An Approach Towards the Verification of Hybrid Rule/Frame-based Expert Systems Using Coloured Petri Nets

Simon C. K. SHIU*, James N. K. LIU, Daniel S. YEUNG
Department of Computing, Hong Kong Polytechnic University
Hung Hom, Kowloon, Hong Kong
*Email : csckshiu@comp.polyu.edu.hk

ABSTRACT

High level Petri Nets have recently been used for many AI applications, particularly for modelling traditional rule-based expert systems. The major effect is to facilitate the analysis of the knowledge inference during the reasoning process, and to support the system verification which increasingly becomes an integral part of expert system development. Nevertheless, there is not much attention being put on systems other than the traditional ones. In this paper, we described an approach to model hybrid (rule- and frame-based) expert systems using Coloured Petri Nets and the concept of controlled state tokens. The analysis of the proposed model is by constructing and examining the reachability tree spanned by the knowledge inference. Such methodology has an implication for supporting the verification process in hybrid systems.

1. INTRODUCTION

Expert Systems (ES) have reached the stage where they are implemented and used in a wide variety of organizations and industries, a selection of operational expert systems in US, Europe, Canada and the Far East can be found in [9][10][15] and [18]. There is increasing need for expert systems validation and verification (V&V) because erroneous advice may lead to invaluable economic loss and even fatal loss of life in some domain applications. Traditionally, attention has been concentrated on using verification techniques to tackle rule-based systems [1][4][11][13][17]. However, these techniques exhibit a limited range of applicability. They could not cope with the kind of hybrid expert systems (HES), rule-based plus frame-based, which many of today's expert systems are developed [3][14]. The use of this hybrid approach integrates the power of organizing data objects in a class hierarchy and reasoning about the objects through user pre-defined logical associations. This advantage accounts for many popular expert system developing software, such as ADS, ART, EXSYS EL, KAPPA-PC, KBMS, Nexpert Object, Level5 Object, ProKappa, ReMind, which combine some sort of frame-based representation with a rule-based inference engine. Although this approach benefits from the advantages of both representation techniques, it complicates the V&V of the expert systems.

Traditionally, there are a few approaches in modelling expert systems, such as Normal Form approach, Decision Table Method, Incidence Matrix Method, Truth Maintenance Systems and Generic Rule Systems. One of the major criticisms of the above techniques is that none or very little consideration is given to allow for the dynamic checking of the knowledge inference. On the other hand, Coloured Petri Nets (CPN)[6], can support a formal description for modelling systems, which consists of concurrent and synchronous activities. Besides, they also have a graphical representation and a well-defined semantics, allowing for dynamic analysis of the modelled system. In this paper, a contribution is made to the modelling of hybrid rule/frame-based expert systems. We will introduce an approach based on CPN plus the concept of state tokens[12] for the representation of knowledge inference in HES, thus enhancing the quality and reliability of the modelled system. We will examine the transition sequences and check against the properties of the network in CPN for HES modelling. The paper has five main sections. Next section describes the knowledge representation and inference of a hybrid expert system, the third Section defines the errors and anomalies of HES and Section four gives the definitions of CPN and illustrates how HES can be modelled by CPN. Section five discusses the methods for analyzing the CPN and the implications of our approach to support formal verification of the systems. The last Section gives the conclusion and discussion.

2. A HYBRID EXPERT SYSTEM

A Hybrid Expert System combines multiple representation paradigms into a single integrated environment. For a Rule- and Frame-based integration, it composes of the following key features: Object Classes, Slot Attributes, Inheritance Relations, Demons, Methods, Rules and Reasoning Strategies. These features can be analysed using three conceptual views [5] of an expert system, they are: (1) An Object View which encapsulates a module of knowledge (or a concept). These knowledge modules (concepts) are represented by Object Classes. Inheritance Relations describe how these knowledge modules are related. (2) A Function View which specifies the functional behaviour of the objects within the expert system. These functions are represented using Methods and Demons. (3) A Control View which specifies the sequence of knowledge inference in the expert system. These controls are represented in terms of Rules and Reasoning Strategies.

