Pergamon

Expert Systems With Applications, Vol. 13, No. 3, pp. 215–230, 1997

Printed in Great Britain
0957-4174/97 $17.00+0.00

PII: S0957-4174(97)00027-4

Formal Description and Verification of Hybrid Rule/Frame-Based Expert Systems

SIMON C. K. SHIU*, JAMES N. K. LIU AND DANIEL S. YEUNG

Department of Computing, Hong Kong Polytechnic University, Kowloon, Hong Kong

Abstract—*A Hybrid Expert System combines multiple representation paradigms into a single integrated environment for modeling and reasoning of complicated real world phenomena. For a Rule- and Frame-based integration, it models the problem domain using the concepts of Classes and Rules together. If Domain knowledge (concepts) is related by production rules and frame hierarchy, then anomalies may arise among this knowledge (concepts) due to the existence of two mutually independent formalism of relations. A contribution is made in this paper to developing formal description techniques for the detection of anomalies attributed to the integration of production rules with the inheritance of object properties within the object hierarchy. The main idea is to convert the HES into a State Controlled Coloured Petri Net (SCCPN) where the object hierarchy, property inheritance and production rules are modelled as separated components in the same SCCPN. The detection and analysis of the anomalies (i.e. Correctness, Consistency and Completeness) in the system are done by constructing and examining the reachability tree spanned by the knowledge inference. © 1997 Elsevier Science Ltd. All rights reserved*

1. INTRODUCTION

THERE ARE A WHOLE RANGE of problems and difficulties hindering the development of Expert Systems. Typically, the bottleneck is knowledge acquisition, representation of surface and deep knowledge, creativity modelling, temporal reasoning, causal and common sense reasoning, uncertainty reasoning, combinatorial explosions, conflict resolutions, and the like. The use of large amounts of domain knowledge to solve real world problems raises some concerns for the creation and maintenance of such systems (Geissman & Schultz, 1988; Duchessi & O'Keefe, 1995). Furthermore, since these systems tend to grow in an evolutionary manner, constant maintenance of knowledge is necessary to ensure correct system performance. The importance of validating and verifying Expert Systems are well documented (Gupta, 1991, 1993; O'Keefe & O'Leary, 1993; Coenen & Bench-Capon, 1993; Liu & Dillon, 1995; Murrell & Plant, 1996). One of the major criticisms of the above techniques is that none or very little consideration is given to allow for the dynamic checking (i.e. the verification is carried out in the process of the system reasoning (Matsumoto et al., 1991; Preece et al., 1996)) of Hybrid Expert Systems.

In a traditional 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 traditional 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. The use of a Hybrid Rule/Frame-based approach integrates the power of organizing data objects in a class hierarchy and reasoning about the objects through user pre-defined logical associations.

A Hybrid Expert System combines multiple representation paradigms into a single integrated environment for modelling and reasoning of complicated real world phenomena. For a Rule- and Frame-based integration, it models the problem domain using the concepts of Classes and Rules together. The essential key modelling features are: Object Classes, Slot Attributes, Inheritance Relations, Demons, Methods, Rules and Reasoning Strategies.

In order to allow for the automation of the verification of the HES process, to tackle the mathematical problems associated with the method, and to provide accurate detection of anomalies in the HES, a more formal approach (i.e. methods which are based on mathematical techniques) of the HES model is necessary. Thus, there are two major problems for HES verifications:

* To whom all correspondence should be addresssed.

- Expert Systems are developed using hybrid techniques, yet very little fundamental research work has been done for their verifications.
- A need to formalize the verification process to allow for automatic detection of anomalies in the HES.

In view of the lacking of proper understanding within this subject, i.e. Hybrid Expert System verification, this paper seeks to address the issues of knowledge description, formulation and verification in HES. It examines the problem of demonstrating a hybrid knowledge base to be correct, consistent and complete in terms of more global issues and provides a framework for verifying hybrid knowledge based systems. This article is organized into six main sections. The first section gives the introduction and motivation of our work. The second section gives some literature reviews of expert systems verification. The third section describes the fundamental principle, definitions, and properties of HES and SCCPN. Problem description and SCCPN representations of anomalies are provided in the fourth section. Section 5 defines a set of propositions for representing the formal properties in the SCCPN in which each of them corresponds to some anomalies in the HES. An assessment of the complexity of the SCCPN methodology is described in Section 6. Finally, the article concludes with a discussion on the future extension of our proposed methodology.

2. EXPERT SYSTEMS VERIFICATION

There has been an explosion of activity in the areas of Validation and Verification (V&V) of Expert Systems over the past 10 years. For example, one of the longest sequences of ongoing workshops at the AAAI (American Association for Artificial Intelligence) meeting has been the Workshop on Verification, Validation and Testing of Intelligent Systems. The first five workshops occurred from 1988–1992. The IJCAI (International Joint Conference on Artificial Intelligence) has had workshops on V&V since 1989. Furthermore, the European Conference on AI (ECAI) has had a number of workshops on V&V. Special Issue on Verification and Validation of Expert Systems had appeared in a number of Journals: International Journal of Human-Computer Studies, Vol. 44, 1996; International Journal of Intelligent Systems, Vol. 9, No. 8, 1994; Internal Journal of Expert Systems, Vol. 6, No. 3, 1993 and Expert Systems with Applications, Vol. 1, No. 3, 1990 and Vol. 8, No. 3, 1995. Large projects of Validation and Verification are funded by agencies including NASA, DARPA, and the European Community's ESPRIT program, such as RCP (Suwa et al., 1982), CHECK (Nguyen et al., 1985), ESC (Cragen & Steudel, 1987), COVADIS (Rousset, 1988), EVA (Chang et al., 1990), KB-REDUCER (Ginsberg, 1988), COVER (Preece, 1989), SACCO (Laurent & Ayel, 1989), NASA MMU-FDIR (Culbert, 1994), VALID (Meseguer, 1994), JIPDEC (Terano, 1994), SYCOJET and SACCO (Ayel & Vignollet, 1994).

This interest is driven by the need to test the large number of Expert Systems that have been developed since the mid-1980s. It also has been derived from the increasing role that intelligent systems are taking in critical situations, such as medicine and defense. The role and importance of verifying Expert Systems is well documented (Gupta, 1991, 1993; O'Keefe & O'Leary, 1993; Coenen & Bench-Capon, 1993; Liu & Dillon, 1995; Murrell & Plant, 1996). While there is controversy over how to define the terms verification and validation, there is a general consensus that validation refers to the process of building the right system (that is, substantiating that a system performs with an acceptable level of accuracy), while verification refers to the process of building the system right (that is, substantiating that a system correctly implements its specifications) (Nguyen et al., 1987; Preece, 1991; O'Keefe & O'Leary, 1993).

Typically, Expert Systems verification approaches are based upon the concept of an anomaly, where an anomaly is an abuse or unusual use of the knowledge representation scheme. An anomaly can be considered a potential error—it may be an actual error that needs correcting, or may alternatively be intended. Considerable research has been done on identifying rule-base anomalies (Gupta, 1991; Gamble et al., 1994; Liu & Dillon, 1995; Murrell & Plant, 1996), with the result that rule anomalies are now quite well understood. These may include

(a) Correctness
  - Redundancy: Identical or chained rules succeed in the same situation and have some common results.
  - Subsumption: Two rules have the same results but one contains additional constraints on the situations in which it will succeed.
  - Ambiguity: Indeterminate rules.
  - Cyclicity: Circular rules (i.e. Without a satisfactory terminating condition.

(b) Consistency
  - Contradiction: Conflicting rules (i.e. Two sequences of rules offering conflicting results).
  - Deadend: Rules which are executed and no other rules can succeed them.

(c) Completeness
  - Unreachability: Rules whose conditions can never be satisfied.
  - Omission: Missing rules.

Although there are comparatively less research work done on verifying Frame-based Expert Systems, both (O'Keefe & O'Leary, 1993) and (Coenen & Bench-Capon, 1993) pointed out that increasingly, implemented Expert Systems employed some variation of object-oriented methods to store attributes and procedural

attachments and provide inheritance. They defined the typical anomalies for a Frame-based Expert System are:

- Redundancy, (e.g. A slot or frame is redundant if it is not used to establish anything that the system is designed to address).
- Missing slots and Frames.
- Misplaced Slots and Frames, (e.g. given the property of inheritance, the location of a slot in a frame hierarchy can be highly significant).
- Duplication, (e.g. Duplicated slots).
- Inconsistency, (e.g. There exists a possible set of facts that would allow an entity to be instantiated to two different frames).
- Incompleteness, (e.g. There exists a possible set of facts that an entity could not be instantiated to a frame).

The earliest references to activities designed to ensure acceptable knowledge base system quality can be traced to efforts on the MYCIN project (Shortiliffe, 1976). Some of these efforts were aimed to fix spelling errors, checks that rules are semantically and syntactically correct through pairwise rule comparison, and to some extent points out potential erroneous interactions among any two rules. With greater acceptance of knowledge base systems as viable solutions for a specific range of problems, the need for more formal mechanisms to assure knowledge based system quality assumed greater importance. Independent research streams addressing the problems of completeness and consistency of the domain knowledge were now identifiable. Strategies include the use of Normal Form Approach (Charles, 1991); Decision Table Methods (Suwa et al., 1982; Nguyen et al., 1985); Incidence Matrix Method (Landauer, 1990); Knowledge Base Reduction (Ginsberg, 1988); Generic Rule Systems (Chang et al., 1990; Stachowitz & Chang, 1988; Stachowitz & Combs, 1987; Preece & Shinghal, 1991a,b); Bayesian Approach (O'Keefe & O'Leary, 1993; O'Leary, 1995); Statistical Investigations (Landauer, 1990; O'Leary, 1988); Rule Clustering (Jacob & Forscher, 1991; Mehrotra, 1991); Using Test Cases, (Cuda & Dolan, 1991) and Petri-Net Systems (Liu, 1996, 1993, 1991; Wu & Lee, 1995; Scarpelli & Gomide, 1994a,b; Yao, 1994; Zhang & Nguyen, 1994; Nazareth, 1993; Agarwal & Tanniru, 1992; Meseguer, 1990).

