Dynamic Verification of an Object-Rule Knowledge Base Using Colored Petri Nets

Chakib Tadj
École de Technologie Supérieure - Département de Génie Électrique
1100, Notre-Dame Ouest, Montreal, Qc, H3C 1K3 Canada

and

Toufik Laroussi
École de Technologie Supérieure - Département de Génie Électrique
1100, Notre-Dame Ouest, Montreal, Qc, H3C 1K3 Canada

ABSTRACT

In this paper, we propose a formal description for the dynamic verification of an Object-Rule Hybrid Knowledge-based System (HKBS), capitalizing on the work carried out within the verification framework of Frame-Rule Hybrid Expert Systems. The main idea is to model an HKBS by means of a Colored Petri Network (CPN). In this way, method invocations, state class changes, rules and productions will be modeled as components of the CPN. Detection and analysis of the HKBS will be carried out by the construction and analysis of the markings graph, which results from the inference process.

Keywords: Dynamic Verification, Object-Rule Knowledge, Colored Petri Nets.

1. INTRODUCTION

Hybrid Expert Systems (HES) technology is a field in the artificial intelligence domain which has moved successfully from the research laboratory to the commercial or industrial application [1,3].
Generally speaking, HES integrates human expertise in computer programs to enable those programs to execute tasks which normally require the intervention of a human expert. There is a series of problems and difficulties underlying the development of an HES or Hybrid Knowledge-based System (HKBS), ranging from the acquisition of knowledge to the representation of that knowledge, including temporal reasoning, uncertainty reasoning, combinatorial explosion, conflict resolution and other similar problems [4-7].
Continuous maintenance is necessary in order to ensure that the system remains efficient, and hence the importance of validation and verification of HES [8].
We have found that little research has been done on the verification of Object-Rule HKBS. The reasons for this are as follows:

- most HKBS developed since the 1980s [6] have been of the Frame-Rule type, and the integration of the object-oriented technique into the HKBS has occurred only recently. Objects are used to represent facts, and rules are used to represent deductive knowledge (the conditions and actions of the rules are formulated using class attributes and approaches);
- what little research has been done on Object-Rule HKBS verification has dealt with the static aspects, that is, verification of the Knowledge Base without rule-firing [2].

Dynamic verification approaches developed for Frame-Rule HES [1, 9-12] do not apply to Object-Rule HES because the facts and rules in frames are specified in a hierarchical model in order to ensure the transmission of attributes from the higher levels to the lower levels of the hierarchy. These approaches do not integrate the presentation power of object classes, facts and reasoning on these objects by means of method invocation. Moreover, since our HKBS is non-monotonic, there may be additions of, modifications to and/or withdrawals of objects through their actions, contrary to the rules expressed by the frames.
The aim of our research is to propose a solution based on a formal approach for the dynamic verification of a non-monotonic Object-Rule HKBS. The main features of the proposed solution are:

- a solution based on the analysis power of Colored Petri Networks (CPN) [13,14];
- inspiration from the solutions proposed for the dynamic verification of Frame-Rule HES in [1, 12];
- modeling definition sets;
- formal proposition sets for error and anomaly verification in the Object-Rule HKBS.

This approach also has:

- the capacity to build the accessibility graph for rules on the basis of the dynamic execution of our rules by the JRules inference engine;
- the capacity to build the occurrence sequences while respecting the formalism defined in [14];
- potential for the analysis of the anomaly situations on the basis of well-defined formal propositions.

The structure of this paper is as follows: In section 2, we present an overview of related work. Sections 3 and 4 present our main contributions. The formal aspects of Object-Rule HKBS modeling by a CPN are defined first, followed by a proposal of a formal solution using the power of marking graph analysis for the verification of each anomaly in the Knowledge Base. The construction of our graph and/or Petri network is performed dynamically. In the final section, we discuss the results obtained during simulation of the proposed examples, and present the conclusions of our work.

2. RELATED WORK

A Knowledge-based System is a category of programs capable of solving problems which do not have a classical algorithmic solution. It is designed to help the user perform a task in an

application area; the system works from knowledge acquired from the area expert.
For the last 20 years, the Validation and Verification (V&V) sectors of expert systems have undergone considerable improvement. Indeed, during all the years from 1988 to 1992, the AAAI (American Association for Artificial Intelligence) organized workshops on the verification, validation and testing of intelligent systems.
The IJCAI (International Joint Conference on Artificial Intelligence) has organized workshops on V&V since 1989. In addition, the European Conference on Artificial Intelligence (ECAI) has held a number of workshops on V&V.
The need to evaluate the large number of expert systems that have been developed since the mid-1980s have created a special interest in the verification of expert systems. The role and importance of such verification are well documented in [5, 8, 15-18].
The research carried out on the verification of the anomalies in systems based on the Frame-Rule has been considerable in comparison with the research carried out on anomalies in an Object-Rule HKBS, where little work has been done. If the earliest Knowledge-based Systems (KBS) in general, and more particularly the HKBS, were, in the majority of cases, verified using a static verification approach; dynamic verification proved to be efficient in the cases where it was used (INDE, SACCO, etc.). Indeed, in the dynamic verification of a Knowledge Base (KB), we use the deductive power of the Rule Base (RB), which allows us to proceed to a more in-depth verification of the RB.

The research closest to ours is that of Shiu et al. [1, 10-12], which involves the verification of systems which are linked by the production of hierarchical Frame rules. Their approach does not allow the use of method invocations in the condition or action clauses of a rule. They use two types of tokens, the first being the "State Token", which registers the class predicate and information state, and the second being the "Object Instance Token", which represents the instance of a particular object in a particular class in the object hierarchy. That approach is used in a monotonic context, which limits the usability of this approach in the Object-Rule context in a non-monotonic environment.

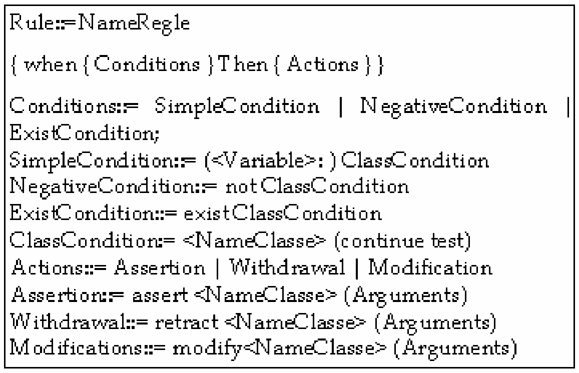
3. PRINCIPLES OF THE PROPOSED SOLUTION

The HES combines several representation paradigms in a single integrated environment. It is made up of the following features:

- object classes encapsulating the knowledge model – the inheritance relationship describes how these knowledge modules interact;
- rules specifying the functional behavior of objects in the expert system – these functions are presented by method invocations;
- a reasoning strategy controlling and specifying the inference sequence of knowledge in the expert system.