In practical HES development [16], Frames are used to represent domain objects, various kinds of Demons are used to implement procedures attached to specific slots, Inheritance is used to inherit Class properties among Object Classes, Message Passing is used for interaction among different objects and Methods are used to perform algorithmic actions or some array manipulation within an object. Rules are used to describe heuristic problem-solving knowledge, Forward and Backward chains are commonly used to reason with rules. Therefore, in HES, the Frame base can be seen as being used to define the vocabulary for the Rule base, i.e. the

0-7803-2559-1/95 $4.00 © 1995 IEEE

possible values that slots can be defined and so specified, and the literal used to construct rules must conform to the restrictions imposed by what is available from the class hierarchy. The Frame base is married together with the Rules designed to manipulate it. The specific integration mechanisms of HES are as follows:

- Rules with Message Passing : Rules send or receive messages to and from objects for testing the Rules' premises.
- Rules with Inheritance : Rules directly read and write data into slots in a parent object and through inheritance of this slot's value to its children objects, trigger other rules to fire.
- Rules with Demons : Rules directly read and write data into slots and cause the execution of the associated Demons, which then trigger other rules to fire.
- Rules with Methods : Rules are embedded as part of an object's methods. Since methods are arbitrary pieces of code attached to an object, they can access the rules through function calls.
- Rules with Instances : Rules can be used to create/delete an instance of a specific Object Class.

Based on the above concepts of integration, a Hybrid Expert System, therefore, can be formally defined as a tuple HES = (C, A, I, In, D, M, R, S) satisfying the requirements below:

C = a finite set of object classes, where each object class is a Cartesian product of (A x D x M).
A = a finite set of attributes. Each attribute is of a simple type.
I = a specific object element from an object class C.
In = an inheritance relation. It is defined from the partially ordered relations in C.
D = a demon function. It is defined from A into an expression such that: $\forall a \in A \wedge \forall c \in C : a \wedge f(a) \in c$. (This means the demon functions can only change a slot's values within the same object instance).
M = a finite set of methods. It is defined as procedures in C.
R = a finite set of rules. Each rule is defined as a function from A such that $a \wedge f(a) \in A$. (This means the literal used to construct rules must come from the attribute set A).
S = a finite set of reasoning strategies.

Object class here is defined as having a set of attributes, demons and methods. Each attribute is defined as of a simple data type: e.g. string. Each specific object element is called an instance of the Object Class and will have different attribute values. Inheritance is defined as a partial order on the set Object Class, it is a relation that is reflexive, antisymmetric and transitive:

- Reflexive : For every Object Class, it inherits the properties from itself.
- Antisymmetric : For every Object Class, if A inherits from B and if B inherits from A, it implies that A is B.
- Transitive : For every Object Class, if A inherits from B and if B inherits from C, it implies that A inherits from C.

The above definition only covers simple inheritance, in the case of multiple inheritance, more elaborate definition is required.

A Demon is defined as a function which is executed when a slot value is either updated, or needed. Sometimes, a Demon can also act like a validation trigger which checks the cardinality and/or constraints imposed on a particular slot. The effects of a Demon are confined always locally to the same Object Class.

Methods are procedures attached to an Object Class, that will be executed whenever a signal is passed through. This way of executing a method is known as Message Passing.

Rules will interact with the information contained in the slots of the various Object Classes within the HES.

Finally, in HES, there should be a set of reasoning strategies. Some common ones are :

- Backward Chain with Inheritance : Goal directed search with inheritance as one of the means to establish the rule chains which across different Object Classes.
- Forward Chain with Inheritance : Data directed search with inheritance as one of the means to establish the rule chains which across different Object Classes.

Other important inference strategies includes : Pattern Matching, Unification, Resolution and Heuristic search.

