calif.), pp. 176-187.

[De Kleer, 1986] J. De Kleer "An assumption-based TMS". *AI Journal*, Vol. 28, Nº 2, pp. 127-162.

[Laurent, 1984] J. P. Laurent "Control structures in expert systems". *Technology and Science of Informatics*, Vol. 3, Nº 3, pp. 147-162.

[Laurent 1992] J. P. Laurent "Proposals for a valid terminology in KBS validation", *Proc. of the 10th European Conference on Artificial Intelligence*, Vienna (Austria).

[Miller, 1989] L.A. Miller "A comprehensive approach to the verification and validation of knowledge-based systems". *Workshop on Validation, Verification and Test of Knowledge-Based Systems*, IJCAI 89, Detroit (USA).

[Morell, 1988] L.L. Morell "Use a Metaknowledge in the Verification of Knowledge-Based Systems". *Proc. of the IEA-AEI*, June 1987, pp. 847-857. Reprinted in [Gupta 1991], pp. 176-187.

VALID (1991) *VALID: VETA Definition*. ESPRIT II Project 2148, VALID Report Deliverable D4.

[Vignollet, 1991] L. Vignollet *Une approche pour la Construction automatique de Jeux de Données de Test pour des Systèmes à Base de Connaissances*. Thèse de Doctorat. Laboratoire d'Intelligence Artificielle, Université de Savoie (Chambéry, France).

[Zlatareva *et al* 1993] N. Zlatareva and A. Preece "State of the Art in Automated Validation of KnwoledgBased Systems". *International Journal of Expert Systems: Research and Applications* (special issue on validation, to be published).

10 One of them, called GIBUS, comes from the European Spatial Agency (in the scope of ViVa (ESPRIT III Project)).