Verification of Multi-Level Rule-Based Expert Systems[1]

Pedro Meseguer

Centre d'Estudis Avançats de Blanes, C.S.I.C.
Cami Sta Barbara s/n
17300 Blanes (Girona) SPAIN
pedro@ceab.es

Abstract

Verification methods and tools developed so far have assumed a very simple model of rule-based expert system (RBES). Current RBES often do not comply this model and require more sophisticated verification techniques. A RBES model including uncertainty and control has been used to analyze four verification issues (inconsistency, redundancy, circularity and useless RB objects), identifying a number of new verification problems. The concepts of labels and environments (deKleer 1986) have been extended to incorporate uncertainty and control information, obtaining the constructs extended-labels and extended-environments. They have been used to express and solve these new verification problems.

1 Introduction

Expert systems (ESs), like any other piece of software, should be verified and validated to assure their correctness and compliance with the user's requirements. Verification is concerned with the correctness of the ES structure, while validation considers the semantical adequacy of the ES output with respect to its input. Little effort has been devoted to ES verification and validation, especially if compared with other aspects of ES design. However, they are mandatory steps to actually guarantee the correctness of the ES structure and the reliability of its behaviour.

In this paper, the problem of verification of rule-based expert systems (RBESs) is considered. Several verifiers for RBESs have been developed in the last ten years. Most of them assume a very simple RBES model, which is far from the advanced capabilities that current shells offer, inside the rule-based paradigm. There is an important gap between available verifier capacities and current RBES requirements. This paper tries to partially fill this gap, developing a verification framework in which features currently present in many RBESs are included.

As new features, the assumed RBES model includes, (i) an *uncertainty management system* (UMS), and (ii) the existence of *control knowledge* in the rule base. The UMS causes that RBES deductions are no longer boolean, but weighted by certainty values (*cvs*). Verification concepts have to be adapted to this new representation. The threshold τ, able to cut deductions with *cvs* less than τ, is a significant element to determine whether or not a fact can be deduced. Control knowledge, implicit or explicit, plays an important role in the RBES behaviour and it should be verified just like domain knowledge. The union of control and domain knowledge is often organized in a hierarchy of levels, each level acting on levels below. In this hierarchy, verification is performed in two steps: (i) considering each level in isolation, and (ii) considering each level against all upper levels acting on it. This hierarchy causes a number of new verification problems, which have not been considered before because of the flatness of the simple RBES model mentioned above.

To check these new verification problems, the concepts of label and environment (introduced by deKleer (deKleer 1986) and used successfully by Ginsberg (Ginsberg 1988) for inconsistency and redundancy checking) have been extended obtaining the new concepts of *extended-label* and *extended-environment* (e-labels and e-environments, for short). This extension adds information concerning uncertainty and control that is relevant for verification purposes. For all the RBES objects, their corresponding e-label can be computed. Testing a number of simple relations among e-labels, such as set inclusion or set compatibility, the verification issues can be checked.

2 Related Work

Early verifiers like ONCOCIN rule checker (Suwa, Scott, & Shortliffe 1982) or CHECK (Nguyen et al. 1985), check rule bases for consistency and completeness. They are based on static comparison of rules (except for cycles), and consistency is only partially checked. In (Cragun & Steudel 1987) the same approach is followed with optimized algorithms. Other systems, KB-REDUCER (Ginsberg 1988), COVADIS (Rousset 1988), are more focused on consistency checking computing all different sets of input data supporting a given fact (KB-REDUCER also checks redundancy). Another system (Meseguer 1990) transforms a rule base into a Petri Net and detects inconsistencies solving linear equation systems. Finally, (Bareiss, Porter, & Murray 1989) present an approach to limit the part of the knowledge base on which the presence of a rule can have consequences. All these verifiers assume propositional rule bases (except CHECK), boolean truth values, no control (exhaustive firing) and monotonic inference.

1 This work has been partially supported by the ESPRIT II project #2148 VALID.

3 The RBES Model

In this paper, the assumed RBES model is (i) based on rules, underlying propositional logic, (ii) with UMS, (iii) with implicit and explicit control, and (iv) monotonic. In the following a detailed description of this model is given.

