Detection of Anomalies 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|csnkliu|csdaniel}@comp.polyu.edu.hk

Abstract

High level Petri Nets have recently been used for many Artificial Intelligence 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 additional anomalies caused by the integration of the object hierarchy, property inheritance and production rules are defined. The detection and analysis of the anomalies of proposed model is done by constructing and examining the reachability tree spanned by the knowledge inference. An algorithm is also given to allow for the generation of such a reachability set of the nets. Our approach can provide formal verification of the correctness, consistency, and completeness of the hybrid knowledge base. A practical expert system for personnel selection is used throughout this paper to illustrate the application of our approach.

Keywords: *Hybrid Expert System, Knowledge Verification, Coloured Petri Nets*

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 [15][16][22] and [29]. 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][8][17][19][26]. 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 the current expert systems are being developed [7][20][28]. 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 [3]'s Normal Form approach, [5][27]'s Decision Table Method, [12]'s Incidence Matrix Method, [6]'s Truth Maintenance Systems and [2][25]'s 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)[11], 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 and analyzing of hybrid rule/frame-based expert systems for the detection of anomalies. We will introduce an approach based on CPN plus the concept of state tokens[18] 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 seven main sections. Next section describes the knowledge representation and inference of a hybrid expert system, the third section gives the definitions of CPN and illustrates how HES can be modelled by CPN. Section four uses a practical expert system as an example to illustrate the potential errors and anomalies in HES due to the integration of rules with inheritance. Section five describes the algorithm for generation of the reachability set of the CPN. Section six describes the methods for detection of anomalies in the HES by analyzing the net properties concerned. 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 analyzed using three conceptual views [9] 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 [23][24], 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, methods and demons 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 using rules. Therefore, in HES, the Frame base can be seen as being used to define the vocabulary for the Rule base, i.e. the 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 value within the same object instance, besides, this demon function: f(a) generates only one output from each given input "a",).
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 or integer. 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 the associated 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 some 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 include : Pattern Matching, Unification, Resolution and Heuristic Search.

3 Modelling the HES Using CPN

3.1 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 : $P \cap T = P \cap A = T \cap A = \emptyset$.
N =	a node function. It is defined from A into $PxT \cup TxP$.
C =	a colour function. It is defined from P into Σ.
G =	a guard function. It is defined from T into expressions such that: $\forall t \in T$:$[Type(G(t))=B \wedge Type(Var(G(t))) \subseteq \Sigma]$
E =	an arc expression function. It is defined from A into expressions such that : $\forall a \in A$: $[Type(E(a))=C(p(a))_{ms} \wedge Type(Var(E(a))) \subseteq \Sigma]$.
I =	an initialization function. It is defined from P into closed expressions such that $\forall p \in P$: $[Type(I(p))=C(p)_{ms}]$.

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).

3.2 Hybrid Expert System Model

3.2.1 Object Classes

Each object class's data structure is represented by a compound colour set, and each object 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 object class's 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, e.g. Color OO = Product AA * BB * CC. Each object class 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 three variables, object "a", which is a particular instance of a Super Class A, object "a1", which is a particular instance of Class A. (ie. "a" IS-A superclass instance while "a1" IS-A class instance) and State "s" which is the state token. State "s" is used to carry the information that identify which object instance had fired from which transition. (i.e. var a : OO, var a1 : OO and var s : text string)

3.2.2 Rules with Inheritance

In CPN, the transition operations are represented by the arc expression functions. By defining the arc expression functions differently, it can help us to model different events in the HES. Therefore, 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, 1b and 1c)

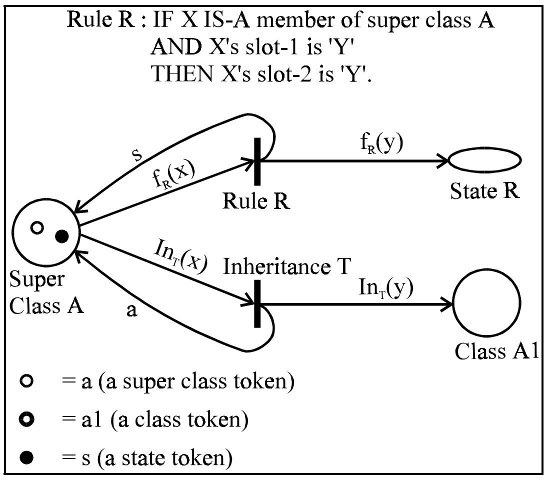

Figure 1a : Rule R with Inheritance (before firing) with an input token “a” & “s” in Super Class A.