In modelling studies, nobody solves the problem—rather, everybody solves the model of the problem. Since an Expert System represents human reasoning and knowledge, we must justify its representation level through some kinds of checking and testing, basically, the verification. While 'verification' and 'validation' might have separate definitions, we can derive the maximum benefit by using them synergistically treating both as an integrated definition.

In this paper, the process of verification involves the checking of correctness, consistency and completeness in Hybrid Expert Systems. The approach adopted by this research illustrates the use of dynamic analysis that involves the execution of the system using a variety of inputs and scrutiny of the output for correct behavior. In general, correctness refers to the accuracy of the knowledge in the knowledge base. Consistency refers to the relationship between the information in the knowledge base and the ability of the inference engine to process the knowledge base in a consistent manner. It includes the checking for and reporting of built-in discrepancies, ambiguities, and redundancies in the contents of the knowledge base. Completeness refers to the amount of knowledge built into the knowledge base. It means that a knowledge base is prepared to answer all possible situations that could arise within its domain. It is hence one measure of robustness. Completeness checking is a debugging aid which finds logical cases that have not been considered, in other words, missing knowledge. As the input parameters increase, the potential number of cases increases exponentially, resulting in great human difficulty determining which situations have not been considered.

As such the verification of an Expert System attempts to show that the software programs of the system are correct in relation to the criteria. Verification tries to prove this correctness by formal means, whose correct application may again be examined by formal means. This provides greater reliability of the statement as to the correctness of a system than can be achieved by other, non-formally controllable validation means. In an effort to preclude confusion of other definition, verification in an Expert System will be constructed to be the demonstration of logical correctness, consistency, and completeness of the knowledge base. The view that verification is a process of ensuring these logical qualities does not necessarily imply enforcement of semantically correct performance. It should also be stressed that these qualities are not restricted to the theorem proving usage of the construct.

3. FUNDAMENTAL PRINCIPLE

A Hybrid Expert System combines multiple representation paradigms into a single integrated environment for modelling and reasoning of complicated real world phenomena. For a Rule- and Frame-based integration, it models the problem domain using the concepts of classes and rules together. The essential key modelling features are: Object Classes, Slot Attributes, Inheritance Relations, Demons, Methods, Rules and Reasoning Strategies. These features can be analyzed using three conceptual views (French & Hamilton, 1994) 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

TABLE 1
Conceptual Interpretation of HES in SCCPNs.

Hybrid Expert System	State Controlled Coloured Petri Net
Frame-based part	
Object Classes	Places
Object Class Types	Colour Sets
Object Instances	Tokens
Slots	Variables in Tokens
Facts in Slots	Binding of Variables with Constants
Inheritances	Transitions
Demon	Arc Expressions
Methods	Arc Expressions
Rule-based part	
Predicates	Places
Predicates States	Tokens
Rules	Transitions
Facts	Binding of Variables with Constants
Transition Operations	Arc Expressions

which specifies the functional behaviour of the objects. These functions are represented using Methods and Demons. (3) A Control View which specifies the knowledge inference in the expert system. These controls are represented in terms of Rules and Reasoning Strategies.

As HES is modelled by SCCPN, a mapping between the two structures is necessary, and is given in Table 1.

As shown in Table 1 the components of the HES are separately represented, which can be modelled explicitly by the SCCPN. The places are taken to correspond to predicates and object classes, and transitions to represent rules implications as well as inheritance. There are two major types of tokens, one is the state token which records the state of the predicate and the class type information. (i.e. Since rules may be fired by either parent class instance or child class instances). The second type of token is the object instance token which represents a particular object instance of a particular class within the object hierarchy. Transitions are fired to represent rules being executed or inheritance is being carried out. The maximum number a rule can be executed is equal to the total number of different class types. (i.e. each class type object instance can fire a particular rule once at most). Each input place of a rule has a self-loop arc for maintaining the state of the predicate. Similarly, the input place of an inheritance also has a self-loop arc for recording the inheritance execution. Methods and Demons are represented by functions in the arc inscription of the SCCPN. The net result is the exchange of colour tokens from places to places and a new marking, which is defined as the distribution of tokens over the places of the SCCPN, is obtained.

The SCCPN notation employed in this paper is an extension of State Controlled Petri Nets proposed by (Liu & Dillon, 1995), and Coloured Petri Nets proposed by (Jensen, 1995, 1996, 1997) and is specified as follows.

DEFINITION 3.1. A SCCPN can be defined as a 10-tuple given by= $(\Sigma, P, T, D, F, A, N, C, E, I)$, where satisfying the requirements below:

$\Sigma=\{\omega_1,\omega_2,...,\omega_i\}$, a finite set of non-empty types, called *colour sets*, $i\geq 1$,

$P=\{P_c, P_r\}$ a finite set of *places*,

$P_c=\{p_{c1}, p_{c2},..., p_{cj}\}$, a finite set of places that model the classes of the HES, called *class places*, $j\geq 1$,

$P_r=\{p_{r1}, p_{r2},..., p_{rk}\}$, a finite set of places that model the predicates of the production rules, called *predicate places*, $k\geq 1$,

$P_c\cap P_r$: the intersection of $P_c\cap P_r$ represents those IS-A *predicates* of the rule sets attached to the specific *classes*,

$T=\{T_c, T_r\}$, a finite set of *transitions*,

$T_c=\{t_{c1}, t_{c2}, ..., t_{cl}\}$, a finite set of transitions that are connected to and from *class places*, called *inheritance transition*, $l\geq 1$,

$T_r=\{t_{r1}, t_{r2},..., t_{rm}\}$, a finite set of transitions that are connected to and from *predicate places*, called *predicate transition*, $m\geq 1$,

$T_c\cap T_r=\varnothing$,

$D=\{d_1, d_2,..., d_n\}$, a finite set of *predicates*, $|P_r|=|D|$, $n\geq 1$,

$F=\{f_1, f_2,..., f_n\}$, a finite set of *classes*, $|P_c|=|F|$, $n\geq 1$,

$A=\{a_1, a_2,..., a_k\}$, a finite set of *arcs*, $k\geq 1$, $P\cap T=P\cap A=T\cap A=\varnothing$,

$N:A\rightarrow P\times T\cup T\times P$, a *node function*, it maps each arc into a pair where the first element is the source node and the second is the destination node, the two nodes have to be of different kinds. The node functions can be further classified into the following eight different types:

Inheritance: $\{\tilde{A}_c, \ddot{A}_c, \tilde{A}_s, \ddot{A}_s\}$ where

$\tilde{A}_c: T_c\rightarrow(P_c)_{MS}$ is an input class function for inheritance, a mapping from inheritance transitions to the bags of class places. *MS* stands for multi-set (or bags).

$\ddot{A}_c: T_c\rightarrow(P_c)_{MS}$ is an output class function for inheritance, a mapping from inheritance transitions to the bags of class places.

$\tilde{A}_s: T_c\rightarrow(P_c)_{MS}$ is an input state function for inheritance, a mapping from inheritance transitions to the bags of class places.

$\ddot{A}_s: T_c\rightarrow(P_c)_{MS}$ is an ouput state function for inheritance, a mapping from inheritance transitions to the bags of class places.

Predicate: $\{\tilde{O}_c, \ddot{O}_c, \tilde{O}_s, \ddot{O}_s\}$ where $\tilde{O}_c: T_r\rightarrow(P_r)_{MS}$ is an input class function for predicates, a mapping from predicates transitions to the bags of predicates.

$\ddot{O}_c: T_r\rightarrow(P_r)_{MS}$ is an output class function for predicates, a mapping from predicates transitions

to the bags of predicates.

$\tilde{O}_s$:$T_r \rightarrow (P_r)_{MS}$ is an input state function for predicates, a mapping from predicates transitions to the bags of predicates.

$\ddot{O}_s$:$T_r \rightarrow (P_r)_{MS}$ is an output state function for predicates, a mapping from predicates transitions to the bags of predicates.

C:$P \rightarrow \Sigma$, a *colour function*, it maps each place into a colour set,

E:$A \rightarrow$ *expression*, an arc expression function, It is defined from A into expressions such that $\forall a \in A$: $[Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \subseteq \Sigma]$ where $p(a)$ is the place of $N(a)$, where *MS* stands for multi-set (or bags),

I:$P \rightarrow$ *expression*, an initialization function. It is defined from P into closed expressions such that: $\forall p \in P$:$[Type(I(p)) = C(p)_{MS}]$.

DEFINITION 3.2. For each transition $t_j \in T$ in a net N,

$$\tilde{O}_s(t_j) \cap \ddot{O}_s(t_j) \neq \varnothing,$$
$$\tilde{O}_c(t_j) \cap \ddot{O}_c(t_j) = \varnothing,$$
$$\tilde{A}_c(t_j) \cap \ddot{A}_c(t_j) \neq \varnothing,$$
$$\tilde{A}_s(t_j) \cap \ddot{A}_s(t_j) = \varnothing,$$

such that

$$p_i \in \tilde{O}_s(t_j) \Rightarrow p_i \in \ddot{O}_s(t_j),$$
$$p_i \in \tilde{O}_c(t_j) \Rightarrow p_i \notin \ddot{O}_c(t_j),$$
$$p_i \in \tilde{A}_c(t_j) \Rightarrow p_i \in \ddot{A}_c(t_j),$$
$$p_i \in \tilde{A}_s(t_j) \Rightarrow p_i \notin \ddot{A}_s(t_j),$$

DEFINITION 3.3. A binding of a transition t is a function b defined on Var(t), such that: $\forall v \in Var(t)$:$b(v) \in Type(v)$ where Var(t) denotes the set of variables in a transition and B(t) denotes the set of all bindings for t.