A rule base *RB* is denoted by a 5-tuple $<F, R, M, MR, IC>$ where F is a set of *facts*, R is a set of *rules*, M is a set of *modules*, MR is a set of *metarules* and IC is a set of *integrity constraints*. A fact $f \in F$ represents an attribute in the problem domain and has both a value and a certainty value (*cv*) associated. Facts are divided in *deducible* and *external* depending on whether their values and *cv*s can be deduced by the RBES or they must be provided as input. The sets of external and deducible facts are noted by EF and DF. Some special facts are called goals and are used to drive the deduction process. A rule $r \in R$ is formed by a conjunction of conditions on facts in its left-hand side (*lhs*), and an assertion about the value of one fact in its right-hand side (*rhs*). A *cv* is attached to r. When r is fired, the concluding fact is asserted with a *cv* computed from the *cv*s of $lhs(r)$ and r. Rules are fired backwards. Each rule belongs to one module. A module $m \in M$ contains a collection of rules and one or several goals. Rules can use facts deduced in modules different from their own module. A metarule $mr \in MR$ is formed by a conjunction of conditions on facts in $lhs(mr)$, and an action in $rhs(mr)$. Two different types of actions are allowed: on modules and on the whole RBES. Depending on these actions, MR is divided in two sets: MRM and MRS. Metarules acting on modules have a *cv* associated, as a measure of the metarule strength. Metarules are fired forward as soon as their conditions are fulfilled. An integrity constraint $ic \in IC$ is an expression involving *cv*s of one or several facts, which should be satisfied by every deduction in order to avoid inconsistencies. Integrity constraints are not tested in real time. A set of facts is *consistent* if no integrity constraint is violated in it.

Concerning the UMS, only one *cv* is assigned to each fact[2]. A *cv* represents always positive evidence. A fact (f) and its opposite ($\neg f$) are represented separately and each has its own *cv* (that should obey some constraints). Rules are also labeled with a *cv*. To perform logical operations with *cv*s the UMS provides two functions: *cv-conjunction* and *cv-modus-ponens*. The function *cv-conjunction* computes the *cv* of a conjunction of facts from the *cv* of each fact. It is used to obtain the *cv* of the whole $lhs(x)$, $x \in R \cup MR$, when x is going to be fired. The function *cv-modus-ponens* computes the *cv* associated to the consequent of a logical implication from the *cv*s corresponding to the antecedent and the implication. It is used to obtain the *cv* of the $rhs(x)$, $x \in R \cup MR$, when x is fired. The *certainty threshold* τ cuts all deductions with a *cv* less than τ.

Control is divided in implicit and explicit. Implicit control is coded as the conflict-set resolution criteria. Only one of these criteria has been considered: select the the *most specific rule* of the conflict set. Given $r, r' \in R$, r is more specific than r' if $rhs(r)=rhs(r')$ and $lhs(r) \supset lhs(r')$. This criterion induces a mutual exclusion relation between those rules on which a relationship of specificity holds. Let $r, r' \in R$, such that r is most specific than r'. If r has been fired, there is no point in firing r' because all the information contained in r' has already been used in r. So r' should never be fired. Conversely, if r' has been fired, it means that r had been tried but failed. Under the monotonicity assumption r cannot be fired later. Therefore, if r is more specific than r', r and r' are mutually exclusive and cannot occur in the same deduction. Explicit control is coded in metarules acting on modules (MRM) or on the whole RBES (MRS). The role of MRM is to keep updated the active module list (ACL). At each time, the ACL contains the modules more adequate to contribute to the final solution. On modules two actions can be performed, *add m* or *remove m*, meaning that m will be added or removed to/from the ACL. A module can be added several times to the ACL, but once it has been removed, it cannot be entered again. On the whole RBES only the *stop* action can be performed.

The RBES model works as follows: when it starts, a metarule builds up an initial ACL. Then the following cycle starts. A module is selected from the ACL as the current module. Their goals are pursued using the rules contained in it. As soon as new facts are deduced, metarules are tested for firing, and the ACL is eventually updated. When every goal in the current module has been tried, a new current module is selected and the cycle restarts. The RBES stops when no more modules are available in the ACL, or a metarule stopping the RBES is fired. From the previous description it is clear that control and domain knowledge form a knowledge hierarchy, acting the former on the latter. This knowledge hierarchy is translated into the following *RB* object hierarchy,

Level 4: metarules acting on the RBES
Level 3: metarules acting on modules
Level 2: modules containing goals and rules
Level 1: rules acting on facts
Level 0: facts

4 Verification Issues

Four classic verification issues are considered: inconsistency, redundancy, circularity, and useless *RB* objects. Following the hierarchy of levels present at the RBES model, each issue is analyzed in a twofold way: *intra-level*, considering the *RB* objects contained at each level in isolation, and *inter-level*, considering the *RB* objects contained in a level together with all the objects belonging to upper levels.