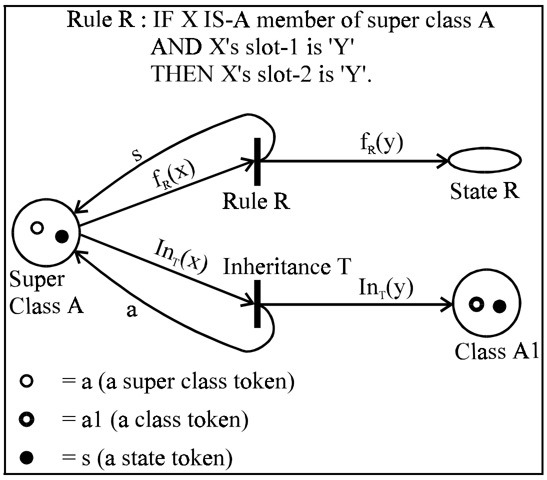

Figure 1b : Rule R with Inheritance (after firing Inheritance T) with an input token “a” & “s” in Super Class A.

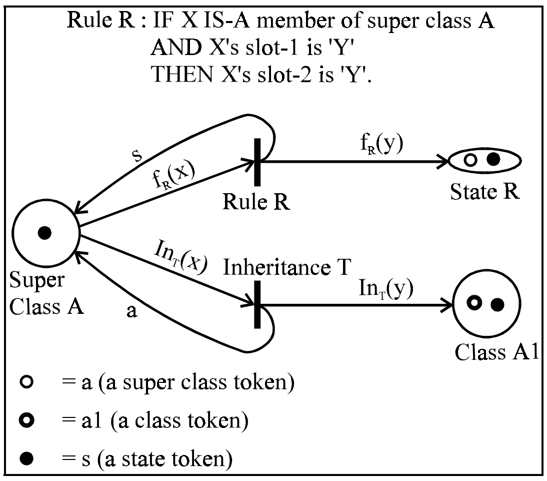

Figure 1c : Rule R with Inheritance (after firing both Rule R and Inheritance T) with output token “a” & “s” in State R and output token “a1” & s” in Class A1. A state token “s” is also created in Super Class A.

Super Class A is a CPN Place with colour set that was used to represent the data structure of all object instances in Super Class A. Class A1 is a CPN Place with colour set that was used to represent the data structure of all object instances in Class A1. Rule R is a CPN Transition which is enabled iff the input arc expression $f_R(x)$ is evaluated to be true (i.e., the premise X IS-A member of super class A AND X’s slot-1 is ‘Y’ is true). If $f_R(x)$ is true then Rule R is fired, it implies that Rule R is executed. All tokens will be removed from Super Class A and a new token “a” will be created in State R with new data values determined by the output arc expression $f_R(y)$. (i.e. $f_R(y)$ will assign 'Y' to X’s slot-2). Inheritance T is a CPN Transition which is enabled whenever there is an “a” token in Super Class A, after firing this transition, a token “a1” is created in Class A1 with all the attributes inherited from A. (ie. a child token is created with the same attributes of its father). These two tokens (“a”, “a1”) can be used for further inference (if any) in the HES. In this way, we can trace the execution path of these two tokens by examining the information carried by the state tokens created within the CPN network. Moreover, we can also examine the contents of this two tokens to see if any attributes are in conflict with each other. These could serve as an indication of the existence of anomalies within the HES. A detail description of this detection method is given in section four later. (Note that in order to preserve the state of the predicate in Rule R, a state token is created in Super Class A via the self-loop of Rule R and an “a” token is created in Super Class A via the self-loop of inheritance T.)

3.2.3 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 object class instance’s data structure is represented by the token of a particular colour set, we can define arc expression such that they directly read and write data in the token's data slots. This can be illustrated by the following simple example: Pass the message “OK” to the object Class A’s slot-promotion.

Colour sets:

```
Color Classes = with ClassA | Class B;
Color Promotion = String;
Color Objects = product Classes * Promotion;
var x : Classes;
```

Arc expression:

```
IF x=ClassA THEN 1`(ClassA, “OK”) ELSE empty.
```

This will serve the purpose of sending or receiving messages (data value) to and from object instance for testing the rules' premises.

3.2.4 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 X'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 2a and 2b.