Although the integration of a Rule- and Frame-based Expert System can take the advantages of both representation paradigm, this systems are not free from errors and anomalies. Therefore, the traditional V&V of Rule-based expert systems will need to be expanded in order to cover the additional anomalies caused by inheritance, methods, demons and other constructs within the object hierarchy. They will be discussed in the following Section.

3. SOME POTENTIAL ERRORS AND ANOMALIES

Subsumption anomalies of hybrid expert systems have recently been discussed in [8]. They defined subsumption as: if literals used in condition statements of a rule refer to objects which inherit from their supersets referred to in literals used in condition statements of another rule, they consider these two statements to be in inheritance relations. For example, (1) IF X has a sports car THEN insurance premium of X is high; (2) IF X has a Prosche THEN insurance premium of X is high. These two rules are in an inheritance relation, they can cause subsumption anomalies because whenever rule 2 succeeds, rule 1 will always succeed.

[8]'s definition of subsumption anomalies in hybrid expert system is very useful for conceptual understanding. However, their use of an Object as an unit of inference (Eg. IF X is registered; IF X is younger than 25; IF X has a Master Degree..etc.) is rather general for practical use. We would like to extend the anomalies concepts in a hybrid expert system based on attributes of objects. (Eg. IF X:Status is registered; IF X:Age is younger than 25; IF X:Qualification is a Master Degree..etc., where Status, Age and Qualification are slot attributes of the Object X.). Therefore, we would like to extend the definition of errors and anomalies which due to the integration of Rules with the Inheritance hierarchy of a hybrid expert system as follows:

Redundancy

In the case of rules which have identical conditions and actions, this implies the existence of redundant rules.

Rule 1 : A∧B⇒C
Rule 2 : A'∧B'⇒C'

(A, B & C are slots in the parent object, A', B' and C' are slots in the child object and A=A', B=B', C=C' because of inheritance).

Rule 3 : A⇒C
Rule 4 : A'⇒B'
Rule 5 : B'⇒C

In the case of a chained inference, some rules could become redundant if the same result could be inferred by alternative transitions even the same input facts are given. Rule 3 could become redundant as C could be inferred by an alternative transition, Rule 5, via Rule 4. (A inherit its slot value to A').

Dead End Rules (Missing Slots)
A value, slot or frame is missing if it appears as the premise or conclusion in the rules but is not defined in the Frame hierarchy. In this case, the antecedent part of the rule cannot be satisfied because it contains a literal which cannot be matched to a fact or a literal in the consequent part of any other rule.

Subsumption
If there are two or more rules which have identical antecedents and consequents except for the order of the literal then the situation is considered as *absolute subsumption.* When two rules have duplicated consequents and the antecedent of one is a subset of that of the other, or when two rules have duplicated antecedents and the consequent of one is a subset of the consequent of the other, or a mixture of these states, we consider this as *complex subsumption.* In a chained inference, a set of rules may be summarized by a single rule, we considered this as *compound subsumption* [2]. In HES, we need to incorporate the semantics of the object relations for identifying implicit subsumption among rules. Thus, if we have two rules :

Rule 6 : A∧B⇒C
Rule 7 : A'∧B⇒C

If the value of slot A inherits to slot A' (i.e. A is the parent and A' is the child), then Rule 7 subsumes Rule 6 because Rule 7 is just a more specialised case of Rule 6. (i.e. whenever Rule 7 succeeds, rule 6 will always succeed.) In a complex frame hierarchy which allows for multiple inheritance, checking for subsumption becomes more difficult because of ambiguity in the behaviour of multiple inherited subclasses.

Circular Rule Sets
If a circular loop can occur when a set of rules are fired, then these rules are considered as a circular rule set. For example :

Rule 8 : B⇒C
Rule 9 : C'⇒B

If slot C is the parent of C', Rule 8 and Rule 9 will form a circular loop. If more than one level of class hierarchy is involved, a implicit cycle may exist where the loop is formed from several rules and different frames' slots in the frame hierarchy.