The use of Petri networks, as shown in Table 1, should allow us to represent classes, rules, conditions, actions and objects, as well as their states. To allow this, we use the properties of CPN to model the HKBS ALCAN [2]. In this section, we will discuss the JRules language that is used in the RB, and then define the modeling of our hybrid system by a CPN and the different types of rule presentation by the CPN.

In the Rule-Object context, a rule can be written using the following syntax:


```
Rule::=NameRegle
{ when { Conditions }Then { Actions } }
Conditions::= SimpleCondition | NegativeCondition |
ExistCondition;
SimpleCondition::= (<Variable>: ) ClassCondition
NegativeCondition::= not ClassCondition
ExistCondition::= exist ClassCondition
ClassCondition:= <NameClasse> (continue test)
Actions::= Assertion | Withdrawal | Modification
Assertion::= assert <NameClasse> (Arguments)
Withdrawal::= retract <NameClasse> (Arguments)
Modifications::= modify<NameClasse> (Arguments)
```

A simple condition makes it possible to find all the objects of the class that pass the test where a negative condition is true when no object of the class verifies the condition. A "condition that exists" is true when there is at least one object of the class which verifies the condition. The conditions are defined in "continue test". These may include tests on class attributes or class method invocation results. The action "assert" adds an object of a given class with its necessary parameters (Arguments). In the same manner, the action "withdrawal" deletes an object referenced by a variable defined in one of the conditions of the rule, and the action "modify" modifies an object referenced by a variable in one of the conditions of the rule. In this approach, the reasoning is object-oriented. In fact, at the beginning, the work memory is initialized by objects called "initial facts". During a problem-solving session, the inference engine uses the rules, and objects are then added, deleted or modified according to the rules invoked. It should be noted that the conditions and the actions are method invocations on object classes, and that the class attributes are private and thus not accessible by JRules.

HES		CPN Modeling
Object Part	Object Classes States Classes	Places
		Example: P={Tank, Simulation, Site, Vain} TankState={Empty,Full, Normal} SiteState={InProd,NormalProd,OverProd}
	Object Instances	Tokens
	ObjectValue State	Function return the state instance object
Rules Part	Conditions	Expression on the arc
	Actions	Expression on the arc
	Rules	Transitions
	Facts	Reachable variables

Table 1. HKS ALCAN modeling with CPN.

3.1 Formal Description of Modeling

The formal definition of CPN has been published elsewhere [14]. It is reproduced here in order to highlight the extension of our formalization. We define our model as follows:

DEFINITION (1): The CPN (R) that models the Rule-Object-based HKS is a tuple:

R = <ψ, C, Expr, P, T, A, N, φ, E, I, σ, StateObject>

where:
ψ = {$w_1, w_2, \ldots, w_i$ } is a finite set of non-empty types, called color sets.

C = {$C_1, C_2, \ldots, C_n$} is a finite set of objects classes.

Expr: A →expression, is an arc expression function. It is defined from A into expression such that:

$$\forall \ a \in A : [Type(Expr(a)) = \varphi(p(a))_{ms} \ \wedge \ Type(Var(Expr(a))) \subseteq \psi]$$

with Type (Expr(a)) is expression type of arc a, and Var(Expr(a)) is a set of expressions of arc a.
P = {$P_1, P_2, \ldots, P_n$},n = 1,…,|E| is a finite set of places of object classes.

T = {$T_1, T_2, \ldots, T_l$} is a finite set of transitions (rules).

A = {$a_1, a_2, \ldots, a_k$} is a finite set of arcs.

N: A → P x T ∪ T x P is a node function. It is defined from A into expressions such that: P ∩ T = P ∩ A = T ∩ A = ∅.

φ: P → ψ is a color function. It is defined from P into ψ, maps each place p to a color φ(p).

E = {Ec1, Ec2,..Eci,…,Ecn} is a finite set of states object classes such that:
Ec_i = {$E_1, E_2, E_3, \ldots, E_k$} is a finite set of states object class C_i.

I: P → expression is a initialization function. It is defined from P into expressions such that:

$$\forall \ p \in P : [Type(I(p)) = \varphi(p)_{ms}]$$

σ: is a set of occurrence sequence.

StateObject: C → E is a state function. It is defined from State object into expressions. It maps for each object class its states, such that:

$$\left[O_i : OBJECT, \ O_i \in C_i \Rightarrow Type\left(Var(StateObject(O_i))\right) \subseteq \psi\right)\right]$$

Compared to the classical Petri Net defined in Jensen [14], we have introduced: C, σ, E, StateObject in order to model the Rule-Object- based HKS. Definitions of CPN found in [14], which have been used here without any change, are as follows:

- *Binding*

DEFINITION (2): A binding of a transition t is a Boolean function b defined on Var(t) such that:

$$\forall \ v \in Var(t) \ : \ b(v) \in Type(v)$$

where Var(t) is a set of transition variables and B(t) is the set of all bindings for t.

- *Marking*

DEFINITION (3): A token element is a pair (p,c): where p∈P and c∈φ(p), while a binding element is (t,b,e) where: t∈T, b∈B(T) and e∈S(t). The set of all token elements is denoted by TE and the set of all binding elements is denoted by BE.

DEFINITION (4): A marking M is a multi-set over TE, while a step is a non-empty and finite multi-set over BE. The initial marking M_0 is the marking that is obtained by evaluating the initial expression:

$$\forall \ (p,t) \in TE : M_0(p,c) = I(p)(c)$$

- *Occurrence Sequence*

DEFINITION (5): A finite occurrences sequence is a sequence of markings and steps:
$M_0[t_1>M_2[t_2>M_3 \ldots M_{n-1}[t_n>M_n$ such that: n ∈ N, M_0 is the initial marking and M_n is the final marking.

- *Reachable Transition*

DEFINITION (6): A marking M_n is reachable from a marking M_0 if and only if (iff) there exists a finite occurrence sequence σ having M_0 as start marking and M_n as final marking. We denote by $t_1t_2t_3 \ldots t_n$, the sequence of steps such that: $M_0[t_1>M_2[t_2>M_3 \ldots M_{n-1}[t_n>M_n$, n ∈ N and σ($t_1, t_2, \ldots t_n$) represents the set of markings that is reachable from M_0 and denoted by: [M_0>.