DEFINITION 3.4. A token element is a pair (p,c) where $p \in P$ and $c \in C(p)$, while a binding element is a pair (t,b) where $t \in T$ and $b \in B(t)$. The set of all token elements is denoted by TE while the set of all binding elements is denoted by BE.

DEFINITION 3.5. 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 which is obtained by evaluating the initialization expressions: $\forall (p,c) \in TE$:$M_0(p,c) = I(p)(c)$. The markings of a SCCPN can be further classified into the following two different types: $\{M_c, M_s\}$ where M_c represents markings of the class tokens, and M_s represents markings of the state tokens.

DEFINITION 3.6. A step Y is enabled in a marking M iff the following property is satisfied: $\forall p \in P$: $\sum_{(t,b) \in Y} E(p,t)<b> \leq M(p)$ where $E(p,t)$ is the expression of (place, transition) and $E(t,p)$ is the expression of (transition, place). The summation indicates the addition of expressions. Expression<b> denotes the binding of the specific expression with a set of constants b. When $(t,b) \in Y$, this denotes that t is enabled in M for the binding b. When (t_1,b_1), $(t_2,b_2) \in Y$ and $(t_1,b_1) \neq (t_2,b_2)$, this denotes that (t_1,b_1) and (t_2,b_2) are concurrently enabled. If E=1, we refer this specific step as inheritance step (i.e. the presence of a token element will enable the step).

DEFINITION 3.7. When a step Y is enabled in a marking M_1 it may occur, changing the marking M_1 to another marking M_2, defined by:

$$\forall p \in P: M_2(p) = (M_1(p) - \sum_{(t,b) \in Y} E(p,t)<b>) + \sum_{(t,b) \in Y} E(t,p)<b>)$$

The first sum is the removed tokens while the second is the added tokens. M_2 is directly reachable from M_1 by the occurrence of the step Y, which can be denoted as $M_1[Y>M_2$.

DEFINITION 3.8. A finite occurrence sequence is a sequence of markings and steps:

$M_1[Y_1>M_2[Y_2>M_3...M_n[Y_n>M_{n+1}$ such than $n \in$ Natural Number and $M_i[Y_i>M_{i+1}$ for all $i \in 1...n$. The marking M_1 is called the start marking of the occurrence sequence, while the marking M_{n+1} is called the end marking. The non-negative integer n denotes the number of steps in the occurrence sequence, or the length of it.

DEFINITION 3.9. A marking M″ is reachable from a marking M′ iff there exists a finite occurrence sequence having M′ as start marking and M″ as end marking, i.e. iff for some $n \in N$ there exists a sequence of steps $Y_1,Y_2Y...Y_n$ such that: $M_1[Y_1>M_2[Y_2>M_3...{}_n>M''$. M″ is reachable from M′ in n steps. A firing or occurrence squence is denoted by

$$\sigma = (Y_1, Y_2 ..... Y_n)$$

The set of markings which are reachable from M′ is denoted by [M′>.

DEFINITION 3.10. The full occurrence graph of a SCCPN is the directed graph OG=(V, A, N) where:

(1) $V = [M_0>$
(2) $A = \{(M_1,b,M_2) \in V \times B E \times V | M_1[b>M_2\}$.
(3) $\forall a = (M_1,b,M_2) \in A$: $N(a) = (M_1,M_2)$.

In OG, a node is a particular marking reachable from M_0. The set of markings which are reachable from M_0 is denoted by $[M_0>$. An arc a with $N(a) = (M_1,M_2)$ is said to go from the source node M_1 to the destination node M_2. An arc with the binding element b is denoted by (M_1,b,M_2).

The occurrence graph (O-graph) has a node for each reachable marking and an arc for each step that occurs - with a single binding element. The source node of the arc is the start marking of the step, while the destination node is the end marking.

4. A TAXONOMY OF ANOMALIES OF HES

Although the integration of a Rule- and Frame-based Expert System can take the advantages of both representation paradigms. The systems are not free from errors and anomalies.

Given that in a closed world situation in which a common concept is derived by a HES (Shiu et al., 1995a,b; 1996a,b; 1997a,b). The anomalies that are relevant to the correctness, consistency, and completeness of the HES, take the following forms:

4.1. Correctness

4.1.1. Redundancy

Case I. Conditions and Actions identical between Parent Class and Child Classes.

Rule 1:$A \wedge B \Rightarrow C$
Rule 2:$A' \wedge B' \Rightarrow C'$

(A, B and 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).

The SCCPN representation of Rule 1 and Rule 2 is Fig. 1.

Initially, if we have a Parent token in Parent Class with both A and B being True, then Rule 1 will fire, and a Parent token will be created in State C with A, B and C being True. At the same time, a Child token will be created in Child Class, having both A′ and B′ being True, because of inheritance. This enables Rule 2, and after firing, a Child token is also created in State C with C′ being True.

Case II. Chained inference

Rule 1:$A \Rightarrow C$
Rule 2:$A' \Rightarrow B'$
: :
Rule N:$N' \Rightarrow C'$

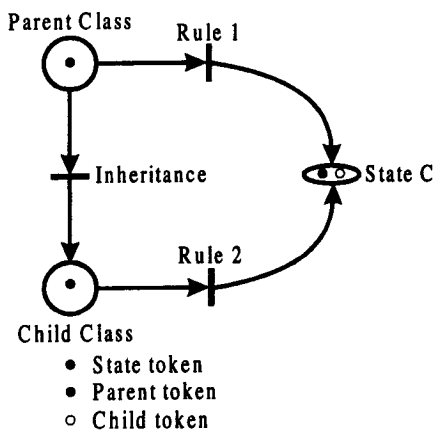

FIGURE 1. SCCPN showing Redundancy Case I.

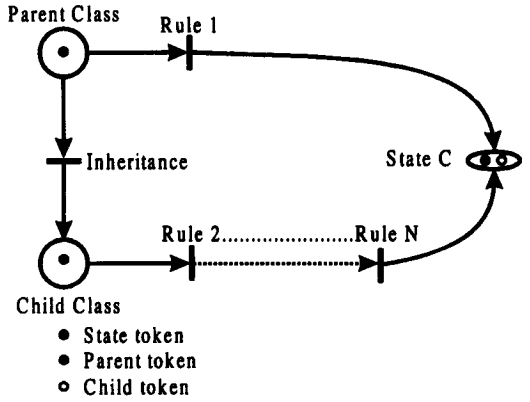

FIGURE 2. SCCPN showing Redundancy Case II.

In general the chain inference can be represented by the above SCCPN in Fig. 2.

Initially, if we have a Parent token in Parent Class with both A and B being True, then Rule 1 will fire, and a Parent token will be created in State C with both A, B, and C being True. After the chain inference from Rule 2 to Rule N, a Child token will be created in State C with A′, B′... and C′ being True.

4.1.2. Subsumption

Case I. Rule 1 is subsumed by Rule 2 (condition part) between Parent Class and Child Classes.

Rule 1:$A \wedge B \Rightarrow C \wedge D$
Rule 2:$A' \Rightarrow C' \wedge D'$

Initially, if we have a Parent token in Parent Class with both A and B being True, then Rule 1 will fire, and a Parent token will be created in State C and State D with A, B, C and D being True. At the same time, a Child token will be created in Child Class, having both A′ and

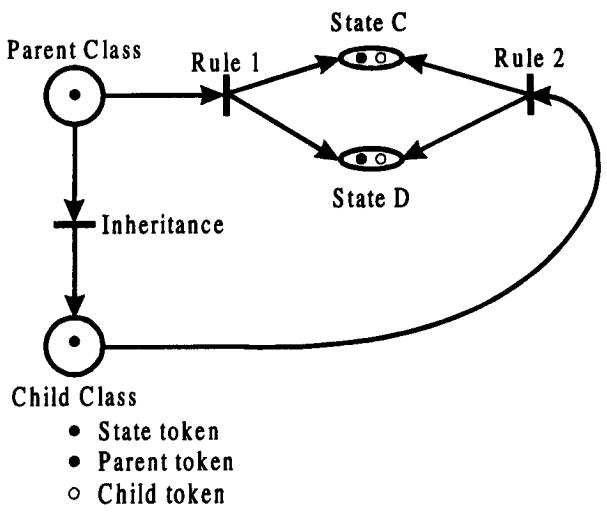

FIGURE 3. SCCPN showing Subsumption Case I.

B′ being True, because of inheritance. This enables Rule 2, and after firing, a Child token is also created in State C and State D with C′ and D′ being True.

Case II. Rule 1 is subsumed by Rule 2 (action part) between Parent Class and Child Classes.

Rule 1:A∧B⇒C∧D
Rule 2:A′∧B′⇒C′

Case III. Rule 1 is subsumed by Rule 2 (condition and action) between Parent Class and Child Classes.

Subsumption Case II and Case III are both represented by Fig. 4. Initially, if we have a Parent token in Parent Class with both A and B are True, then Rule 1 will fire, and a Parent token will be created in State C and State D with A, B, C and D being True. At the same time, a Child token will be created in Child Class, and having both A′

Rule 1 : A∧B⇒C∧D
Rule 2 : A'⇒ C'

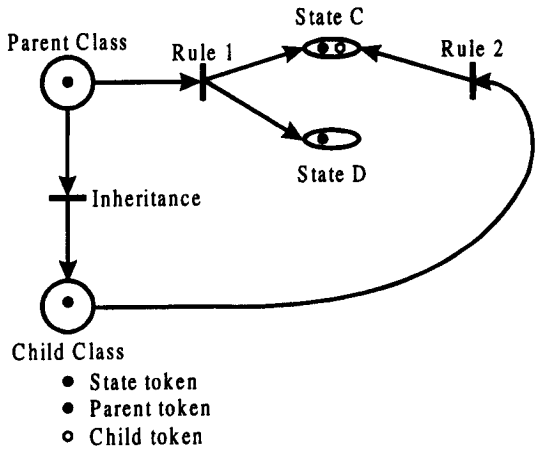

FIGURE 4. SCCPN showing Subsumption Cases II and III.

and B′ being True, because of inheritance. This enables Rule 2, and after firing, a Child token is also created in State C with C′ being True.