Incompleteness
It exists if there are possible legal sets of facts in a frame's slot from which no useful conclusion can be inferred even having an exhaustive search of all relevant inference. Thus, if we have the following four rules :

Rule 10 : A(1)∧B(1)⇒C(1)
Rule 11 : C(1)∧D(1)⇒E(1)
Rule 12 : C(1)∧D(2)⇒E(2)
Rule 13 : A(1)∧B(2)∧D(1)⇒E(3)

The Arabic numerals represent the Cardinality, the number of possible values that may be placed in a particular slot, in the Frame slots A, B, C, D and E. We can find out that nothing can be inferred from {A(1) and B(2) and D(2), A(2) and B(2) and D(1), A(2) and B(2) and D(2)}.

Similarly, if we have two rules :

Rule 14 : A∧B(<10)⇒C
Rule 15 : A∧B(>20)⇒D

The value in slot B is within a numeric range, e.g. from 0 to 100, Rule 14 and Rule 15 is said to be incomplete because no inference exists if B is greater or equal to 10 and less than or equal to 20. In HES, Rule 14 and Rule 15 may be implemented as Demons attached to the slot B.

Inconsistency
If two rules have duplicate antecedents but in the consequents a clause is both affirmed and denied, we refer this as inconsistency. (ie Rule 16 and Rule 17 contradict to one another directly).

Rule 16 : A∧B⇒C
Rule 17 : A'∧B'⇒¬C

4. MODELLING THE HES USING CPN

Definition of Coloured Petri Net
A Coloured Petri Net can be defined as a tuple CPN = (Σ, P, T, A, N, C, G, E, I) satisfying the requirements below:

Σ = a finite set of non-empty types, called colour sets.
P = a finite set of places.
T = a finite set of transitions.
A = a finite set of arcs such that : P∩T=P∩A=T∩A=∅.
N = a node function. It is defined from A into PxT∪TxP.
C = a colour function. It is defined from P into Σ.
G = a guard function. It is defined from T into expressions such that: ∀t∈T :[Type(G(t))=B∧Type(Var(G(t)))⊆Σ]
E = an arc expression function. It is defined from A into expressions such that : ∀a∈A : [Type(E(a))=C(p(a))ms ∧Type(Var(E(a)))⊆Σ].
I = an initialization function. It is defined from P into closed expressions such that ∀p∈P : [Type(I(p))=C(p)ms].

The set of colour sets determines the types, operations and functions that can be used in the net inscriptions. The places, transitions and arcs are described by three sets P, T and A which are required to be finite and pairwise disjointed. The node function N maps each arc into a pair where the first element is the source node and the second the destination node. The two nodes have to be of different kind (i.e. one of the nodes must be a

place while the other is a transition). Several arcs may be allowed to link between the same ordered pair of nodes. The colour function C maps each place, p, to a colour set C(p). This means that each token on p must have a token colour that belongs to the type C(p). The guard function G maps each transition, t, to an expression of type Boolean, i.e., a predicate. All variables in G(t) must have types that belong to Σ . A guard is allowed to be a list of Boolean expressions [Bexpr1, Bexpr2..etc]=B. This means that the binding must fulfill each of the Boolean expression in the list. The arc expression function E maps each arc, a, into an expression which must be of type C(p(a))ms. This means that each evaluation of the arc expression must yield a multi-set over the colour set that is attached to the corresponding place. The initialization function I maps each place, p, into a closed expression which must be of type C(p)ms, ie a multi-set (a set which may contain multiple occurrences of the same element) over C(p).

HES Model

Frame data structure: Each framed data structure is represented by a compound colour set, and each frame instance is represented by a token in that set. For instance, if there are fifteen sets of non-empty types or colour sets being used to represent one frame data structure, i.e. Σ = AA,BB,...OO; Color AA may be defined as text strings; Color BB may be as Boolean; ...and Color OO may be defined from some already declared coloured sets, i.e. Color OO = Product AA * BB * CC. Each frame instance is declared as a variable of a particular colour set, i.e. var Instance-1 : OO (var denotes variable declaration which introduces one or more variables). Here we have one variable, Instance-1, which is with colour OO. We may use var Instance-1, Instance-2, Instance-3 : OO for declaring three different instances of the same object class with colour OO. In the following sections, we will use two variables, Object A, which is a particular instance of an Object Class A and State which is the state token with possible values of either Yes or No. (i.e. var Object A : OO and var State with Yes | No.)