- *Transition Quasi-Alive*

DEFINITION (7): A transition t is quasi-alive iff there exists an occurrence sequence σ, such that t is reachable from M_0.

- *Reachable Propriety*

DEFINITION (8): The marking M_j is included in marking M_i, iff for all place p: $M_j(p) \leq M_i(p)$.

Formal definition of CPN which has been adapted from [14] in the Rule-Object-based HKS context:

- *Token*

DEFINITION (9): We defined the number of tokens in place p by: M(p) = ∑ | O_i |, i >0, such that O_i is an instance object class in place p.

- *Markings Update*

DEFINITION (10): when a transition <t> is enabled, we have the following:

1. If A_s is produced, the marking M_1 changes without producing another marking M_2. We denote that by:
$$\forall \ p \in P : M_1(p) = M_1(p) - \sum_{(t,b) \in Y} E(p,t) < b >$$
2. If A_c is produced, the marking M_1 changes to produce another marking M_2. We denote that by:
$$\forall \ p \in P : M_2(p) = (M_1(p) - \sum_{(t,b) \in Y} E(p,t) < b > + \sum_{(t,b) \in T} E(t,p) < b >)$$
3. If A_m is produced, the value of the instance object (token) in marking M_1 is updated.

- *Equality of Markings*

DEFINITION (11): Tow markings are equal iff their places are equal one to one.

- *Graph of Markings*

DEFINITION (12): The occurrence graph in CPN is a tuple OGCPN = (V,A, E, N) satisfying the following requirements:
1. "V" is a set of reachable markings from M_0, denoted by:
   V = [M_0>;

2. "A" is a set of finite arcs, denoted by:
   A = { (M_1,b,e,M_2) ∈ V x BE x SE x V | M_1[b>M_2 and e ∈ SE};

3. "N” is a set of nodes, such as:
a = (M1,b,e,M2) ∈ A and N(a) = (M1,M2).

DEFINITION (13): A change of state is a Boolean function B. This function returns TRUE if the instance state object O_i has changed while the transition t was fired. We note that:

$$[B(StateObject(O_i)\ ,\ t) \subseteq \psi]$$

We also denote by S(t) the state of binding t.

The formal definition of CPN that has been introduced to complete modeling of the Rule-Object-based HKS:

- *State of Object Class*

DEFINITION (14): To evaluate the state of an object class, we define three functions:

A_s: $T_c \rightarrow \{\varnothing\}$ is an output class function used when the object instance is removed.

Ac: $Tc \rightarrow Pc$ is an output class function used for modification with a change of state of the object class.

A_m: $T_c \rightarrow P_c$ is an input class function used for modification without a change of state of the object class.

- *Equalities of Objects*

DEFINITION (15): Two places are equal iff:
1. They have the same number of tokens (objects);
2. The tokens are equal one to one; and
3. They have the same state-of-object class.

DEFINITION (16): Two objects (instances) are equal iff:
1. They have the same object identification (ID) (each object instance has a unique ID); and
2. They have the same color.

- *Condition to Firing Rule*

DEFINITION (17): We can fire the same rule iff: after execution of actions in the rule, the attributes of the instance object have changed their values (that is, a JRule reasoning strategy).

3.2 Presentation of Certain Rules in the CPN

A color domain is associated with each place, and each token in place is colored by an element of the domain of that place (several tokens may have the same color). The inscription of a place is thus a multi-set of colors, a set in which an element may occur several times.

A transition is enabled if and only if each entry place to the transition contains a sufficient number of tokens for each color of the place domain. Independently of the evaluation of the colored functions, a transition may not be allowed if its attached expression does not satisfy some of the attributes.

In order to differentiate between the instance objects while comparing markings, we use the object ID as a (unique) identification key.

4. USING A CPN FOR THE ANALYSIS OF ANOMALIES IN AN OBJECT-RULE HKBS

There are two analysis approaches for researching anomalies in a HKBS. The first is based on marking graph analysis, the second on the analysis of the final places of the Petri network.

In this paper, we present the first technique. In this approach, anomaly analysis is based on an analysis of the marking graph or occurrence graphs that represent the accessibility game of the CPN [13, 14].

The idea is to build a graph containing a node for each accessible marking and an arc for each appearance of the marking element. Indeed, we concentrate our analysis by allowing a specific transition (that is, one which corresponds to a few significant initial facts) and verifying next, at a given time, the aggregate of markings.

The problem may then be localized by an analysis of the trace of the transition sequences that may supply the alternative or multiple markings effects.

4.1 Heuristic CPN Marking Graph Construction Algorithm

We propose a heuristic search method to construct the occurrence graph for particular marking. The search for anomalies in HKBS needs an adequate initialization of sequence of transition (
Table 2, steps 1,2, 3, 4) and construction the reachibility three (steps 6 to 15).

The strategy used is to start from position “i” and put it initially in the CONFLIT_LIST the list of candidate rules (transitions) to be fired by JRules. Put in CANDIDATE_RULE the rule that JRule fires and general marking Mi which result from firing CANDIDATE_RULE if this marking M_i is not an element in MARKING_GRAPH then add Node(M_i) and arc(M_i, R_i, M_{i+1}) in MARKING_GRAPH and the new position to be prepared in MARKING_GRAPH is i=i+1 else add only arc (M_i,t,M_i) in the MARKING_GRAPH. Remove the CANDIDATE_RULE from the LIST_CONDIDATE, if LIST_CONDIDATE is not empty save the context (J=J+1), and put first rule in CANDIDATE_LIST. The process is repeated until CANDIDATE_LIST is empty.

```
MarkingConstrucGraph MCG{
1.  Construct the initial marking $M_0$
2.  Initialize variables I=1 and J=1
3.  Add ($M_0$) to the occurrence marking graph
    MARKING_GRAPH
4.  CONFLIT_ LIST = list of candidate rules
5.  If CONFLIT_ LIST = $\varnothing$ | stop by user | number nodes
    reached Then
      Goto (13)
6.  CANDIDATE_RULE = rule fire by JRule
7.  Fire the rule CANDIDATE_RULE
8.  Construct the actual marking into $M_{I+1}$
9.  IF Verify ($M_{I+1}$ , Node(MARKING_GRAPH)) Then
      Create ARC ($M_I$, , CANDIDATE_RULE, $M_I$)
    Else
      Add $M_{I+1}$ into the occurrence graph MARKING_GRAPH
      Create ARC ( $M_I$,CANDIDATE_RULE,$M_{I+1}$)
      I = I + 1
    Endif

10. Update CONFLIT_LIST
11. Save context of marking $M_I$
12. IF CONFLIT_LIST ≠ $\varnothing$ Then
    J = J +1
    CANDIDATE_RULE = the first element in CONFLIT_LIST
    Goto (7)
  Endif

13. Restore context $M_{I-1}$
14. J = J – 1
15.  If J >=0 Then
        Goto (4)
     Endif
}End
```

```
Verify (Marking Mx , My) {
1.   for each  object class Ci in Mx
2.   If Exist (Ci, My) and (State(Ci, My) ≠ State(Ci, Mx))
          Then return False
     Else
          Return True
     Endif
}End
```

Table 2. CPN marking graph construction algorithm.