4.1.3. Ambiguity

Case I. Rule with inclusive disjunction of IS-A conditions from different Object Classes.

Rule 1: A IS-A member of Class X v A IS-A member of Class Y⇒C

Rules with inclusive disjunction of IS-A conditions from different Object Classes can be represented in a slightly different fashion. In Fig. 5, assertion of either IS-A Class X or IS-A Class Y or both will result in State C being asserted. Owing to the ambiguous condition of the rule involved, the rule can be unfolded into three optional sub-rules, each of which is represented by an alternative set of markings. i.e.

Rule 1a: A IS-A member of Class X⇒C
Rule 1b: A IS-A member of Class Y⇒C

Case II. Rule with inclusive disjunction of IS-A Actions for different Object Classes.

Rule 1: C⇒A IS-A member of Class XvA IS-A member of Class Y

Rules with inclusive disjunction of IS-A actions from different Object Classes can be represented by the alternative sets of marking as shown in Fig. 6. Firing of the rule will infer the assertion of either IS-A Class X or IS-A Class Y or both. In general, when a HES enters into this indeterminate situation, some sort of selection tactics would have to be executed by the system to choose the best alternative it could have. This requires a greater

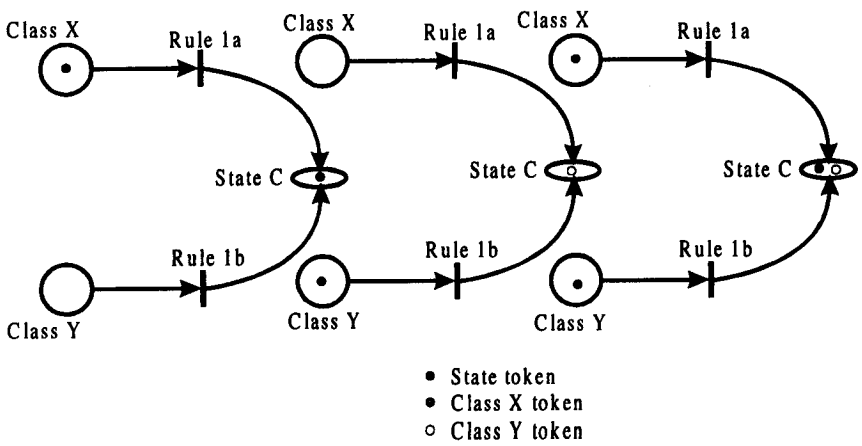

FIGURE 5. SCCPN showing Ambiguity Case I.

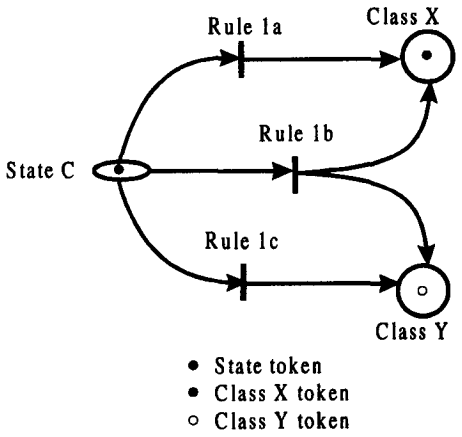

FIGURE 6. SCCPN showing Ambiguity Case II.

degree of strategy evaluation. i.e.

Rule 1a: C⇒A IS-A member of Class X
Rule 1b: C⇒A IS-A member of Class X∧A IS-A member of Class Y
Rule 1c: C⇒A IS-A member of Class Y

4.1.4. Circular Rule Sets

Case I. Self-reference rule

Rule 1: A′⇒A∧C

A SCCPN representation of this self-reference rule using a typical example (e.g. If X is a University Student THEN X is a Student AND X has a Student Identity Card) as in Fig. 7. Here, Student includes the Sub-Class University Student, therefore, the firing of Rule 1 will continue to create Parent tokens in Parent Class, and this forms a circular loop.

Case II. Self-reference chain of inference

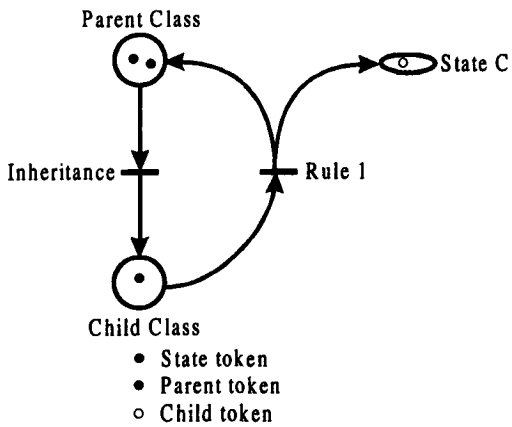

FIGURE 7. SCCPN showing Circular Rule Sets Case I.

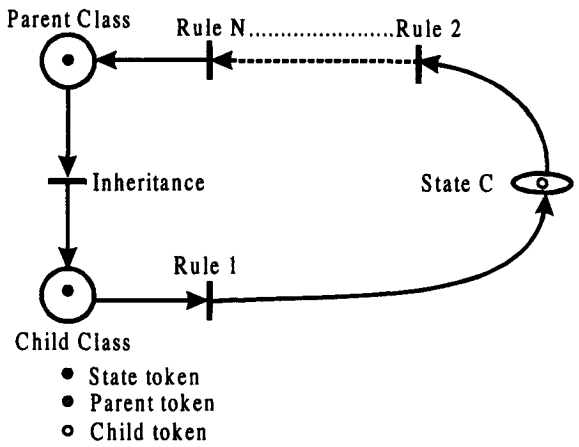

FIGURE 8. SCCPN showing Circular Rule Sets Case II.

Rule 1: B′⇒C′
Rule 2: C′⇒D′
: :
Rule N: N′⇒B

In general the self-reference chain of inference can be represented by the above SCCPN in Fig. 8.

4.2. Consistency

4.2.1. Contradiction

Case I. Self-contradictory rule

Rule 1: A⇒C
Rule 2: A′⇒¬C

Initially, if we have a Parent token in Parent Class A is True, then Rule 1 will fire, and a Parent token will be created in State C with both A and C being True. At the same time, a Child token will be created in Child Class, having A′ being True, because of inheritance. This enables Rule 2, and after firing, a Child token is also created in State C but with C′ being FALSE.

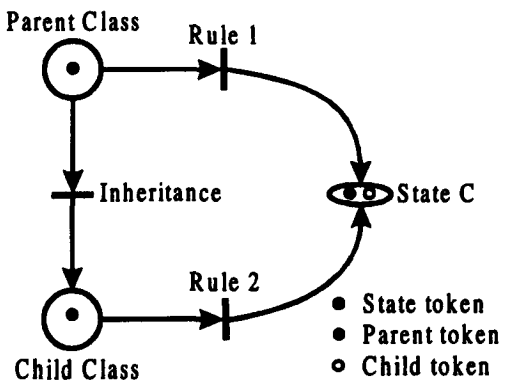

FIGURE 9. SCCPN showing Contradiction Case I.

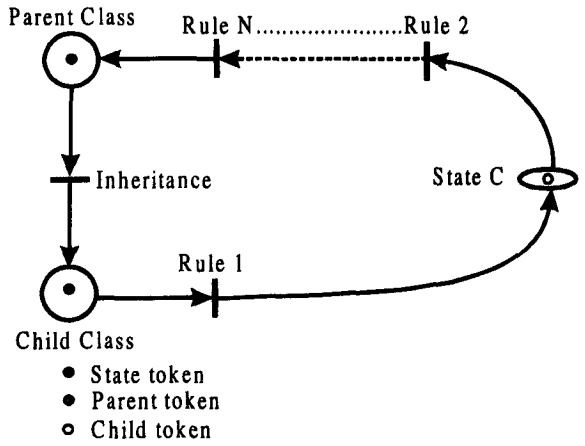

FIGURE 10. SCCPN showing Contradiction Case II.

Case II. Self-contradictory chain of inference

Rule 1: $B' \Rightarrow \neg C$
Rule 2: $C' \Rightarrow D$
: :
Rule N: $N' \Rightarrow B$

Initially, if we have a Parent token in State C with C is True, then Rule 2 will fire, after the chain inference from Rule 2 to Rule N, a Parent token will be created in Parent Class with B being True. After inheritance, a Child token will be created in Child Class with B′ being True, and this will enables Rule 1 to fire. This time, the State C is asserted to be FALSE by Rule 1 contradicting to the initial fact C which is TRUE.

Case III. Contradictory pairs of rules

Rule 1: $A \wedge B \Rightarrow C$
Rule 2: $A' \wedge B' \Rightarrow \neg C$

If we have a Parent token in Parent Class with A is TRUE, and a State token in State B indicating State B is

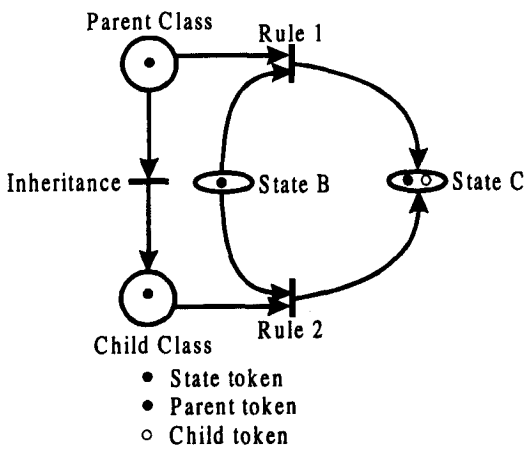

FIGURE 11. SCCPN showing Contradiction Case III.