Inconsistency. A rule base *RB* is inconsistent if conflicting situations among *RB* objects can be achieved from a valid input. An input is valid when it represents some situation that happens in the real world. Assuming that the integrity constraints contained in IC are a good

2. UMSs assigning two certainty values to facts, such as those based on theory of evidence, are not considered.

model of the real world, an input is considered valid if it is consistent. Conflicting situations are:

Intra-level

I-1) At level 3: if after an action *remove m*, an action *add m* is performed, on the same module *m*.

I-2) Al level 0: if two or more facts violating an integrity constraint are deduced together from a valid input.

Inter-level

I-3) Between levels 3 and 1: if the firing of a metarule adding the module *m* prevents a rule in *m* to be fired.

I-4) Between levels 3 and 1: if the firing of a metarule removing the module *m* would hypothetically cause a rule in *m* to be fired.

Redundancy. A rule base *RB* is redundant if it contains repeated or duplicated knowledge. Redundancy can either have no effect on the RBES functionality (just affecting the computational efficiency) or influence some deductions especially when they are weighted with *cv*s. Redundancy between *RB* objects can occur in:

Intra-level

R-1) At level 4, 3 or 1: let $x,x' \in MR \cup R$. Then, x' is redundant with x if (i) $rhs(x)=rhs(x')$, and (ii) whenever x' is fireable, x is also fireable with identical results.

R-2) At level 0: let $f, f' \in DF$, f' is redundant with f if whenever f' is deduced, f is also deduced with the same *cv*.

Inter-level

R-3) Between levels 3 and 1: let $m \in M$, $mr \in MRM$, and $r \in R$, such that $r \in m$ and $rhs(mr)=add\ m$. r is redundant with mr if whenever mr is fireable, r is also fireable.

Circularity. A rule base *RB* is circular if it contains a cycle. A cycle exists if an object depends on itself. Different kind of cycles can exist, as a function of the dependencies among *RB* objects. In the RBES model, two kind of dependencies exist: (i) the dependency of rules and metarules on their respective left-hand sides, and (ii) the implicit dependency of rules on those metarules adding the modules to which these rules belong. The first dependency is denoted by $r \leftarrow f$ (reads "r depends on f"), where r and f are connected by a sequence of zero or more rules. Second dependency is denoted by $r \Leftarrow mr$ (reads "r is allowed by mr"). Potential cycles are:

Intra-level

C-1) At level 1: a fact depends on itself by a rule chain,

$f \leftarrow r$
$r \leftarrow f$ $\quad f \in DF, r \in R$

Inter-level

C-2) Between levels 3 and 1: a fact f depends on rules contained in different modules, and a metarule adding one of these modules depends on f.

$f \leftarrow r$
$r \leftarrow f'$ $\quad f,f' \in DF, r,r' \in R, rhs(r)=f, rhs(r')=f'$
$f' \leftarrow r'$ $\quad m,m' \in M, r \in m, r' \in m'$
$r' \Leftarrow mr'$ $\quad mr' \in MRM, rhs(mr')=add\ m', f \in lhs(mr')$
$mr' \leftarrow f$

C-3) Between levels 3 and 1: two metarules *mr*, *mr'* adding respectively the modules *m* and *m'*, depend on facts f' and f, deduced by rules contained in *m'* and *m*.

$f \leftarrow r$
$r \Leftarrow mr$ $\quad f,f' \in DF, r,r' \in R, rhs(r)=f, rhs(r')=f'$
$mr \leftarrow f'$ $\quad m,m' \in M, r \in m, r' \in m'$
$f' \leftarrow r'$ $\quad mr \in MRM, rhs(mr)=add\ m, f' \in lhs(mr)$
$r' \Leftarrow mr'$ $\quad mr' \in MRM, rhs(mr')=add\ m', f \in lhs(mr')$
$mr' \leftarrow f$

***RB* useless objects.** A *RB* object is useless if it will never be used. Useless objects include non-fireable, unreachable and shadowed objects. An object is non-fireable when it is supported by a non-valid input. An object is unreachable when there exists a gap in the dependency graph linking this object with inputs. An object is shadowed if there exist other objects that prevent it to be used. Potential cases for useless objects are:

Intra-level

U-1) At level 4, 3 or 1: a non-fireable metarule or rule.