Rules with Message Passing: Places in the CPN are taken to correspond to predicates of the production rules and the transitions in corresponding to the implications of the rules. Since the frame data structure is represented by the token of a particular color set. We can define arc expression such that they directly read and write data in the token's data slots. This will serve the purpose of sending or receiving messages(data value) to and from objects for testing the rules' premises.

Rules with Inheritance: Places in the CPN are taken to correspond to two different elements in the HES. First, places are taken to correspond to predicates of the production rules which are pre-defined earlier by the user. Secondly, places are taken to correspond to the Objects class in the HES's Frame hierarchy. Similarly, transitions in the CPN correspond to two different events in the HES. First, the transitions correspond to the implications of the rules. Secondly, the transitions correspond to the inheritance of the properties from Classes. The transition operations are represented by the arc expression functions. (e.g. A Rule R can be represented in CPN as shown in Figures 1a and 1b:)

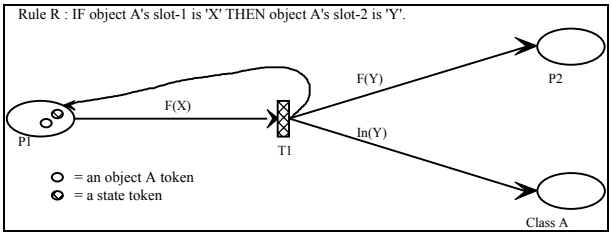

Figure 1a : Rule R with Inheritance (before firing) with an input token Object A and a state token in P1 with 'Yes'.

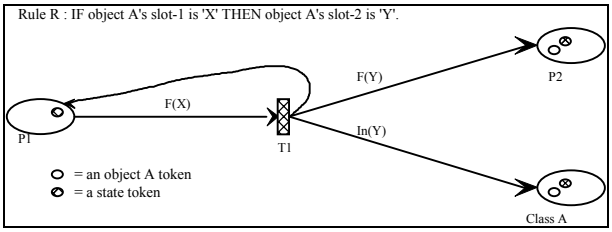

Figure 1b : Rule R with Inheritance (after firing) with an output token Object A both in P2 and Class A. A state token is also created in P2 and Class A with 'Yes' respectively.

P1, P2 and Class A are Places with the colour set that was used to represent the data structure of Object A. T1 is a Transition which is enabled iff the input arc expression F(X) is evaluated to be true (i.e., the premise IF Object A's slot-1 is 'X' is true). If F(X) is true then T1 is fired, it implies that Rule R is executed. All tokens will be removed from P1 and a new token Object A will be created in both P2 and Class A with new data values determined by the output arc expression F(Y) and In(Y) (i.e. F(Y) will assign 'Y' to Object A's slot-2; In(Y) will assign 'Y' to Object A's slot-2 and inherit this 'Y' to Object A's Children, the subclasses, through). A state token with 'Yes' value will also be created in P2 and Class A. It is used for further inference (if any) in P2, and is used for further inheritance (if any) in Class A. In order to preserve the state of the predicate in P1, a state token marked with 'Yes' is created in P1 via the self-loop.

Rules with Demons: Similarly, a Rule with Demon can also be represented by a Places/Transition tuple, e.g. if a demon is attached with object A's slot-overtime, whenever the value of slot-overtime is updated to 'Y' then the demon will execute and compute the slot-salary value by the formula 1.2*basic salary. This can be represented by Figure 2.