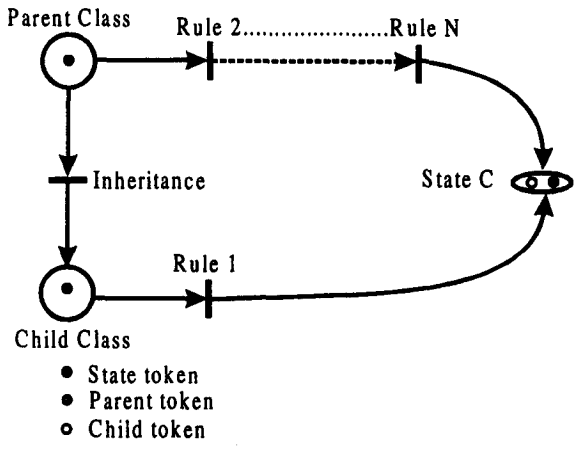

FIGURE 12. SCCPN showing Contradiction Case IV.

TRUE, State C will be asserted to be TRUE by Rule 1 but FALSE by Rule 2 indicating contradictory state of inference.

Case IV. Contradictory chains of rules

Rule 1: $A' \Rightarrow \neg P$
Rule 2: $A \Rightarrow B$
: :
Rule N: $N \Rightarrow P$

4.2.2. Deadend. 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 (Fig. 13).

Rule 1: $A \Rightarrow B$

A is not defined in the slot of the class hierarchy.Since A is not defined, no tokens will be created in State A.

4.2.3. Unnecessary IF Condition

Rule 1: $X \wedge A \Rightarrow B$
Rule 2: $X' \wedge B \Rightarrow C$

When rule 2 is backward chained to rule 1, (i.e. in order that C is true, we have to check whether B is true and X′ is true). Rule 2 is equivalent to the testing of X′, X and A, (Rule 2):

Rule 1+Rule 2: $X' \wedge X \wedge A \Rightarrow C$

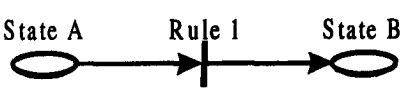

FIGURE 13. SCCPN showing Deadend.

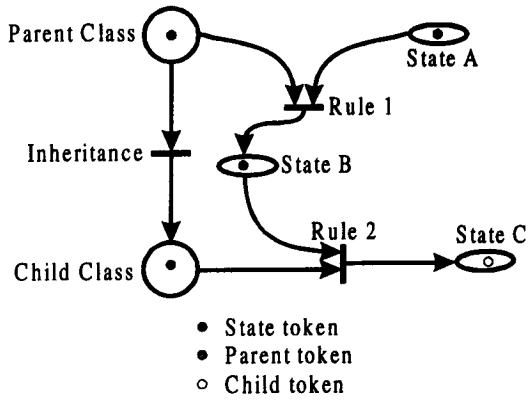

FIGURE 14. SCCPN showing Unnecessary IF Condition.

Since X′ and X are in inheritance relation, we may want to remove either the condition IF X′ or IF X (Fig. 14).

4.3. Completeness

4.3.1. Unreachability

Case I. Mutually exclusive classes, (a rule with two or more IS-A condition statements in its antecedent part)

Case Ia

Rule 1: ClassA ∧ ClassA′⇒C (applied to Parent Class)
Rule 2: ClassA ∧ ClassA′⇒C (applied to Child Class)

Rules with mutually exclusive classes can be represented by the alternative sets of rules in Fig. 15. Rule 1 will check all Parent tokens deposited in the Parent Class to see if they are also Child tokens. Similarly, Rule 2 will check all Child tokens deposited in the Child Class to see

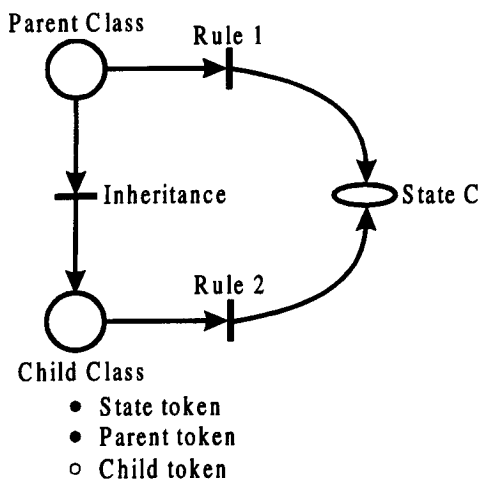

FIGURE 15. SCCPN showing Unreachability Case Ia.

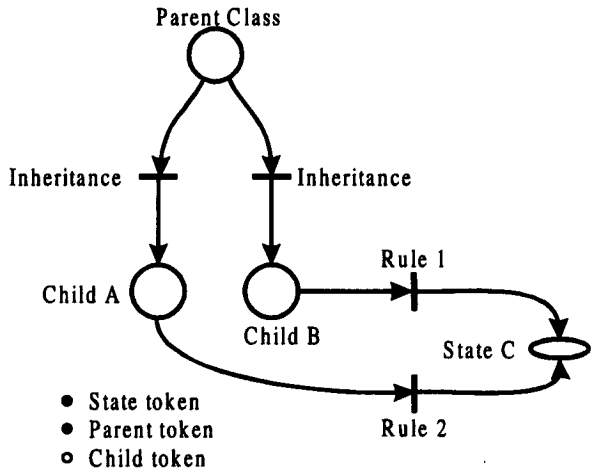

FIGURE 16. SCCPN showing Unreachability Case Ib.

if they are also Parent tokens. This will be unsuccessful, and State C will never be asserted TRUE. In general, when a HES enters into this unreachable state, some sort of selection tactics would have to be executed by the system to choose the best alternative it could have or the modeller have to review which class instantiation is more appropriate for the system.

Case Ib

Rule 1: ClassA ∧ ClassB⇒C (applied to Class A)
Rule 2: ClassA ∧ ClassB⇒C(applied to Class B)

Similar to the previous case, Child Class A and Child Class B are both children of the Parent Class, it is not possible for any object instance to be both belonging to

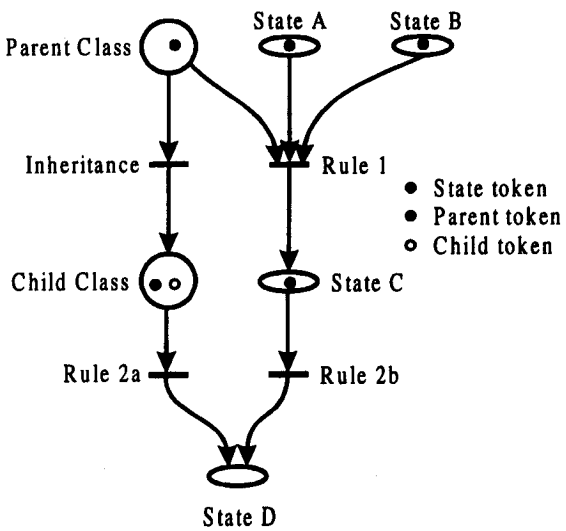

FIGURE 17. SCPPN showing Unreachability Case II.

two different mutually exclusive classes.

Case II. Mutually exclusive classes chains

Rule 1: Class $X \wedge A \wedge B \Rightarrow C$
Rule 2a: Class $X' \wedge C \Rightarrow D$ (applied to Class X')
Rule 2b: Class $X' \wedge C \Rightarrow D$ (applied to State C)

If Rule 2b is backward chained to Rule 1, this causes an unreachable condition because Rule 2b's condition part and Rule 1's condition parts are having mutually exclusive class instantiation.

5. FORMAL VERIFICATION OF THE CORRECTNESS, CONSISTENCY AND COMPLETENESS PROBLEMS

Altogether, eight propositions are defined for representing the formal properties in the SCCPN in which each of them corresponds to some anomalies in the HES. There are totally thirty-four forward and converse case proofs correspond to the seventeen anomalies cases described in section four. The detail proofs of all these propositions can be obtained from (Shiu et al., 1997b).

5.1. Correctness

5.1.1. Redundancy. Proposition 5.1. For a given marking M_0, that minimally enables a nontrivial transition sequence σ_i, iff the HES has incorrect rules causing redundancy between the parent and child object classes, then $\exists\sigma_j$, $\exists k$, such that these sequences have the following properties:

(i) $\sigma_i \cap \sigma_j = \varnothing$;
(ii) $T_c \cap \sigma_i = \varnothing; T_c \cap \sigma_j \neq \varnothing$;
(iii) $M' = \delta(M_0, \sigma_i)$, $M'' = \delta(M_0, \sigma_j)$;
(iv) $M_{sk} = 0$, $M'_{sk} > 0$, $M''_{sk} > 0$;
(v) $M_{ck} = 0$, $M'_{ck} > 0$, $M''_{ck} > 0$;
(vi) $\exists (p_{rk}, c_{ck})' \in M'_{ck}$, $\exists (p_{rk}, c_{ck})'' \in M_{ck}''$
(vii) $(p_{rk}, c_{ck})' = (p_{rk}, c_{ck})''$

Explanation: Property (i) denotes that there should exist two nontrivial transition sequences and they are disjoint one another. Property (ii) denotes that transition sequence σ_i does not involve any inheritance while transition sequence σ_j involves inheritance. Property (iii) denotes that marking M′ is reachable from initial marking M_0 by the first sequence σ_i and marking M″ is reachable from M_0 by the second sequence σ_j. Property (iv) denotes that no state token is deposited in Place k in the initial marking. While in markings M′ and M″, there is at least one state token deposited in Place k. Property (v) is similar to (iv) except that the markings are referring to class tokens. Property (vi) denotes that there exists a class token element $(p_{rk}, c_{ck})'$ in predicate place k of M′. There is also a class token element $(p_{rk}, c_{ck})''$ which exists in predicate place k of marking M″. Property (vii) tells us that the colour (data value) of this two class tokens in predicate k are the same.