U-2) At level 0: let $f \in F$, $r \in R$, $f \in rhs(r)$, f is unreachable if every r is non-fireable.

Inter-level

U-3) Between levels 4 and 3: let $mr \in MRS$, $mr' \in MRM$, mr' is shadowed by mr if mr is always fired before mr'.

U-4) Between levels 4 and 1: let $mr \in MRS$, $r \in R$, r is shadowed by mr if mr is always fired before r.

U-5) Between levels 3 and 2: let $m \in M$, $mr \in MRM$, $rhs(mr)=add\ m$, m is unreachable if every metarule mr is either non-fireable or shadowed.

U-6) Between levels 2 and 1: let $m \in M$, $r \in R$, $r \in m$, r is unreachable if m is unreachable.

U-7) Between levels 1 and 0: let $f \in F$, $r \in R$, $f \in rhs(r)$, f is unreachable if every r is either non-fireable, shadowed or unreachable.

5 Extended Labels and Environments

The concepts of label and environment for a deducible fact f were introduced by (deKleer 1986) in the ATMS context. An environment for f, $E_i(f)$, is a minimal conjunction of external facts supporting f. The label for f, $L(f)$, is the minimal disjunctive normal form of external facts supporting f. Clearly, $L(f)$ includes all the $E_i(f)$ as disjunctions. These constructs has been successfully used in (Ginsberg 1988), but they do not contain all the required information to verify RBESs with uncertainty and control features. Some more information is needed about (i) the *cv* range in which a fact can be deduced and (ii) the control actions required for a fact to be deduced.

An *extended environment* (e-environment, for short) for a deducible fact f, $EE_i(f)$, is a triplet $<SS_i(f), RCV_i(f), RS_i(f)>$. $SS_i(f)$ is a minimal set of external facts supporting f, that is to say, an environment in deKleer or Ginsberg sense. $RCV_i(f)$ is the range of *cv*s in which f can be deduced from $SS_i(f)$. $RCV_i(f)$ is represented by the interval $(LCV_i(f), UCV_i(f)]$, where $LCV_i(f)$ and $UCV_i(f)$ are respectively the lower and upper bounds. $RS_i(f)$ is the rule sequence connecting $SS_i(f)$ with f, and it is recorded for control reasons: (a) to identify e-environments that are incompatible with this one because mutual exclusion

between their corresponding rule sequences, and (b) to identify the set of metarules that have been eventually fired to enable the rules contained in the rule sequence. Two rule sequences RS_i, RS_j are *mutually exclusive* (m-exclusive, for short) if there exist $r \in RS_i$ and $r' \in RS_j$ such that r and r' are m-exclusive. Two rules are m-exclusive if one is more specific than the other (see section 3). The *extended label* (e-label for short) for a deducible fact f, $EL(f)$, is the minimal collection of e-environments for f.

The concepts of e-environment and e-label can also be applied to rules and metarules. An e-environment for x, $EE_j(x)$, $x \in R \cup MR$, is formed by the same components $<SS_j(x), RCV_j(x), RS_j(x)>$ with the same meanings: $SS_j(x)$ is the set of external facts causing x to be fired, $RCV_j(x)$ is the allowed range for the cv of $rhs(x)$, and $RS_j(x)$ is the rule sequence required for x to be fired. Similarly, the e-label for x, $EL(x)$ is the minimal collection of e-environments. E-environments and e-labels for m, $m \in M$, are defined in terms of e-environments for metarules that introduce m in the ACL. Thus, $EL(m) = \cup EL(mr)$, $mr \in MR$, such that $rhs(mr) = add\ m$.