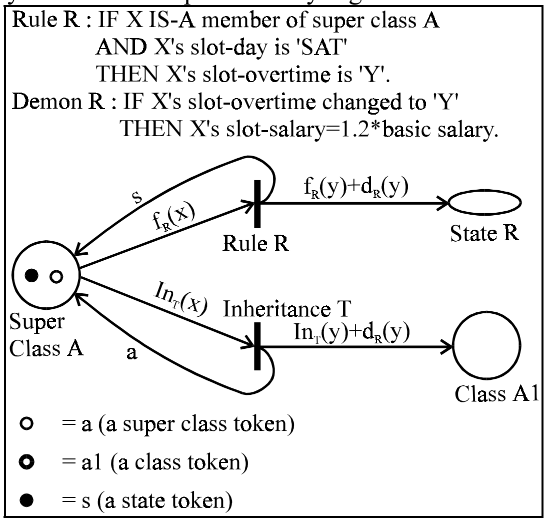
Rule R : IF X IS-A member of super class A
AND X's slot-day is 'SAT'
THEN X's slot-overtime is 'Y'.
Demon R : IF X's slot-overtime changed to 'Y'
THEN X's slot-salary=1.2*basic salary.

Figure 2a : Rule R with Demon (before firing) with an input token "a" and a state token "s" in Super Class A.

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

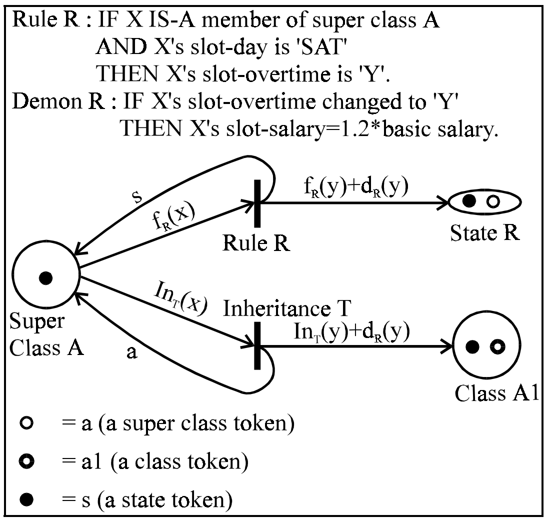
Rule R : IF X IS-A member of super class A
AND X's slot-day is 'SAT'
THEN X's slot-overtime is 'Y'.
Demon R : IF X's slot-overtime changed to 'Y'
THEN X's slot-salary=1.2*basic salary.

Figure 2b : Rule R with Demon (after firing) with output token "a" & "s" in State R and output token "a1" & "s" in Class A1. A state token "s" is also created in Super Class A.

3.2.5 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. For example, a typical function which tells whether the argument is even or not might be:

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 statments; 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 follows (Figures 3a-d, the self loops are omitted for clarity reason)

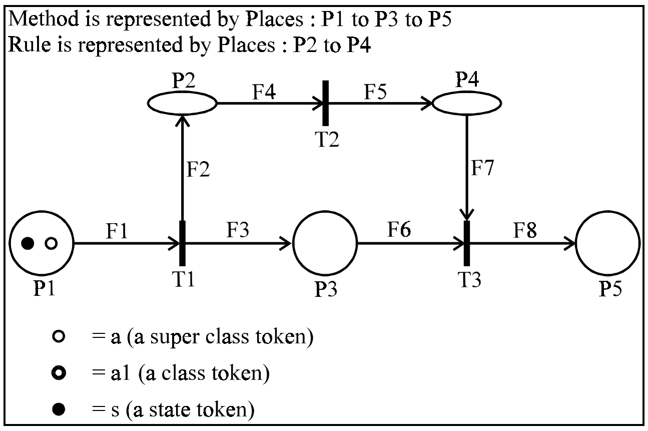
Method is represented by Places : P1 to P3 to P5
Rule is represented by Places : P2 to P4

Figure 3a : Rule with Method (before firing) with an input token "a" and a state token "s" in P1.

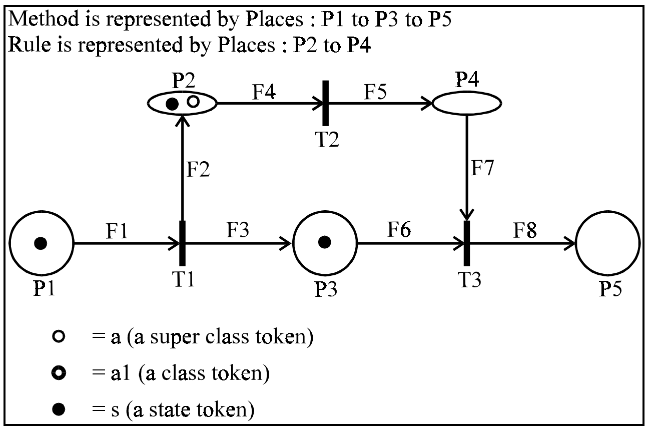
Method is represented by Places : P1 to P3 to P5
Rule is represented by Places : P2 to P4