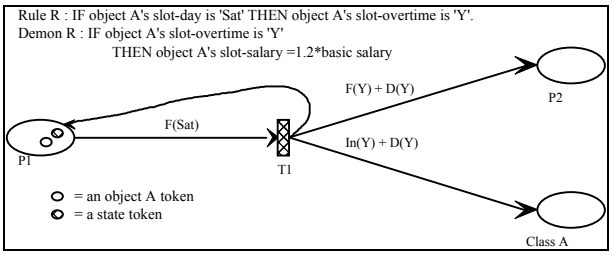

Figure 2 : Rule R with Demon (before firing) with an input token Object A and a state token in P1 with 'Yes'.

The demon function, D(Y), is represented as an arc expression. The firing of Rule R will trigger the demon function to execute.

Rules with Methods: Methods are procedures attached to an Object class, they can be represented by the Functions and

Operations declarations in CPN. The function takes a number of arguments and returns a result. The arguments and the result have a type which is a declared colour set, the set of all multi-sets over a declared colour set. A declared function can be used in arc expressions, guards and initialization expressions in the CPN. An example function which tells whether the argument is even or not might be as follow:

fun Even(n:integers)=((n mod 2)=0)

Operations can also be used to represent Methods. In both Functions and Operations declarations, different kinds of control structures can be built. e.g. CASE; IF b is true THEN statement 1 ELSE statement 2; WHILE b is true DO; REPEAT statement 3 UNTIL b is true.

The Rules with Methods can thus be represented by CPN as follow (Figure 3a, 3b, 3c and 3d, the self loops are omitted for clarity reason)

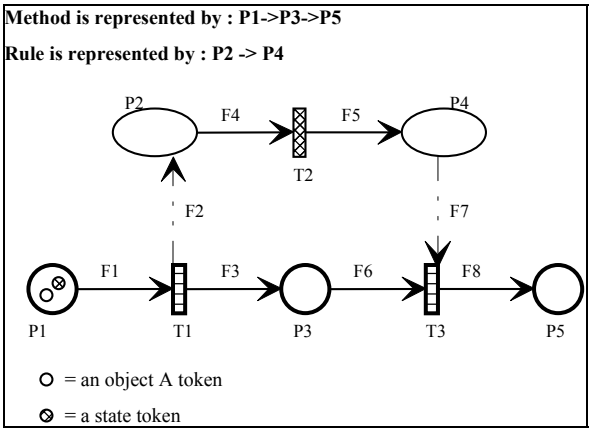

Figure 3a : Rule with Method (before firing) with an input token Object A and a state token in P1 with 'Yes'.

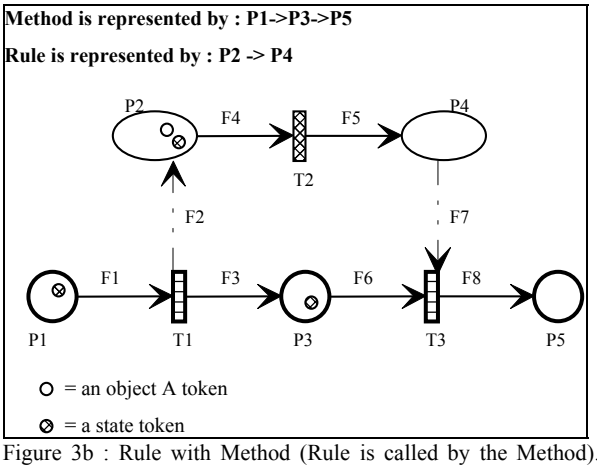

Figure 3b : Rule with Method (Rule is called by the Method). The token Object A was passed to P2 and a state token in P1, P2 and P3 'Yes' respectively.