5.1.2. Subsumption. Proposition 5.2. For a given marking M_0, that minimally enables a nontrivial transition sequence σ_i, iff the HES has incorrect rules causing subsumption between the parent and child object classes, then $\exists\sigma_j$, $\exists k$, such that these sequences have the following properties:

(i) $\sigma_i \cap \sigma_j = \varnothing$;
(ii) $T_c \cap \sigma_i = \varnothing; T_c \cap \sigma_j \neq \varnothing$;
(iii) $M' = \delta(M_0, \sigma_i)$, $M'' = \delta(M_0, \sigma_j)$;
(iv) $M_{sk} = 0$, $M'_{sk} > 0$, $M''_{sk} > 0$;
(v) $M_{ck} = 0$, $M'_{ck} > 0$, $M''_{ck} > 0$;
(vi) $\exists (p_{rk}, c_{ck})' \in M'_{ck}$, $\exists (p_{rk}, c_{ck})'' \in M_{ck}''$
(vii) $(p_{rk}, c_{ck})'' \subseteq (p_{rk}, c_{ck})'$

5.1.3. Ambiguity. Proposition 5.3. For a given marking M_0, that minimally enables $\Gamma = \{\sigma_i, \sigma_j\}$ for a nontrivial transition sequence σ_i, σ_j, iff the HES has incorrect rules causing ambiguous conditions of events between different object classes, then $\exists k$, $\forall p_{rk} \in \ddot{O}_s(\Gamma)$, $\forall p_{rk} \in \ddot{O}_c(\Gamma)$, such that these sequences have the following properties:

(i) $\sigma_i \cap \sigma_j = \varnothing$;
(ii) $M' = \delta(M_0, \sigma_i)$, $M'' = \delta(M =, \sigma_j)$;
(iii) $M_{sk} = 0$, $M'_{sk} \geq 1$, $M''_{sk} > 1$;
(iv) $M_{ck} = 0$, $M'_{ck} \geq 1$, $M''_{ck} > 1$;
(v) $\exists (p_{rk}, c_{ck})' \in M =_{ck}$, $\exists (p_{rk}, c_{ck})'' \in M_{ck}''$
(vi) $(p_{rk}, c_{ck})' = (p_{rk}, c_{ck})''$

5.1.4. Circular Rule Sets. Proposition 5.4. For a given marking M^0, that minimally enables transition sequence α, iff the HES has incorrect rules causing cyclicity between the parent and child object classes, then $\exists_j \geq i$, $\exists k$ such that the sequence has the following properties:

(i) $M^i \in [M^0> = \{M^0, M^1, M^2, M^i, ..M^j\}$,
(ii) $M^j = \delta(M^0, \alpha)$ for $j > 0$,
(iii) $T_c \cap \alpha \neq \varnothing$;
(iv) $M^i_{sk} = 0$, $M^i_{sk} > 0$, $M^j_{sk} > 1$;
(v) $M^i_{ck} = 0$, $M^i_{ck} > 0$, $M^j_{ck} > 0$;

5.2. Consistency

5.2.1. Contradiction. Proposition 5.5. For a given marking M_0, that minimally enables a nontrivial transition sequence σ_i, iff the HES has incorrect rules causing contradiction between the parent and child object classes, then $\exists\sigma_j$, $\exists k$, such that these sequences have the following properties:

(i) $\sigma_i \cap \sigma_j = \varnothing$;
(ii) $T_c \cap \sigma_i = \varnothing; T_c \cap \sigma_j \neq \varnothing$;
(iii) $M' = \delta(M_0, \sigma_i)$, $M'' = \delta(M_0, \sigma_j)$;
(iv) $M_{sk} = 0$, $M'_{sk} > 0$, $M''_{sk} > 0$;
(v) $M_{ck} = 0$, $M'_{ck} > 0$, $M''_{ck} > 0$;
(vi) $\exists (p_{rk}, c_{ck})' \in M'_{ck}$, $\exists (p_{rk}, c_{ck})'' \in M_{ck}''$
(vii) $(p_{rk}, c_{ck})' = \neg (p_{rk}, c_{ck})''$

5.2.2. Deadend. The problem of deadend is not caused by any conflict in the rule set attached to the object hierarchy, but by the inaction of some events (or conditions). In other words, it only causes concern if the execution of the deadend rule fails to achieve any goal state which belongs to a collection of terminating goals under a specific domain of inference. Consequently, in any SCCPN simulation of HES inference that determines its consistency, we need to achieve a finite termination upon any given state, yet satisfy the goal states.

Let the collection of goal states be Ω, we define that, for a deadend rule, λ, $\exists p_{rk}$, such that $p_{rk} \in \ddot{O}_s(\lambda)$, $p_{rk} \in \ddot{O}_c(\lambda)$, and $p_{rk} \notin \Omega$ and $\neg \exists t_i$ such that $p_{rk} \in \tilde{O}_s(t_i)$ and $p_{rk} \in \tilde{O}_c(t_i)$.

Proposition 5.6. Iff the rule set has inconsistent rules that involve deadend applied to the object hierarchy, then $\exists$ a marking M such that $M_{sk} = 0$ for $\forall p_{rk} \in \Omega$, and $\forall \sigma_j$ where $M' = \delta(M, \sigma_j)$, $M'_{sk} = 0$.

5.2.3. Unnecessary IF Condition. Proposition 5.7. For a given marking M_0, that minimally enables $\Gamma = \{\sigma_i, \sigma_j\}$ for a nontrivial transition sequence σ_i, σ_j, iff the HES has incorrect rules causing unnecessary IF conditions between the parent and child object classes, then $\exists Y$ (a step y), such that these sequences have the following properties:

(i) $\sigma_i \cap \sigma_j = \varnothing$;
(ii) $T_c \cap \sigma_i = \varnothing; T_c \cap \sigma_j \neq \varnothing$;
(iii) $M' = \delta(M_0, \sigma_i)$, $M'' = \delta(M_0, \sigma_j)$;
(iv) $\exists Y$, $\sum_{(t,b) \in Y} E(p,t) < b > \leq (M' \cup M'')$
(v) $\exists (p_r, c'_c) \in M'_c$, $\exists (p_r, c''_c) \in M_c''$, $c'_c \cap c''_c \neq \varnothing$;

5.3. Completeness

5.3.1. Unreachability. Proposition 5.8. Iff the rule set has incomplete rules that involve unreachability applied to the object hierarchy, then $\forall$ marking M, such that $M_{sk} = 0$ for $p_{rk} \in \Gamma \subseteq \Omega$, and $\forall \sigma_j$ where $M' = \delta(M, \sigma_j)$, $M'_{sk} = 0$.

6. COMPLEXITY ANALYSIS OF SCCPN METHODOLOGY

6.1. Complexity Issues

Modelling and verifying an Expert System, in particular, its knowledge base, is a complex process. The extent of that complexity has some readily identifiable costs. For instance, in a more complex system, we can anticipate a much longer time for testing the knowledge base, hence, a relatively high maintenance and management cost for the system. In addition, it is likely that the quality of the system is a function of this complexity. Such a problem is complicated further due to the weakness in human performance on complex inference tasks. As a result of these cost and quality issues, it is important that the complexity of any methodology developed which purports for modelling and verifying the behaviour of a system can be measured so that determinants of that complexity can be monitored and managed through further investigations or so. Complexity is described by (Bundy, 1997) as "the measurement of some aspect of the complexity of the current Problem State in a search problem. For instance, the depth of a goal is the length of the path from the current goal to the origin of the Search Space. Complexity measures are sometimes associated with the labels of nodes in a search space, especially when these are logical expressions describing the current goal, e.g. the depth of function nesting of an expression is the maximum amount of nesting in the functions in it. The size of an expression is the number of symbols in it. These symbols can also be weighted and the weights totalled". According to (Someren, 1997), many problems can be represented as an initial state, a goal state and a set of operators that define operations to go to new states from a given state. The states that can be reached from the initial state by applying the rules in all possible ways define the state space. The problem is then to reach the goal state from the initial state.

The criteria of interest for model evaluation include adequacy of representation, ability of the representation scheme to recognize problems correctly, the ability to formulate an algorithm to detect the errors, and the efficiency of the algorithms.

6.2. Measuring Complexity

There is substantial reason to suggest that the underlying structure of the knowledge base is a major component of complexity (O'Leary, 1991). In fact, the set of components in an Expert System (i.e. user interface, database interface, inference engine and knowledge base) allows for the same set of interfaces and inferences to be used in many different situations. Thus, complexity of the methodology for modelling Expert Systems comes from constructing and processing the knowledge base. One of the primary vehicles from which the structural nature of a component of knowledge can be assessed is network theory, alternatively referred to as graph theory. The State Controlled Coloured Petri Nets (SCCPNs) model has adapted well founded mathematical net theory with a number of extensions. Consequently it can provide a measure of the complexity of the process that involves a transformation of a Hybrid Expert System into a SCCPN network.

The structural complexity of a knowledge base in a

HES refers to the extent to which interaction between production rules within the object-hierarchy makes the process of representing the knowledge complex. This depends on a number of factors that could be determined as follows:

6.2.1. The Number of Object-Classes in the Frame Hierarchy. The number of object classes and their hierarchical relationships characterizes the size of the Frame hierarchy. Quite a few object classes with a large number of rules attached to them will generate a verification task of comparable complexity to another employing a large number of object classes with a smaller number of rules. In addition, although it is undesirable to introduce ambiguity to the knowledge base as a result of any existing indeterminate rules, an intention to partition these rules in order to determine their individual possible effects may honor such practice. This, however, will inevitably increase the complexity of the model transformation.

6.2.2. The Size of the Rule Set and Its Connectivity. Connectivity among rules attempts to evaluate the relatedness which constitutes the rule chains and search paths. This connectivity is in some manner reflected by the ability to create partitioned subsets of the rules which are relatively but not completely disjoint. The SCCPN is subjected to greater effort of verification in the case of higher degrees of interconnection in the rule set attached to one particular object class family (i.e. Father, Son and grandson).