A number of relations can hold between e-labels and e-environments. Let $\mathfrak{EE}$ be the set of e-environments, EE_i, $EE_j \in \mathfrak{EE}$. EE_i is *compatible* with EE_j if (a) $SS_i \cup SS_j$ is consistent, and (b) RS_i is not m-exclusive with RS_j. EE_i *subsumes* EE_j if (a) SS_i is contained in SS_j and (b) RS_i is not m-exclusive with RS_j. EE_i *includes* EE_j if (a) EE_i subsumes EE_j and (b) RCV_i contains RCV_j. Let $\mathfrak{EL}$ be the set of e-labels, $EL, EL' \in \mathfrak{EL}$. EL is *compatible* with EL' if there exists $EE_i \in EL$, $EE_j \in EL'$ such that EE_i is compatible with EE_j. EL is *fully compatible* with EL' if for all $EE_i \in EL$ there exists a $EE_j \in EL'$ such that EE_i is compatible with EE_j. EL *partially subsumes* EL' if there exist $EE_i \in EL$, $EE_j \in EL'$, such that EE_i subsumes EE_j. EL *totally subsumes* EL' if for all $EE_j \in EL'$ there exists a $EE_i \in EL$, such that EE_i subsumes EE_j. EL *totally includes* EL' if for all $EE_j \in EL'$ there exists a $EE_i \in EL$, such that EE_i includes EE_j.

Operations. To effectively compute e-labels and e-environments for *RB* objects, the following operations are required: (1) a disjunction $\vee$ between e-labels, (2) a conjunction-1 $\wedge_1$ between e-labels or between e-environments, (3) a modus-ponens $\otimes$ between rules or metarules and e-labels or e-environments, and (4) a conjunction-2 $\wedge_2$ between e-labels or between e-environments. These operations allow us to represent all potential steps that the RBES can perform in terms of e-labels and e-environments. They are defined in the following. The same symbol is used to indicate the same operation on e-labels or e-environments.

(1) Disjunction: models the different ways to conclude a fact. It is defined by,

$\vee: \mathfrak{EL} \times \mathfrak{EL} \rightarrow \mathfrak{EL}$

$EL \vee EL' = \{EE_i \mid EE_i \in EL \text{ or } EE_i \in EL'\}$

(2) Conjunction-1: models the computations performed at $lhs(x)$, $x \in R \cup MR$, when x is going to be fired. It is defined by,

$\wedge_1: \mathfrak{EL} \times \mathfrak{EL} \rightarrow \mathfrak{EL}$

$EL \wedge_1 EL' = \{EE_i \wedge_1 EE_j \mid EE_i \in EL, EE_j \in EL\}$

$SS_k = SS_i \cup SS_j$ if the union is consistent
$= empty$ otherwise

$RCV_k = [LCV_k, UCV_k]$ if $UCV_k > \tau$
$= empty$ otherwise

$RS_k = RS_i \cup RS_j$ if they are not m-exclusive
$= empty$ otherwise

where $LCV_k = cv\text{-}conjunction\ (LCV_i, LCV_j)$ and $UCV_k = cv\text{-}conjunction\ (UCV_i, UCV_j)$. When any of the parts is *empty*, the resulting environment is empty. Let $x \in R \cup MR$, assuming that $lhs(x)$ is the conjunction of the facts y and z, $EE_i\ (y) \wedge_1 EE_j(z)$ models the computation performed to check if $lhs(x)$ is satisfied: (a) if the union of their support sets is consistent, (b) if the *cv-conjunction* of their *cv*s is greater than the threshold, and (c) if their rule sequences are not m-exclusive.

(3) Modus ponens: models the computations performed when $x \in R \cup MR$ is fired, assuming $lhs(x)$ is satisfied. It is defined by,

$\otimes: \mathfrak{R} \times \mathfrak{EL} \rightarrow \mathfrak{EL}$ $\quad x \otimes EL = \{ x \otimes EE_i \mid EE_i \in EL\}$

$\otimes: \mathfrak{R} \times \mathfrak{EE} \rightarrow \mathfrak{EE}$ $\quad x \otimes EE_i = EE_k$, defined by

$SS_k = SS_i$

$RCV_k = [LCV_k, UCV_k]$

$RS_k = RS_i$ $\quad x \in MR$
$RS_i \cup \{x\}$ $\quad x \in R, x, RS_i$ are not m-exclusive
$empty$ $\quad x \in R, x, RS_i$ are m-exclusive

where $\mathfrak{R} = R \cup MR$, $LCV_k = cv\text{-}modus\text{-}ponens\ (LCV_i, cv(x))$, and $UCV_k = cv\text{-}modus\text{-}ponens\ (UCV_i, cv(x))$. If EE_i is an e-environment satisfying $lhs(x)$, $x \otimes EE_i$ models the action of firing x: (a) if $x \in R$ it should be not m-exclusive with the rule sequence of $lhs(x)$, and (b) if it is fired, the cv of $rhs(x)$ is obtained from the *cv*s of $lhs(x)$ and x itself, using the function *cv-modus-ponens*.