4.2 Types of Anomalies

Our research is a contribution to the study of the accuracy of a Knowledge Base. This includes completeness (the problems of redundancy and subsumption), consistency (the problems of incoherence or contradiction, and useless premises) and perfection (the problem of unreachable rules) of a Knowledge Base.

We describe below a set of the most frequently treated anomalies in verification currently; some of these anomalies, such as contradiction, are deemed serious since they cause incorrect functioning of the KB.

a) Proposition of Conflict

In an HKS, two rules R_i and R_j are in conflict (Figure 1) iff:

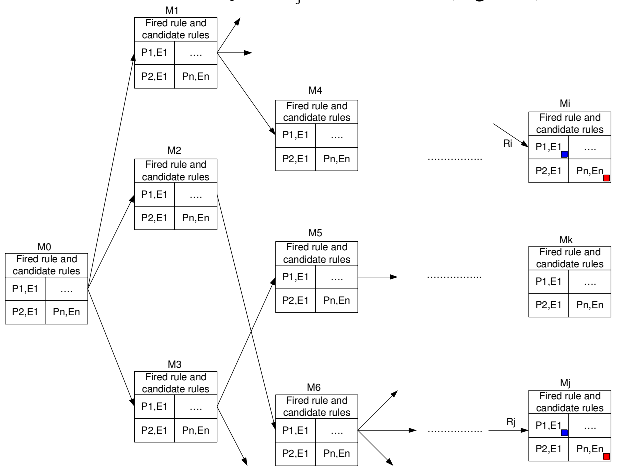


Figure 1. Conflict situation in a Rule-Object HKS.

- there exist two final sequences σ_i and σ_j resulting from final markings M_i and M_j respectively, such that M_i is produced by firing R_i, noted $M_i = \delta(M_0,\sigma_i)$ and M_j is produced by firing R_j and noted $M_j = \delta(M_0, \sigma_j)$;
- they do not have the same path: $\sigma_i \cap \sigma_j = \varnothing$;
- there exists a set of places $\Phi \neq \varnothing$, in which a place in M_i and M_j have the same color, and is noted by:

$$\Phi = \{P_s: P, M_i(P_s, \varphi(P_s)) = M_j(P_s, \varphi(P_s)), s>0\}.$$

b) Proposition of Redundancy

In an HKS, two rules, R_i and R_j, cause redundancy between object classes iff:

1. R_i and R_j are in conflict (i), (ii) and (iii); and
2. For any place Ps in the marking M_i, the color of the place φ(Ps) in M_i is equal to the color of place Ps in M_j. We denote:

$$\{\forall P_s \in \Phi,\ M_i(P_s, \varphi(P_s)) = M_j(P_s, \varphi(P_s)),\ s>0\} \Leftrightarrow [M_i = M_j]$$

As an example, we consider the following rules written in a simplified notation of JRules.

```
rule R1 { when { obj1: Tank ( obj1.getCode() != "RCP");
     obj1.getPercentageFull() > 90; );
     obj2: Site ( obj2.getCode()== "CP"));
  } then  { modify obj1{
ojb1.setDischargeRiskShortTime(100); };  } };
```

```
rule R2 { when  { obj1: Tank ( obj1.getPercentageFull()>90);
    obj2: Site (  obj2.getCode() = = "CP");
   } then  { modify obj1{
obj1.setDischargeRiskShortTime(100);};
              modify ?st{ obj2.setMiddleRisk()+5;} } }};
```

Figure 2 shows the markings after a complete simulation of the firing rules in the above example.

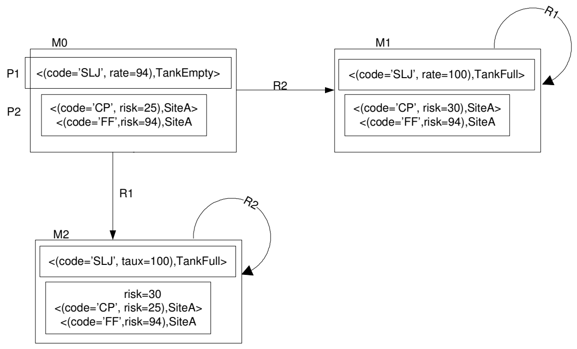

M0
M1
M2
P1
P2
<(code='SLJ', rate=94),TankEmpty>
<(code='CP', risk=25),SiteA>
<(code='FF',risk=94),SiteA
R2
R1
<(code='SLJ', rate=100),TankFull>
<(code='CP', risk=30),SiteA>
<(code='FF',risk=94),SiteA
R1
R2
<(code='SLJ', taux=100),TankFull>
risk=30
<(code='CP', risk=25),SiteA>
<(code='FF',risk=94),SiteA

Figure 2. Marking graph - redundant rule.

Analysis of the above graph shows that R_1 and R_2 are redundant. Indeed:

- there exists a final sequence $\sigma_1 = R_1$ resulting from the marking sequence $M_0[R_1>M_2$, denoted by $M' = \delta(M_0, \sigma_1)$;
- there exists a final sequence $\sigma_2 = R_2$ resulting from the marking sequence $M_0[R_2>M_1$, denoted by $M'' = \delta(M_0, \sigma_2)$;
- we have $\sigma_1 \cap \sigma_2 = \varnothing$;
- there exists a set $\Phi = \{P_1, P_2\}$, for which the marking $M_1(P_1, \varphi(P_1)) = M_2(P_1, \varphi(P_1))$ and the marking $M_1(P_2, \varphi(P_2)) = M_2(P_2, \varphi(P_2))$;
- the set Φ is equal to the set P of places, therefore $[M_0 = M_1]$.

c) Proposition of subsumption

A rule R_j is subsumed by rule R_i iff:

1. R_i and R_j are in conflict (i), (ii) and (iii);
2. there exists a set E, such that $E \neq \varnothing$, where

$$E = [Ps : P / Ps \notin \Phi] \Rightarrow [\Phi \cap E = \varnothing]$$

3. for any place belonging to E, we have:
   a) the number of tokens of P_s in M_i is the same as in M_j, denoted by:
   $$\forall P_s \in E \Rightarrow |M_i(P_s, \varphi(P_s))| = |M_j(P_s, \varphi(P_s))|$$
   b) for any place P_s in M_i, its state is the same as in M_j, denoted by:
   $$\forall P_s \in E \Rightarrow State(P_s)_{M_i} = State(P_s)_{Mj}$$
4. there exists a set of transitions ΩM_i for which R_i does not change its state while firing these last transitions, and/or there exists a set of transitions ΩM_j for which R_j does not change state while firing these transitions, and then: $\Omega M_i \neq \varnothing$ and/or $\Omega M_j \neq \varnothing$;
5. if $[R_i \subseteq \Omega M_j]$, we say that R_i is subsumed by R_j, i.e. the set of consequents of R_i includes the same consequents as R_j, and the set of premises of R_i is the same as for R_j; and
6. if $[R_j \subseteq \Omega M_i]$, we say that R_j is subsumed by R_i, i.e. the set of consequents of Rj includes the same consequents as R_i, and the set of premises of R_j is the same as for R_i.

A problem of subsumption between two rules is encountered if, starting from the same initial marking M_0, we construct two different occurrence sequences, leading to two markings M_i and M_j, one of which is included in the other (if $M_i = M_j$, we find the redundancy case). Note that a subsumption anomaly is a special case of a redundancy.

As an example, we consider the following rules written in a simplified notation of JRules.

```
rule S1
{ When { obj1: Simulation(); obj2: Tank (
obj2.getCode().equals("RCP");
obj2.getPercentageFull()<=95 ;
obj1.getDateStep()<="3103"; obj1.getDateStep()>="0101"
); obj3: Site ( obj3.getCode().equals("CP"));
} Then { modify obj3 { obj3.setMiddleRisk(100);};
          modify
obj2.setRiskLevelDischargeShortTime(4)); }};
```

```
rule S2
{  When { obj1: Simulation();  obj2: Tank (
obj2.getCode().equals("RCP"));
 obj3: Site ( obj3.getCode().equals("CP") );} Then {modify
obj3 { obj3.setMiddleRisk(100);}; modify
obj2{obj2.setLevelRiskDischargeShortTime(4);};
```

Figure 3 shows the markings after a complete simulation of firing rules of the above example.

We note that:

1. S_1 and S_2 are in conflict:
   a) if there exist two final markings M_1 and M_2, such that M_1 is reachable from $M_0[>S_1[>M_1$ and denoted by $\sigma_1 = \delta(M_0, \sigma_1)$, and M_2 is reachable from $M_0[>S_2[>M_2$ and denoted by $\sigma_2 = \delta(M_0, \sigma_2)$,
   b) $\delta_1 \cap \delta_2 = \varnothing$,
   c) if there exists $\Phi = \{P_2,P_3\}$ such that $M_1(P_2, \varphi(P_2)) = M_2(P_2, \varphi(P_2))$;
2. The color in place P_3 is the same in M_1 and M_2 and noted by $\Phi = \{P_3\}$ such that $M_1(P_3, \varphi(P_3)) = M_2(P_3, \varphi(P_3))$;
3. There exists a non-empty set $E = [P_1: P / P_1 \notin \Phi]$ for which we have:
   a) $|M_1 (P_1, \varphi(P_1))| = |M_2 (P_1, \varphi(P_1))|$,
   b) StateObject(P_1,M_1) = StateObject(P_1, M_2) = TankFull;
4. There exists $\Omega_{M1} = \{S_2\}$, which implies that S_1 is subsumed by S_2. The marking M_1 is included in M_2. Indeed, S_1 and S_2 work on the same variables (methods) and they lead to the same states P_1 (Tank 'RCP' Full) and same states P_2 (Site 'CP' InProd) in M_1 and M_2.

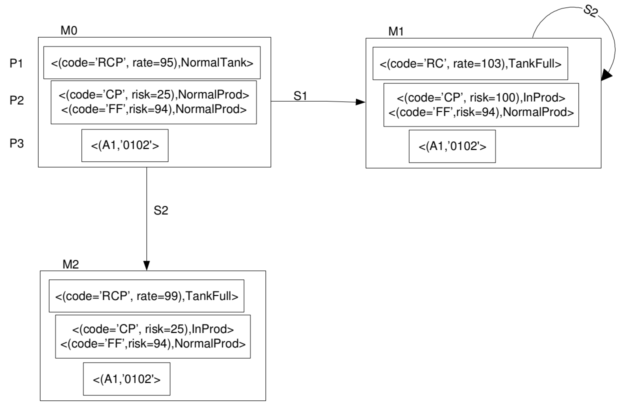

M0
P1
P2
P3
<(code='RCP', rate=95),NormalTank>
<(code='CP', risk=25),NormalProd>
<(code='FF',risk=94),NormalProd>
<(A1,'0102'>
S1
S2
M1
<(code='RC', rate=103),TankFull>
<(code='CP', risk=100),InProd>
<(code='FF',risk=94),NormalProd>
<(A1,'0102'>
S2
M2
<(code='RCP', rate=99),TankFull>
<(code='CP', risk=25),InProd>
<(code='FF',risk=94),NormalProd>
<(A1,'0102'>

Figure 3. Marking graph - subsumed rule.

d) Proposition of contradiction

An HES has contradictory rules R_i and R_j iff:

1. R_i and R_j are in conflict (i), (ii) and (iii);
2. There exists a set $E \neq \varnothing$ containing all the places that do not belong to Φ, i.e. :
$$E = [P_s : P / P_s \notin \Phi] \Rightarrow [\Phi \cap E = \varnothing]$$
3. For any place in E ($\forall$ $P_s \in$ E), all objects instances in place P_s of M_i, are identical to the object instances of P_s in M_j, and denoted by $O_i(P_s,M_i) = O_i(P_s, M_j)$. Moreover, the state of place P_s of object O_i in M_i is different from the state of this object in M_j. We denote this by: State(O_i, (P_s, M_i)) $\neq$ State(O_i, (P_s, M_j)).