The number of rules increases, the number of possible interactions between rules increases exponentially (Chen & Suen, 1994), the complexity of the potential matches for each pattern in a rule increases and the number of possible combinations of factors required for testing the patterns increases exponentially.

6.2.3. The Depth of Reasoning Structure. The depth of the reasoning structure is characterized by the length of inference chains in the HES. This determines the scope of the verification task. Longer chains introduce more transitions, increase the computation effort for reachability in the representation, and makes the checking of SCCPN network a more complex task.

6.2.4. The Nature of Semantic Information. The semantic information utilized by the verification procedures relates to the number of mutually exclusive sets of input facts attached to individual object token that govern the firing of the transition. Larger sets of such may impair the performance of the algorithms. On the other hand, semantic information required to be passed over for any transition firing and operation may incur overheads for the verification process. The analysis could be further complicated when commonsense reasoning including deterministic, probabilistic and stochastic estimates of individual situations are taken into consideration. An extensive analysis that covers all of these situations, however, is beyond the scope of this research. Consequently, attention is limited to cases which are deterministic and applied to well defined sets of input object tokens.

6.3. Complexity Analysis

The complexity of verifying the anomalies in knowledge base, in the context of this paper, is defined to include the effort to transform the rules and object hierarchy into transitions, to derive the reachability tree, to check the markings and the token colours for error examination.

6.3.1. Transformation of Rules and Object Hierarchy to SCCPN. Let the Rule-based part of the HES have k rules, each with u conditions and v actions. It is required to create predicate transitions to match rules. There can be a maximum of $k(u+v)$ predicate places representing $2k(u+v)$ possible colour tokens (depicted by the presence of the object token and the state token), and $k+c$ predicate transitions representing rules where c is the extra number of transitions created as a result of possible indeterminate rules in the HES. It is noted that $c = 0$, if there exists none of this type of rule explicitly in the rule set. However, rules of this nature may exist implicitly in the knowledge base, presenting inter-related properties of redundancy and subsumption.

Let the Frame-based part of the HES have m object classes. There can be a maximum of m object class places and 2m possible colour tokens (depicted by the presence of the object token and the state token), and $(m-1)$ inheritance transitions.

The transition sequence, σ, will be represented by a n–vector where n is the number of transitions (predicate as well as inheritance) in the SCCPN. n is derived through the transformation by

$$n=(k+c)+(m-1)$$

Let 2p denote the number of token facts, with $2p \leqslant 2k(u+v) + 2m$. The total number of storage places, S, for the computation, therefore, will be

$$S=2p+n$$

Each storage place will have a colour type, which was defined by the object class type. (i.e. each object class type will have different slot numbers, slot size, etc and therefore require a different data type for storage).

More storage places will be needed if any additional transitions and operations are included for the SCCPN simulation.

6.3.2. Derivation of Occurrence Graph. The basic idea behind Occurrence Graphs is to construct a graph which

has a node for each reachable marking and an arc for each occurring binding element. Obviously, such a graph may become very large, even for small SCCPNs. They may grow exponentially with respect to the number of independent processes, (i.e. if a system has n independent processes each of which can be in m states the full Occurrence Graphs (state space) have m^n nodes (states)). However, recent research (Li et al., 1993; Christensen & Petrucci, 1995; Kemper, 1996; Kondratyev et al., 1996) has been taken to allow for a partial examination of a subportion of the reachability graph, therefore reduce the efforts in deriving possible solutions. The main idea of the above methods is to apply the concept of clustering / partition to the analysis of the Occurrence Graphs. Large systems (such as HES) may consist of a set of modules. Local Properties of each module can be checked separately, before checking the validity of the entire system, hence reducing the complexity of the state space of the entire system. (e.g. A SCCPN may be divided formally into a set of sub-nets, each sub-net is called a module, and performs independent analysis). Other techniques (Jensen, 1995) for limiting the size of the Occurrence Graphs include (1) Occurrence Graphs with Equivalence Classes; (2) Occurrence Graphs with Symmetries; (3) Place Invariants and (4) Transition Invariants.

However, the development of the partition algorithms, theories of sub-net analysis and reduction methodologies for Occurrence Graphs are beyond the scope of this research, therefore, we concentrate our analysis by adequately initiation of the sequence of transitions and closely examining the reachability markings in the full Occurrence Graphs.

The following algorithm is usually used for generating the (Occurrence Graph) reachability set of a SCCPN:

```
Reachability Set={M0}, where M0 is the initial
  marking
  Reachability Graph={}
  UnfiredMarkingList=[M0]
  repeat
    select some marking M in the
    UnfiredMarkingList
    for each transition t which is enabled at M
    do begin
    generate marking M' which results from
    firing t at M
    if M' is not an element of ReachabilitySet
    then begin
      add M' to ReachabilitySet
      append M' to UnfiredMarkingList
      end
    add arc(M,T,M') to ReachabilityGraph
  end
until UnfiredMarkingList is empty
```

In most automated SCCPN simulations, the first element of the UnfiredMarkingList is always selected, and so the reachability graph is produced in breadth-first order.

In verifying the HES against the problems of correctness, consistency, and completeness, we use an automated computer aid (e.g. DESIGN/CPN) for the generation of the reachability set. The SCCPN is initialized by placing tokens in the place and setting the values of data variables. The operation of the net can be investigated by the program either in a step by step manner or in an automatic mode.

6.3.3. Heuristic Search Method of Occurrence Graph for Particular Marking. The checking of the irregularities and anomalies in HES requires exhaustively or heuristically an adequate 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. Therefore, some guided search strategy is necessary for reducing the computational complexity. It is essential that if we are to investigate whether a given marking is reachable from an initial marking, we have to construct the reachability tree, but the complete construction and exhaustive search are not efficient methods in general. Knowledge of the structure of the SCCPN can be used to limit the search of the reachability tree and heuristics can be used to reduce the search space. We purposed the follow heuristic based on the concept of clustering (Mehrotra, 1991).

(1) Put the Start Node [M0] in a list called HIS
(2) If HIS is empty, exit with failure ELSE continue
(3) Select the leftmost marking M in HIS
(4) For each transition t which is enabled at M, calculate the distance metric of all the enabled transitions using the formula:

$$D(r_i, r_j) = \frac{\textit{Total no. of literals in rule ri and antecedent of rj}}{\textit{No. of overlapping literals in rule ri and antecedent of rj}}$$

where D(ri,rj) is the distance metric

(5) generate a priority list of transitions with increasing distance (i.e. the top transitions will have the highest score)
(6) generate marking M' which results from firing the transitions which have the minimal distance in the distance metric
(7) closely examine the reachability markings in M' for detection of anomalies using the Propositions 5.1 to 5.8.
(8) If M' is not an element in HIS then add M' to HIS, add arc(M,t,M') to HIS
(9) Goto Step 4
(10) Until no transition is enabled in M

Using the distance matrix as the evaluation function, the search algorithm for a particular marking changed from breadth-first search to heuristic search. Rules with closer distance are having larger chances of anomalies, and therefore should be checked first. Using the above algorithm should reduce the time to search through the occurrence graph for location of errors and anomalies, nevertheless, an exhaustively search still have to be done in order to guarantee an error free knowledge base.

7. CONCLUSION

In this paper, we have described a formal description technique based on State Controlled Coloured Petri Nets to model hybrid (rule- and frame-based) expert systems. The technique allows the use of reachability theory for the verification of the systems. The paper illustrates the capability of the technique to identify the anomalies due to the incorrectness, inconsistency or incompleteness of the hybrid knowledge base. The verification was done exhaustively by minimally initiating any sequence of transitions and closely examining the reachability markings at each transition.

Future work will include (1) Extension of Methodology for Modelling Hybrid Expert Systems with Uncertainty; (2) Extension of Methodology for Modelling Hybrid Expert Systems with Temporal Properties; (3) Extension of Methodology for Modelling Hybrid Expert Systems with Case-Based Systems, and (4) Extension of Methodology for Modelling Hybrid Expert Systems with Conventional Software Systems.

Acknowledgement—I would like to thank Prof. T. S. Dillon of La Trobe University for his helpful comments to this paper. This research project is supported by the staff development programme of the Hong Kong Polytechnic University.

REFERENCES

Agarwal, R., & Tanniru, M. (1992). A Petri net based approach for verifying the integrity of production systems. *International Journal of Man-Machine Studies*, **36**, 447–468.

Ayel, M., & Vignollet, L. (1994). SYCOJET and SACCO, Two tools for verifying expert systems. *International Journal of Intelligent Systems*, **9**, 357–382.

Bundy, A. (1997) *Artificial intelligence techniques*, ed. A. Bundy, 4th Edition, p. 43. Springer Verlag

Chang, C. L. et al., (1990). A report on the Expert Systems Validation Associate (EVA). *Experts Systems with Applications*, **1**(3), 219–230.

Charles, E. (1991). Checking knowledge bases for inconsistencies and other anomalies. In *Workshop Notes from the Ninth National Conference in Artificial Intelligence, AAAI-91, Knowledge-Based Systems Verification, Validation and Testing*, 17 July, Anaheim CA.

Chen, Z., & Suen, C. Y. (1994). Measuring the complexity of rule-based expert system. *Expert Systems with Applications*, **7**(4), 467–481.

Christensen, S., & Petrucci, L. (1995) Modular state space analysis of coloured Petri nets. In G. De Michelis and M. Diaz (Eds.) *Application and theory of Petri nets 1995*, (pp. 201-217)

Coenen, F., & Bench-Capon, T. (1993) *Maintenance of knowledge-based systems*. Academic Press.

Cragen, B. J., & 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*, **26**, 633–648.

Cuda, T. V., & Dolan, C. P. (1991) Automating the refinement of knowledge-based systems. *Proceedings ECAI-90*, Stockholm, August 6-10, pp. 167-172.

Culbert, C. (1994) *NASA MMU-FDIR system*, Lyndon B. Johnson Space Centre. USA.