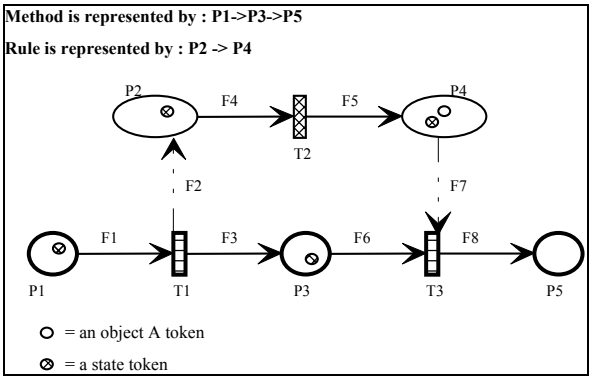

Figure 3c : Rule with Method (After firing). The token Object A is in P4 and a state token in P1, P2 and P3 and P4 with 'Yes' respectively.

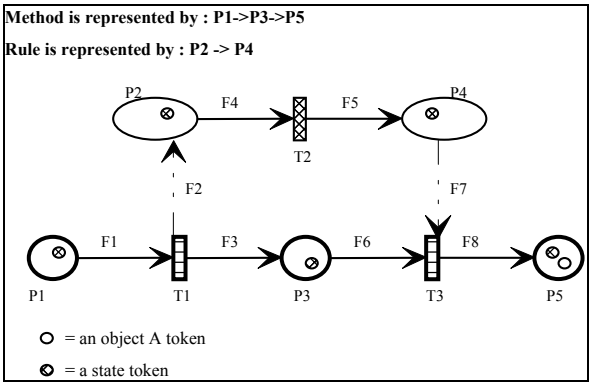

Figure 3d : Rule with Method (Method resumes control). The token Object A was passed to P5. A state token in P1, P2, P3, P4 and P5 with 'Yes' respectively.

P1 to P3 to P5 represent the main body of the Method. P2 to P4 represent the Rule embedded within the Method. F1 is the first part of the Method which executes first, then control is passed to the Rule by F2 which will create the Object A in P2. After firing of the Rule (T2 is enabled and fired), P3 and P4 will allow T3 to be fired. F8 represented the remaining part of the Method which will act on Object A correspondingly. After execution of this Rule with Method, a state token is deposited in all the Places, P1 to P5 for preservation of the states.

Rules with Instances: This is represented in CPN by the arc expressions because the number of removed/ added tokens and the colours of these tokens are determined by the value of the corresponding arc expressions.

5. ANALYSIS OF COLOURED PETRI NETS

The major analysis technique, within the context of expert system verification, is the use of reachability tree which represents the reachability set of the CPN (or occurrence graph in Jensen's terminology). The basic idea behind is to construct a tree/graph containing a node for each reachable marking and an arc for each occurring binding element. In expert system verification, it refers to exhaustively exploring all the useful and relevant interactions of predicates within the model. From a given initial state, all possible transitions are generated, leading to a number of new states. This process is repeated for each of the newly generated

states until no new states are generated. Obviously such a tree/graph may become very large even for a small CPN. However, research [7] has been taken to allow for a partial examination of a subportion of the reachability graph, therefore reduce the efforts in deriving possible solutions. For simplicity reason, without taking any transition conditions or transition operations into consideration, we concentrate our analysis by enabling a specific transition and then check the reachability set for any irregularities of the associated predicate places. The checking of the irregularities explained in Section 3 can be done exhaustively or heuristically by adequately initiation of the sequence of transitions and closely examining the reachability markings. The problems can be located through the trace of the sequence of transitions which may provide alternative or multiple marking effects.

6. CONCLUSION AND DISCUSSION

In this paper, we have described an approach to model hybrid rule/frame based expert system using Coloured Petri Nets and the concept in State Controlled Petri Nets. The frame data structures are represented using colour tokens. The data value of each colour token may be of an arbitrarily complex type, thus enabling various classes of frames be represented. The rules and demons are represented by a Place/Transition tuple with a self-loop in order to preserve the state of the predicate. The analysis of the CPN is by constructing and examining the reachability tree to detect irregularities of the predicate places.