(4) Conjunction-2: models the relations that should exist between a rule r belonging to a module m and a metarule mr activating m, to allow r to be fired. It is defined by,

$\wedge_2: \mathfrak{EL} \times \mathfrak{EL} \rightarrow \mathfrak{EL}$

$EL \wedge_2 EL' = \{EE_i \wedge_2 EE_j \mid EE_i \in EL, EE_j \in EL\}$

$\wedge_2: \mathfrak{EE} \times \mathfrak{EE} \rightarrow \mathfrak{EE}$ $\quad EE_i \wedge_2 EE_j = EE_k$, defined by,

$SS_k = SS_i \cup SS_j$ if the union is consistent
$= empty$ otherwise

$RCV_k = RCV_i$

$RS_k = RS_i \cup RS_j$ if they are not m-exclusive
$= empty$ otherwise

$EE_i\ (r) \wedge_2 EE_j(mr)$ models the conditions for r to be fired: (a) the union of their support sets should be consistent and (b) their rule sequences should be not m-exclusive.

Computing e-labels. Using the operations defined above, the e-labels for the *RB* objects can be expressed in the following way,

$$f \in EF \quad EL(f)=\{EE0(f)\} \tag{1}$$

$$f \in DF \quad EL(f)= \bigvee_{r \in R,\, f=rhs(r)} EL(r) \tag{2}$$

$$r \in R,\ r \in m \quad EL(r)= r \otimes [\bigwedge_{1\ f \in F,\, f \in lhs(r)} EL(f)] \wedge_2 EL(m) \tag{3}$$

$$mr \in MR \quad EL(mr)=mr \otimes [\bigwedge_{1\ f \in F,\, f \in lhs(mr)} EL(f)] \tag{4}$$

$$m \in M \quad EL(m)= \bigvee_{mr \in MR,\, rhs(mr)=add\ m} EL(mr) \tag{5}$$

where $EE0(f)=<\{f\}$, [unknown, true], $\emptyset>$. The meaning of these expressions is straightforward. In (1) the e-label of an external fact f is composed by only one e-environment $EE0(f)$, the terminal environment, trivially built. In (2) the e-label of a deducible fact f is the disjunction of the e-labels of the rules concluding f. In (3) the e-label of a rule r is obtained in three steps: (a) conjunction-1 of the e-labels of facts appearing in $lhs(r)$, (b) modus ponens with r, and (c) conjunction-2 with the e-label of m, the module to which r belongs. The justification of each step is clear: step (a) requires $lhs(r)$ to be satisfied, computing the cv of the whole premise (that should be greater than τ), and checking that no m-exclusive rules have been used to deduce facts in $lhs(r)$, step (b) adds the rule r in the computation, including its cv and checking again r for mutual exclusion with those rules previously used, and step (c) establishes that r can only be fired when the explicit control has activated the module containing r. In expression (4) the e-label of a metarule is computed like the e-label of rule that is not included in any module. Finally, in (5) the e-label of a module m is the disjunction of the e-labels of the metarules activating m.

It is possible to discriminate, for each piece of information contained in an e-environment EE, whether it has been generated by the action of domain or control knowledge. According to this EE can be decomposed in an e-environment restricted to the domain $EE|_d$, and an e-environment restricted to the control $EE|_c$, related by the operator $\wedge_2$, $EE=EE|_d \wedge_2 EE|_c$. These components are quite adequate to check a number of verification issues, specially those in which a rule is tested against the metarules dealing with its module. Similar expressions to (1)-(5) are deduced for e-labels and e-environments restricted to domain knowledge (Meseguer 1991),

$$f \in EF \quad EL(f)|_d =\{EE0(f)\} \tag{6}$$

$$f \in DF \quad EL(f)|_d = \bigvee_{r \in R,\, f=rhs(r)} EL(r)|_d \tag{7}$$

$$r \in R,\ r \in m \quad EL(r)|_d = r \otimes [\bigwedge_{1\ f \in F,\, f \in lhs(r)} EL(f)|_d] \tag{8}$$

$$mr \in MR \quad EL(mr)|_d=mr \otimes [\bigwedge_{1\ f \in F,\, f \in lhs(mr)} EL(f)|_d] \tag{9}$$

$$m \in M \quad EL(m)|_d = \bigvee_{mr \in MR,\, rhs(mr)=add\ m} EL(mr)|_d \tag{10}$$