As an example, we consider the following rules written in a simplified notation of JRules.

```
rule C1
{ When  { obj1: Tank ( obj1.getCode().equals("RCP");
obj1.getPercentageFull()>90); obj2: Site(
obj2.getCode().equals("CP"));
}  Then { modify obj2 { obj2.setMiddleRisk(100);};};
```

```
rule C2
{  When { obj1: Tank (  obj1.getCode().equals("RCP");
obj1.getPercentageFull()>90);  obj2: Site (
obj2.getCode().equals("CP"));
} Then {  modify obj2 {  obj2.setMiddleRisk(90);}};
```

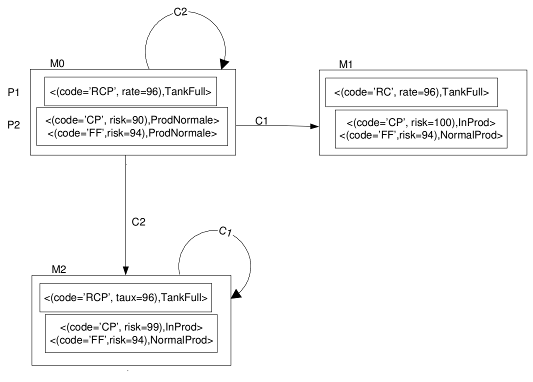
Figure 4 shows the markings after a complete simulation of firing rules in the above example.

We note that:

1. C_1 and C_2 are in conflict:
   a) if there exist two final markings M_1 and M_2, such that M_1 is reachable from $M_0[>C_1[>M_1[>C_2[>M_3$ and denoted by $\sigma_1 = \delta(M_0, \sigma_1)$, and M_2 is reachable from $M_0[>C_1[>M_2$ and denoted by $\sigma_2 = \delta(M_0, \sigma_2)$,
   b) $\delta_1 \cap \delta_2 = \varnothing$,
   c) if there exists $\Phi = \{P_1\}$ such that $M_1(P_1 , \varphi(P_1)) = M_2 (P_1, \varphi(P_1))$;

2. There exists a set $E = [P_2: P / P_2 \notin \Phi]$ for which:
for any object instance O_i in P_2, O_i exists in marking M_1 and M_2. We denote this by:
O_i: Object: $O_i(P_2, M_1) = O_i(P_2, M_2) =$ ("CP",100) and,
$\text{StateObject}(P_2,M_1) = \text{InProd}$ and,
$\text{StateObject}(P_2, M_2) = \text{NormalProd}$.

This results in $\text{StateObject}(P_2,M_1) \neq \text{StateObject}(P_2, M_2)$.



C2
M0
M1
P1
P2
<(code='RCP', rate=96),TankFull>
<(code='CP', risk=90),ProdNormale>
<(code='FF',risk=94),ProdNormale>
C1
<(code='RC', rate=96),TankFull>
<(code='CP', risk=100),InProd>
<(code='FF',risk=94),NormalProd>
C2
C1
M2
<(code='RCP', taux=96),TankFull>
<(code='CP', risk=99),InProd>
<(code='FF',risk=94),NormalProd>

Figure 4. Marking graph – contradiction rule.

We can say that firing C_1 and C_2 produces places with the same object instances but in different states, which is obviously a contradiction.

e) Proposition of unnecessary premises

An HES has unnecessary IF conditions between two rules R_i and R_j iff:

1. There exist two final sequences σ_i and σ_j resulting from final markings M_i and M_j respectively, such that M_i is produced by firing R_i, denoted $Mi = \delta(M_0,\sigma_i)$ and M_j is produced by firing R_j and denoted $M_j = \delta(M_0, \sigma_j)$;
2. They do not have the same path: $\sigma_i \cap \sigma_j = \varnothing$;
3. For any place P_s from the set P of places, there exist two different colors (c, c') for this place in M_i and M_j respectively, denoted by:

$$\forall\ p: P, \exists\ c,c' \in \psi\ / (p,c) \in M_i \text{ and } (p,c') \in M_j \Rightarrow c \cap c' = \varnothing.$$

f) Proposition of unreachability

We know that for a rule to be reachable, it is necessary that one of its triggering states be obtained directly from an initial marking or indirectly from an intermediate marking in the reachability graph. If in the graph a rule does not use any initial or intermediate marking, then this rule is unreachable. In an HES, a rule R_i is unreachable iff:

1. $\forall\ \sigma_i$, a marking sequence such that M' is obtained from M_0, and thus we have:
$M' = \delta(M_0, \sigma_i) \Rightarrow \sigma_I \cap \{R_i\} = \varnothing$.

4.3 Dynamic Verification Principle

The users of the system are experts in their field. It is the case, however, that after having built the knowledge base, or any modification or update to it, inconsistencies or anomalies can easily occur, since no tool for checking anomalies is integrated into the automatic rules manager. The following steps show how we construct a graph-marking-based verification model:

1. Construct reachability graph nodes (markings) progressively, while there is an available rule to be fired and according to formal definitions 2,3,4,5,6,7 and 10;
2. Starting with the initial node (initial marking), build all the occurrence sequences as defined in section 3.1, definitions 1 and 4;
3. Analyze these occurrence sequences to detect anomalies according to formal definitions 12,13,14,15,16 and the formal propositions.

5. EXPERIMENTS AND DISCUSSION

Implementation of the simulation is composed of a set of Java classes and of an RB in which to present the anomalies (.ilr extension files). Once the Java classes are compiled (Figure 5), the inference engine is fired via JRules API imported into Java. The latter uses a rule file as the entrance, compiles the file to check for errors and initializes the work memory according to the initial facts contained in that rule file.

The reasoning session will start at that moment with a search for the rules that will verify the objects in the work memory. Only one rule is then chosen to be fired. That cycle will continue until there are no more tangible rules to apply or until the inference session is explicitly stopped by the user.

We carried out two types of simulations. The first uses the examples presented in section 4 and extracted from HKS ALCAN [2]. The second uses the well-known Game of Life for several types of patterns [22].

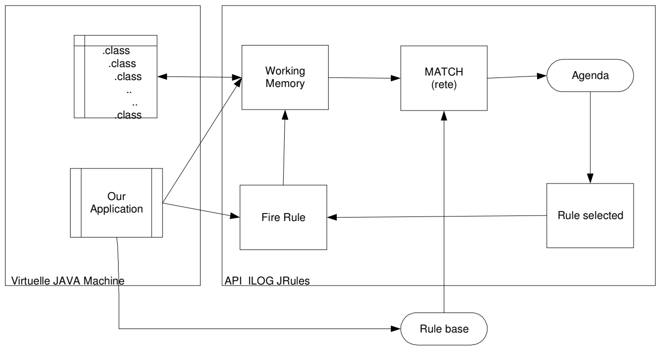


Figure 5. JRules - Java interaction.