In a pure frame-based expert system, reasoning is by comparing descriptions of incoming facts with the frames in the knowledge base, and retrieving the class frame that best matches the situation. The main inference mechanism or strategy for applying general information to specific instances is inheritance. This reasoning mechanism is rather limited in practical situations. In a pure rule-based expert system, reasoning is by firing a sequence of rules using incoming facts. Although this method is simple and useful, complex domain knowledge could not be represented. Our approach is useful to model systems that combined rule/frame based representation techniques.

Future work will include formalizing our approach and developing of algorithms to detect irregularities in the HES. We would also like to investigate further the capability of the methodology to handle fuzzy systems and the complexity involved against the traditional approaches.

7. REFERENCES

[1] Beauvieux, A. A General Consistency Checking and Restoring Engine for Knowledge Bases. In *Proceedings of the 9th European Conference on Artificial Intelligence,* Stockholm, Sweden, 1990.

[2] Coenen, F. and Bench-Capon T., *Maintenance of Knowledge-Based Systems*, Academic Press, 1993.

[3] Durkin, J. *Expert Systems: Design and Development,* Macmillan Publishing Company, pp.12-23, pp.711-771, 1994.

[4] Evertsz, R. The Automated Analysis of Rule-based Systems Based on their Procedural Semantics. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*. Sydney, Australia, 1991.

[5] French, S.W. and Hamilton, D. A Comprehensive Framework for Knowledge-Base Verification and Validation. In *International Journal of Intelligent Systems,* Vol. 9, John Wiley & Sons, Inc., pp.809-837, 1994.

[6] Jensen, K. *Coloured Petri Nets : Basic Concepts, Analysis Methods and Practical Use, Vol. 1*, Springer-Verlag, 1992. Vol. 2, Springer-Verlag, 1995.

[7] Li, X., Lai, R. and Dillon, T.S. A New Decomposition Method to Relieve the State Space Explosion Problem. In *Proceedings of the 5th International Conference on Computing and Information*, Sudbury, Ontario, Canada, pp.150-154, 1993.

[8] Lee, S. and O'Keefe, R.M., Subsumption Anomalies in Hybrid Knowledge Bases. International Journal of Expert Systems, Vol. 6, No. 3, pp.299-320, 1993.

[9] Lee, J.K., Yeung, D.S., Mizoguchi R. and Narasimhalu D. *Operational Expert System Applications in the Far East*, Published by Pergamon Press, 1991.

[10] Liebowitz, J. *Operational Expert System Applications in the United States*, Published by Pergamon Press, 1991.

[11] Liu, N.K. and Dillon, T.S. An Approach Towards the Verification of Expert Systems Using Numerical Petri Nets. *International Journal of Intelligent Systems,* 6, pp. 255-276, 1991.

[12] Liu, N.K. *Formal Description and Verification of Expert Systems*. Ph.D. Dissertation, La Trobe University, Bundoora, Victoria, Australia, 1991.

[13] Nguyen, T.A., Perkins, W.A., Laffey, T.J. and Pecora, D. Checking an expert system knowledge base for consistency and completeness. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence,* Los Angeles, CA. 1985.

[14] O'Keefe, R.E. and O'Leary, D.E. Expert System Verification and Validation: A Survey and Tutorial, *Artificial Intelligence Review* 7, pp.3-42, 1993.

[15] Suen, C.Y. and Chinghal, R. *Operational Expert System Applications in Canada*, Published by Pergamon Press, 1991.

[16] Shiu, S.C.K., Liu, J.N.K. and Yeung, D.S. Modelling Hybrid Rule/Frame-based Expert Systems using Coloured Petri Nets, In *Proceedings of the 8th International Conference on Industrial & Engineering Applications of Artificial Intelligence and Expert Systems*, IEA/AIE-95, Melbourne, June 6-8, pp.525-531, 1995.

[17] Suwa, M., Scott, A.C. and Shortliffe, E.H. An Approach to Verifying Completeness and Consistency in a Rule-based Expert System. *AI Magazine*, pp.16-21, 1982.

[18] Zarri, G.P. *Operational Expert System Applications in Europe*, Published by Pergamon Press, 1991.