It is not possible to define e-labels restricted to control knowledge, and the most general expression for e-environments restricted to control knowledge is the following,

$$EE(x)|_c= \bigwedge_{2\ r_i \in RS(x)|_d, r_i \in m_i} EE(m_i),\ x \in F \cup R \cup MR \cup M \tag{11}$$

To compute all e-labels is enough computing all the e-labels restricted to domain knowledge (Meseguer 1991).

6 Verification Procedure

The verification issues can be reformulated in terms of relations between e-labels in the following way:

Inconsistency

I-1) Let $m \in M$, $mr, mr' \in MRM$, such that $rhs(mr)=add\ m$ and $rhs(mr')=remove\ m$. There exists inconsistency if $EL(mr')$ partially subsumes $EL(mr)$.

I-2) Let $f, f' \in F$, $ic \in IC$, $f, f' \in ic$. There exists inconsistency if there exist $EE_i(f) \in EL(f)$ and $EE_j(f') \in EL(f')$ such that they are compatible and ic is violated in $RCV_i(f)$ and $RCV_j(f')$.

I-3) Let $m \in M$, $mr \in MRM$, $r \in R$, such that $rhs(mr)=add\ m$ and $r \in m$. There exists inconsistency if $EL(mr)$ is not fully compatible with $EL(r)|_d$.

I-4) Let $m \in M$, $mr \in MRM$, $r \in R$, such that $rhs(mr)=remove\ m$ and $r \in m$. There exists inconsistency if $EL(r)|_d$ partially subsumes $EL(mr)$.

Redundancy

R-1) Let $x, x' \in MR \cup R$, such that $rhs(x)=rhs(x')$. x' is redundant with x if $EL(x)$ totally includes $EL(x')$.

R-2) Let $f, f' \in DF$, f' is redundant with f if $EL(f)$ totally includes $EL(f')$.

R-3) Let $m \in M$, $mr \in MRM$, $r \in R$, such that $r \in m$ and $rhs(mr)=add\ m$. r is redundant with mr if $EL(r)|_d$ totally subsumes $EL(mr)$.

Circularity

C-1) Let $r \in R$, $m \in M$, $r \in m$, there is a cycle if there exists $EE_i(r)|_d \in EL(r)|_d$, such that $r \in RS_i(r)|_d$.

C-2) Let $r \in R$, $m \in M$, $r \in m$, $EE_i(r)|_d \in EL(r)|_d$. Let $m_1,...,m_k$ the modules to which rules in $RS_i(r)|_d$ belong. There is a cycle if there exists $EE_l(m_j)|_d \in EL(m_j)|_d$ such that $r \in RS_l(m_j)|_d$, $j=1,...,k$.

C-3) Let $m \in M$, $EE_i(m)|_d \in EL(m)|_d$. Let $m_1,...,m_k$ the modules to which rules in $RS_i(m)|_d$ belong. There is a cycle if there exists $EE_l(m_j)|_d \in EL(m_j)|_d$ and $r \in m$, such that $r \in RS_l(m_j)|_d$, $j=1,...,k$.

Useless *RB* objects

U-1) Let $x \in R \cup MR$, x is non-fireable if $EL(x)=\emptyset$.

U-2) Let $f \in F$, f is unreachable if $EL(f)=\emptyset$.

U-3) Let $mr \in MRS$, $mr' \in MRM$, mr' is shadowed by mr if $EL(mr)$ totally subsumes $EL(mr')$.

U-4) Let $mr \in MRS$, $r \in R$, r is shadowed by mr if if $EL(mr)$ totally subsumes $EL(r)$.

U-5) Let $m \in M$, $mr \in MRM$, $rhs(mr)=add\ m$, m is unreachable if every metarule mr is either non-fireable or shadowed.
U-6) Let $m \in M$, $r \in R$, $r \in m$, r is unreachable if m is unreachable.
U-7) Let $f \in F$, $r \in R$, $f \in rhs(r)$, f is unreachable if every r is either non-fireable, shadowed or unreachable.