5.1 Simulations with ALCAN

The objective of ALCAN [2] can be summarized in the following points:

- Efficient use of water;
- Consideration of future hydrological uncertainty;
- Satisfaction of the energy demand;
- Respect for security constraints.

For this purpose, integrating a tool for knowledge-base verification is necessary in order to ensure the maintenance of the knowledge base in a coherent state, thus guaranteeing reasoning devoid of inconsistencies and anomalies.

5.2 Simulations with the Game of Life

The Game Of Life is an animated representation of a population of cells developed first by the John Horton Conway [22]. In fact, it is a lattice of cells with a set of rules describing how successive generations of cells develop. The main interest of the game is that it enables complex phenomena to emerge from simple rules. The cell automaton in two dimensions is considered as a reference by researchers who are interested in the field of artificial life, because it shows that very simple rules may make it possible to bring to light non-trivial operations, and because it may simulate the abundance and diversity of life. Here is an example of an RB written in JRules which we simulated to create the situation of a redundancy.

```
import CasGameLife.*;
ruleset Life {
property context.class =
CasGameLife.ConstruireGraphe;};
rule fill_cell { when { ?c: Cell(isDead(); neighbors()==3); }
    then { setAlife(?c); } }; // R1
rule fill_Redondante { when { ?c: Cell(neighbors()==3); }
    then { setAlife(?c); }}; // R7
rule leave_empty { when { ?c: Cell(isDead();
neighbors()!=3); }
    then { setDead(?c); } }; //R2
rule loneliness { when { ?c: Cell(isAlife();
neighbors()<=1); }
    then { setDead(?c); } }; //R3
rule happiness {when {?c: Cell(isAlife(); ?n:neighbors();
                    (?n==2 || ?n  ==3));}
    then {setAlife(?c);}}; //R4
rule overcrowding { when { ?c: Cell(isAlife(); neighbors()
> 3);
    then { setDead(?c); } }; //R5
```

Globally speaking, simulations have revealed that:

1. The results manually predicted in section 4 were confirmed by the computerized application; we can confirm that, the modelisation of HKBS and checking anomalies proposed in this paper can be automated by a program software;
2. In order to verify the correctness of the heuristic CPN marking graph construction algorithm, we have conducted simulations for several patterns ("Glider»,» Small Exploder", "Exploder", "10 Cell Row", "Fish", "Pump", "Shooter", "Slow", "Fast", "Hyper") in the Game of Life application and all these simulations gave us satisfactory results. However, only pattern Exploder is presented here for brevity.
3. The *Game of Life* case enabled us to confirm that the construction of the marking graph and the analysis of occurrence sequences would definitely make it possible to predict an anomaly. Actually, the results of the inferences of the RB without anomalies (Figure 6.a) and those with redundancy (Figure 6.b) are the same. Only the analysis of occurrence sequences allows us to predict the differences (Figure 7).
4. To the author's best knowledge, this work is the first research for dynamic verification of rule-object KBSH; it can be used as a reference approach for all new research in this area.

6. CONCLUSION

Even though an non-monotonic Object-Rule system is not a simple task, few researchers have dealt with that set of problems (section 3). In this paper, we have proposed an approach for the dynamic verification of anomalies in an Object-Rule HKBS which may be implemented by a computer program. We successfully tested this technique on some examples. However, because of the complexity of the problem and the lack of similar work, certain aspects should be improved. Our main contributions are:

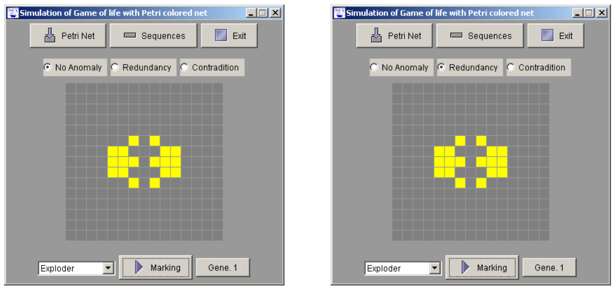
a) to propose a complete modeling of the Object-Rule Hybrid Knowledge-Based System by a CPN; in fact, we propose five redefinitions and four new formal definitions of the CPN for constructing the model of our system, a model in which all the parts of an HKB (classes, rules, conditions, actions and objects) are presented by the modeling elements of the Petri networks (places, transitions, expression of arcs and tokens);

b) to propose a formal solution based on the analysis of the sequences of the markings graph to analyze the anomalies in Object-Rule HKBS.

This paper illustrates the capacity of the proposed technique to identify situations of incompleteness anomalies in an Object-Rule HKBS in an environment of interactive execution. That verification is based on the analysis of the traces of the sequences and occurrence sequences and on marks.

Extensions of this research work are currently being studied:

- to complete the formal analysis proposals for the case of concentricity;
- to propose a new approach, based strictly on an analysis of the final places of the actual Petri network;
- to carry out simulations on large Knowledge Bases.



(a) (b)

Figure 6. Game of Life (a) Result without anomaly. (b) Result with redundancy.

7. ACKNOWLEDGMENT

We wish to acknowledge the funding provided by the Natural Sciences and Engineering Research Council of Canada (NSERC), the Décanat à la formation of École de technologie supérieure (ÉTS), and the Centre de recherche informatique de Montréal.

8. REFERENCES

[1] S.C.K. Shiu et al., "Formal Description and Verification of Hybrid Rule/Frame-Based Expert Systems", *Expert Systems with App.*, Vol. 13, N° 3, pp. 215-230, 1997.

[2] M. Essalihe, "Vérification Dnamique d'une Base de Connaissances Hybride Objet-Règle", *Mémoire de maîtrise, CRIM 2000*, pp. 1-160, 2000.

[3] F. Coenen, C. de Banc, "Maintanance of Kowledge Based Systems", *Academic Press*, 1993.