Duchessi, P., & O'Keefe, R. M. (1995). Understanding expert systems success and failure. *Expert Systems with Applications*, **9**(2), 123–133.

French, S. W., & Hamilton, D. (1994). A comprehensive framework for knowledge-base verification and validation. *International Journal of Intelligent Systems.*, **9**, 809–837.

Gamble, R. F. et al., (1994). Applying formal verification methods to rule-based programs. *International Journal of Expert Systems.*, **7**(3), 203–239.

Geissman, J. R., & Schultz, R. D. (1988). Verification and validation of expert systems. *AI Expert*, **3**(2), Feb., 26–33.

Ginsberg, A. (1988) Knowledge-base reduction: A new approach to checking knowledge bases for inconsistency and redundancy. In *Proc. 7th National Conference on Artificial Intelligence (AAAI-88)*, pp. 585–589.

Gupta, U. G. (1991) *validating and verifying knowledge-based systems*. IEEE Computer Society Press.

Gupta, U. G. (1993). Validation and verification of knowledge-based systems: a survey. *Journal of Applied Intelligence*, **3**, 343–363.

Jacob, R. J. K., & Forscher, J. N. (1991). A Software engineering methodology for rule-base systems. *IEEE Transactions on Knowledge and Data Engineering*, **2**(2), 173–189.

Jensen, K. (1995) *Coloured Petri nets: basic concepts, analysis methods and practical use*, Vol. 2. Springer-Verlag.

Jensen, K. (1996) *Coloured Petri nets: basic concepts, analysis methods and practical use*, Vol. 1, 2nd edn. Springer-Verlag.

Jensen, K. (1997) *Coloured Petri nets: basic concepts, analysis methods and practical use*, Vol. 3. Springer-Verlag.

Kemper, P. (1996) Reachability analysis based on structured representations. In J. Billington and W. Reisig (Eds.), *Application and theory of Petri nets 1996* (pp. 269-288).

Kondratyev, A. et al. (1996) A structural approach for the analysis of petri nets by reduced unfoldings. In J. Billington and W. Reisig (Eds.), *Application and theory of Petri nets 1996* (pp. 346-365).

Landauer, C. (1990). Correctness principles for rule-based expert Systems. *Expert Systems with Applications*, **1**(3), 291–317.

Laurent, J. P., & Ayel, M. (1989) Off-line coherence checking for knowledge based systems. In *IJCAI-89 Workshop Proceedings on Verification, Validation and Testing of Knowledge-Based Systems*, Detroit.

Li, X., Lai, R., & Dillon, T. S. (1993) 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.

Liu, N. K. (1991) Formal description and verification of expert systems. Ph.D. dissertation, Department of Computer Science and Computer Engineering, School of Mathematical and Information Sciences, La Trobe University, Bundoora, Victoria, Australia, 1991.

Liu, N. K. (1993) Formal description technique for the verification of fuzzy knowledge base redundancy and subsumption. In *IEEE Proceedings of the 1st New Zealand International Conference on Artificial Neural Networks and Expert Systems*, Dunedin, New Zealand, November, pp. 142–145, 1993.

Liu, N. K. (1996). Formal verification of some potential contradictions in knowledge base using a high level net approach. *Applied*

Intelligence, **6**, 325–343.

Liu, N. K., & Dillon, T. S. (1995). Formal description and verification of production systems. *International Journal of Intelligent Systems*, **10**, 399–442.

Matsumoto, K. et al. (1991) A dynamic verification method for knowledge-based systems. In *Proceedings IFIP WG 54/IFAC Workshop on Dependability of AI Systems (DASIY-91)*, Vienna, Austria, 27–29, May, 1991.

Mehrotra, M. (1991) Rule groupings: a software engineering approach towards verification of expert systems. NASA Contractor Report 4372, Washington, DC.

Messeguer, P. (1994). The VALID Project: goal, development, and results. *International Journal of Intelligent Systems*, **9**, 867–892.

Meseguer, P. (1990) A new method to checking rule bases for inconsistency: a petri net approach. In *Proceedings of ECAI-90, 9th European Conference on Artificial Intelligence*, pp. 437-442.

Murrell, S., & Plant, R. (1995). Formal semantics for rule-based systems. *J. Systems Software*, **29**, 251–259.

Nazareth, D. L. (1993). Investigating the applicability of Petri nets for rule-based system verification. *IEEE Transactions on Knowledge and Data Engineering.*, **4**(3), 402–415.

Nguyen, T. A. et al., (1987). Knowledge base verification. *AI magazine*, **8**(2), 69–75.

Nguyen, T. A. et al. (1985) Checking an expert system knowledge base for consistency and completeness. In *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, Los Angeles, CA.

O'Keefe, R. E., & O'Leary, D. E. (1993). Expert system verification and validation: A survey and tutorial. *Artificial Intelligence Review.*, **7**, 3–42.

O'Leary Daniel, E. (1995). The impact of semantic ambiguity on Bayesian weights. *European Journal of Operational Research*, **84**, 163–169.

O'Leary, D. E. (1988). Methods of validating expert systems. *Interfaces*, **18**(6), 72–79.

O'Leary, D. E. (1991) Measuring and managing complexity in knowledge-based systems: a network and mathematical programming approach. In D. E. Brown and C. C. White, (Eds), *Operations research and artificial intelligence: the integration of problem-solving strategies.*

Preece, A. D. (1989) Verification of rule-based expert systems in wide domains. In N. Shadbolt (Ed.), *Research and development in expert systems VI (Proc. Expert Systems 89)*, (pp. 66–77). Cambridge University Press.

Preece, A. D. (1991). Practical issues in specifying expert systems. *Intelligent Systems Review*, **2**(3/4), 3–26.

Preece, A. D., & Shinghal, R. (1991a) COVER: a practical tool for verifying rule-based systems. knowledge-based systems verification, validation and testing. Workshop Notes from the 9th National Conference on Artificial intelligence, AAAI-91, Anaheim CA, 17 July, 1991.

Preece, A. D., & Shinghal, R. (1991b) DARC: a procedure for verifying rule-based systems. In J. Liebowitz (Ed) *Expert Systems World Congress Proceedings*, Vol. 2, (pp. 971–979). Pergamon Press.

Preece, A. D. et al., (1996). Validating dynamic properties of rule-based systems. *Int. Journal Human-Computer Studies*, **44**, 145–169.

Rousset, M. C. (1988). On the consistency of knowledge bases: The COVADIS system. *Computation Intelligence*, **4**, 166–170.

Scarpelli, H., & Gomide, F. (1994). A high level net approach for discovering potential inconsistencies in fuzzy knowledge bases. *Fuzzy Sets and Systems*, **64**, 175–193.

Scarpelli, H., & Gomide, F. (1994) Consistency checking based on high level fuzzy petri nets. In *Proceedings of 3rd IEEE conference on Fuzzy Systems*, **3**, 1957-1962.

Shiu, S. C. K. (1997b) Formal description techniques for the verification of expert systems, Ph.D. thesis, Department of Computing, Hong Kong Polytechnic University, Hong Kong.

Shiu, S. C. K. et al. (1995a) Modelling hybrid rule/frame-based expert systems using coloured Petri nets. In *Proceedings of 8th International Conference on Industrial and Engineering Applications of AI and ES*, Melbourne, Australia, pp. 525–532.

Shiu, S. C. K. et al. (1995b) An approach towards the verification of hybrid rule/frame-based expert systems using coloured Petri nets. In *Proceedings of 1995 IEEE International Conference on System Man and Cybernetics*, Vancouver, pp. 2257–2262.

Shiu, S. C. K. et al. (1996a) An approach towards the verification of fuzzy hybrid rule/frame-based expert systems using coloured Petri nets. In *Proceedings of ECAI-96 Workshop in Validation, Verification and Refinement of KBS*, Budapest, pp. 105–113.

Shiu, S. C. K. et al. (1996b) Detection of anomalies of hybrid rule/frame-based expert systems using coloured Petri nets. *Australian Journal of Intelligent Information Processing Systems*, **3**(3), 59–76.

Shiu, S. C. K. et al., (1997a) Formal verification of the correctness in hybrid expert systems. In *Proceedings of The First International Conference on Conventional and Knowledge-Based Intelligent Electronic Systems*, KES' 97, 21st - 23rd May, 1997, Adelaide, Australia, Vol. 2, pp. 419–428.

Shiu, S. C. K. et al., (1997c) Formal verification of some potential contradictions in hybrid expert systems. In *Proceedings of the 1997 IEEE International Conference on Systems, Man, and Cybernetics*, October 12-15, 1997, Orlando, Florida, USA, pp. 4424–4429.

Shortliffe, E. H. (1976) *Computer-based medical consultations: MYCIN*, New York: Elsevier.

Someren, M. van (1997) *Artificial intelligence techniques*. A. Bundy (Ed.) 4th Edn, Springer Verlag, p. 262.

Stachowitz, R. A., & Chang, C. L. (1988) Verification and validation of expert systems. Tutorial note at AAAI-88.

Stachowitz, R. A., & Combs, J. B. (1987) Validation of expert systems. *Proceedings of the 20th Annual Hawaii International conference on Systems Sciences*, pp. 686–695.

Suwa, M. et al. (1982) An approach to verifying completeness and consistency in a rule-based expert system. *AI Magazine*, pp. 16-21.

Terano, T. (1994). The JIPDEC checklist-based guideline for expert system evaluation. *International Journal of Intelligent Systems*, **9**, 893–952.

Wu, C., & Lee, S. J. (1995) Knowledge validation with an enhanced high level Petri net model. In *Proceedings of 11th Conference on Artificial Intelligence for Applications*, pp. 126–132.

Yao, Y. (1994). A Petri net model for temporal knowledge representation and reasoning. *IEEE Transactions on SMC*, **24**(9), 1374–1382.

Zhang, D., & Nguyen, D. (1994). PREPARE: a tool for knowledge base verification. *IEEE Transactions on Knowledge and Data Engineering*, **6**(6), 983–989.