So, a verification procedure can be built up with the following steps: (i) compute for every *RB* object their e-labels restricted to the domain knowledge, detecting all circularities of type 1 (otherwise the procedure could loop itself), (ii) detect all useless RB objects, removing them for further processing, (iii) check circularities, cases 2 and 3, (iv) check inconsistency, and (v) check redundancy. This procedure has been implemented in Common Lisp, and their results testing three rule bases developed with the shell MILORD (Sierra 1989) on a SUN-4/260, are contained in the Table 1.

	#M	#R	#MR	#e-environments	memory (kb)	CPU time (min)
1	1	122	0	144	64	1
2	3	177	52	502	192	3
3	7	344	59	41,235	7,424	1,225

Table 1. Experimental Results

These results show how the complexity increases with the *RB* size, specially when metarules contain in their left-hand sides goals of modules (third rule base). It is worth noticing that, once a rule base has been verified, this procedure can be used to incrementally verify changes on it. To do that, the step (i) in the verification procedure has to be repeated, but steps (ii) to (v) have to be performed on changed *RB* objects only. Computing e-labels is not very expensive computationally, it requires 1% of the total CPU time (on average over the three rule bases). The computational complexity of this procedure is, in the worst case, exponential. By worst case it is understood that all the possible combinations of values and *cv*s for facts are present in the rule base. But this is quite far from reality, so it may be reasonable to expect an average case complexity that allows one to effectively check a rule base of several hundreds rules and one hundred metarules.

7 Conclusions

From this work three main conclusions can be extracted. First, it is clear that current RBESs present new verification problems that available verifiers are not able to deal with. These problems can be very serious and require mandatory checking. Second, previous constructs, labels and environments, can be successfully extended including more information to solve these problems. And third, an effective testing of these problems can be performed in a reasonable amount of time and space for medium size rule bases.

Acknowledgements. I thank Ramón López de Mántaras and Lluis Godo for reading a previous version of this paper and providing many useful comments. I also thank the anonimus reviewers of this paper for their useful criticisms.

References

Bareiss, R.; Porter, B.W.; and Murray, K.S. 1989. Supporting Start-to-Finish Development of Knowledge Bases. *Machine Learning* 4:259-283.

Cragun, B.J.; and Steudel, H.J. 1987. A Decision-Table-Based Processor for Checking Completeness and Consistency in Rule-Based Expert Systems. *International Journal of Man-Machine Studies* vol 26:633-648.

deKleer, J. 1986. An Assumption-based TMS. *Artificial Intelligence* 28:127-162.

Ginsberg A. 1988. Knowledge-Base Reduction: A New Approach to Checking Knowledge Bases for Inconsistency and Redundancy. In Proceedings of the Seventh National Conference on Artificial Intelligence, 585-589. St Paul, Min: American Association for Artificial Intelligence.

Meseguer, P. 1990. A New Method to Checking Rule Bases for Inconsistency: A Petri Net Approach. In Proceedings of 9th European Conference on Artificial Intelligence, 437-442. Stockholm, Sweden: European Coordinating Committee for Artificial Intelligence.

Meseguer, P. 1991.Verification of Multi-Level Rule-Based Expert Systems. Research Report, IIIA-91/10, CEAB.

Nguyen, T.A.; Pekins, W.A.; Laffey, T.J.; and Pecora, D. 1985. Checking an Expert System Knowledge Base for Consistency and Completeness. In Proceedings of the Ninth International Joint Conference on Artificial Intelligence, 375-378. Los Angeles, Cal.: International Joint Conferences on Artificial Intelligence, Inc.

Rousset, M.C. 1988. On the Consistency of Knowledge Bases: the Covadis System. In Proceedings of the 8th European Conference on Artificial Intelligence, 79-84. Munich, Germany: European Coordinating Committee for Artificial Intelligence.

Sierra, C. 1989. MILORD: Arquitectura multinivell per a sistemes experts en classificacio. PhD diss., Universitat Politecnica de Catalunya.

Suwa, M.; Scott, A.C.; and Shortliffe, E.H. An Approach to Verifying Completeness and Consistency in Rule-Based Expert Systems. *AI Magazine*, 3(4):16-21.