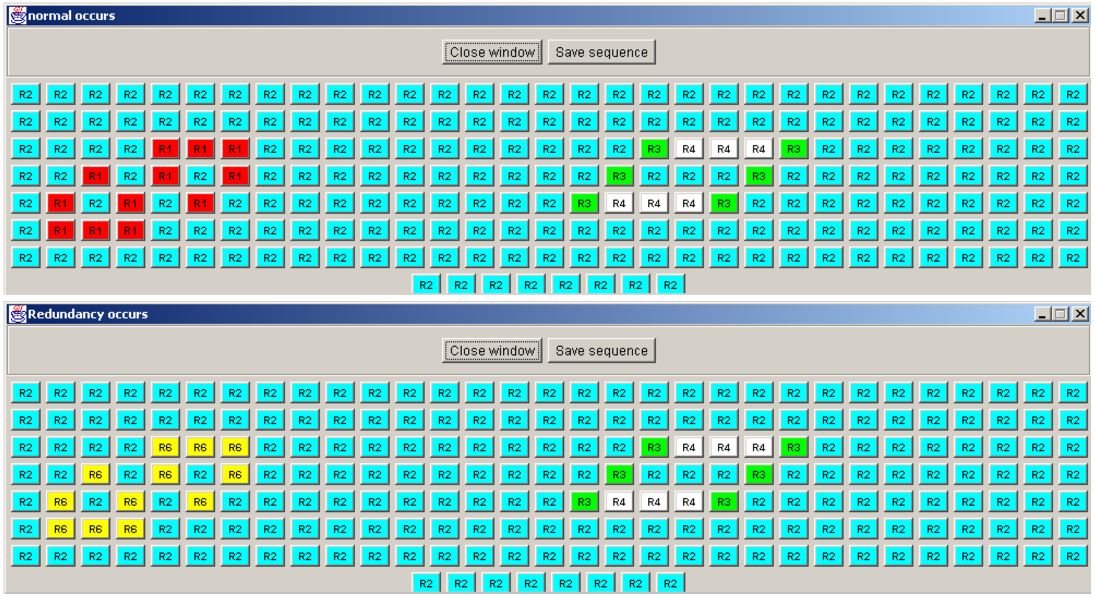

normal occurs
Close window
Save sequence
R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2
R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2
R2 R2 R2 R2 R1 R1 R1 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R3 R4 R4 R4 R3 R2 R2 R2 R2 R2 R2 R2 R2
R2 R2 R1 R2 R1 R2 R1 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R3 R2 R2 R2 R3 R2 R2 R2 R2 R2 R2 R2 R2 R2
R2 R1 R2 R1 R2 R1 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R3 R4 R4 R4 R3 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2
R2 R1 R1 R1 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2
R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2
R2 R2 R2 R2 R2 R2 R2 R2
Redundancy occurs
Close window
Save sequence
R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2
R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2
R2 R2 R2 R2 R6 R6 R6 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R3 R4 R4 R4 R3 R2 R2 R2 R2 R2 R2 R2 R2
R2 R2 R6 R2 R6 R2 R6 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R3 R2 R2 R2 R3 R2 R2 R2 R2 R2 R2 R2 R2 R2
R2 R6 R2 R6 R2 R6 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R3 R4 R4 R4 R3 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2
R2 R6 R6 R6 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2
R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2 R2
R2 R2 R2 R2 R2 R2 R2 R2

Figure 7. Game of Life - occurrence sequences.

[4] R. O'keef, D. O'leary, "Expert System Verification and Validation: A Survery and Tutorial", *Art. Int. Review*, pp. 3-42, 1993.

[5] U. G. Gupta, "Validation and Verifying Knowledge Based Systems", *IEEE computer, Society Press*, 1991.

[6] S.C.K. Shiu et al., "Formal verification of Some Potentiel Contradictoires In Hybrid Rule-Frame Based Expert Systems", Int. Con. on system MAN & Cybernics, USA, pp. 4424-4429, 1997.

[7] M. Tamiru, R. Agarwal, "A Petri Net Based Approach for Verification the Integrity of Production Systems", International journal of Man-machine studies, pp. 447-468, 1991.

[8] A. Kandel, Smith Suzanne, "Verification and Validation of Rule Based Expert Systems", CRC Press Inc 2000, pp. 53-103, 1993.

[9] A. D. Preece, R. Shinghal, "Cover: Practical Tool for Verifying Rule-Based Systems, Validation and Testing", *Workshop notes from 9th national conference on Art. Int., AAAI,* 1991.

[10] S.C.K. Shiu et al., "Modeling Hybrid Rule-Frame Based Expert Systems Using Coloured Petri Nets", *In proc. of 8th int. conf. on Industrial & Eng. App. of AI and ES, Australia*, pp. 525-532, 1995.

[11] S.C.K. Shiu et al., "An Approach Towards the Verification of Hybrid Rule/frame-based Expert Systems Using coloured Petri Nets", *Proc. of Int. Conf. on sys. and Cyb.,* pp. 2257-2262, 1996.

[12] S.C.K. Shiu et al., "Formal Verification of the Correctness in Hybrid Expert Systems", 1st Int. conf. on Knowledge Based Intelligent, *Electronic Sys.*, Vol. 02, pp. 419-428, 1997.

[13] K. Jensen, "Coloured Petri Nets: Basic Concepts, Analysis Methods and Pratical Use", *Springer-Verlag*, vol. 2, 1995.

[14] K. Jensen, "Coloured Petri Nets : Basic Concepts, Analysis Methods and Practical Use", *Springer-Verlag*, vol. 3, 1997.

[15] R.E. O'keef, D.E O'leary, "Expert System Verification and Validation: A Survery and Tutorial", *Art. Int. Review*, pp. 3-42, 1993.

[16] R.A Stachowitz, C.L. Chang, "Verification and Validation of Expert Systems", *Tutorial note at AAAI-88*, 1988.

[17] M. Suwa et al., "An Approach to Verifying Completeness and Consistency in Rule-Based Expert System", *AI Mag.*, pp. 16-21, 1982.

[18] F. Coenen, Chapon de Banc, "Maintenance of Knowledge Based Systems", *Academic Press*, 1993.

[19] C.L. Chang et Al., "A Report on the Expert Systems Validation Associete (EVA)", *Exp. Sys. with Application*, pp. 219-230, 1990.

[20] D. E. O'Leary, "The Impact of Semantic Ambiguity on Bayesian Weights", *European jour. of op. research*, pp. 163-169, 1995.

[21] C.K Shiu et al., "Formal Verification of Some Potentiel Contradictoires in Hybrid Rules-Frame Based Expert Systems Using Coloured Petri Nets", *int. conf. on sys. Man and Cybernics, Florida, USA*, pp. 4424-4429, 1997.

[22] Martin Gardner, "Mathematical Games, The fantastic combinations of John Conway's new solitaire gamelife",*http://ddi.cs.unipotsdam.de/HyFISCH/Produzieren/lis_projekt/proj_gamelife/ConwayScientificAmerican.htm*, 2003.