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

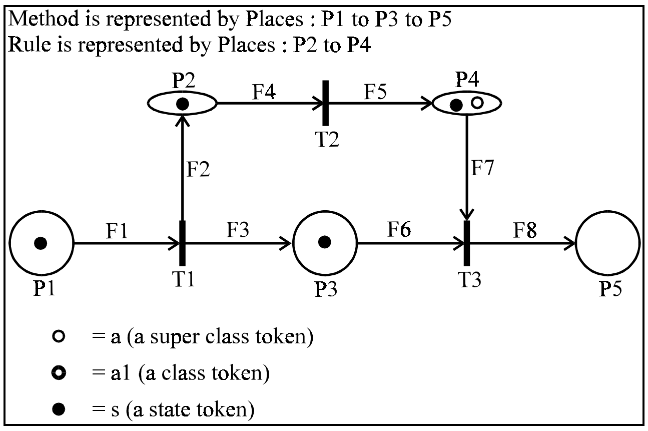
Method is represented by Places : P1 to P3 to P5
Rule is represented by Places : P2 to P4

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

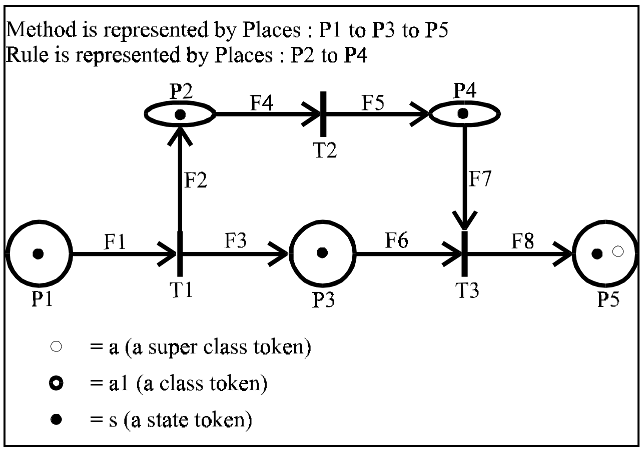
Method is represented by Places : P1 to P3 to P5
Rule is represented by Places : P2 to P4
P2
F4
F5
P4
T2
F2
F7
F1
F3
F6
F8
P1
T1
P3
T3
P5
○ = a (a super class token)
○ = a1 (a class token)
● = s (a state token)

Figure 3d : Rule with Method (Method resumes control). The token "a" was passed to P5. A state token "s" was subsequently created in P1, P2, P3, P4 and P5 respectively.

The modelling of methods is divided into two parts. First the state of the method. (i.e. (1) executed some of the program codes and waiting to pass the control to the Rule, (2) waiting for the Rule to pass back the control, (3) executed all the program codes and waiting to pass the control to other process.) Secondly, the actual program codes of the method itself. (i.e. Represented by the arc expression functions.) In Figure 3a-d, P1 to P3 to P5 represent three states of the Method describe above. P2 to P4 represent the Rule embedded within the Method. Arc expression function F1 is the first part of the Method which executes first, then control is passed to the Rule by F2 which will create the "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 "s" is deposited in all the Places, P1, P2, P3, P4 and P5 for preservation of the states.

3.2.6 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.

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. In a pure rule-based system, errors and anomalies are redundancy, dead-end rules, subsumption, duplication, circular rule sets, unsatisfiable conditions, missing rules..etc. In a pure frame-based system, error and anomalies may occur due to the problems of message passing and concurrency, problems of inheritance (including simple, repeated and multiple inheritance) and problems of polymorphism. Instead of covering all the possible errors and anomalies caused by the integration of the above two representation paradigms, we would like to focus ourselves on the additional errors and anomalies attributed to the integration of rules with inheritance of object properties. Details will be discussed in the following Section.

4 Errors and Anomalies in HES due to Integration of Rules with Inheritance

To illustrate the HES modelling by our proposed CPN methodology, we adopt a simplified version of a personnel selection expert system currently being used in Hong Kong [10]. This system is used to find out, among all the clerks in the organization, who should be promoted to senior clerk. The organization's employee data structure is represented in a frame-based hierarchy as shown in Figure 4 and details of relevant frames in the hierarchical structure are given below.

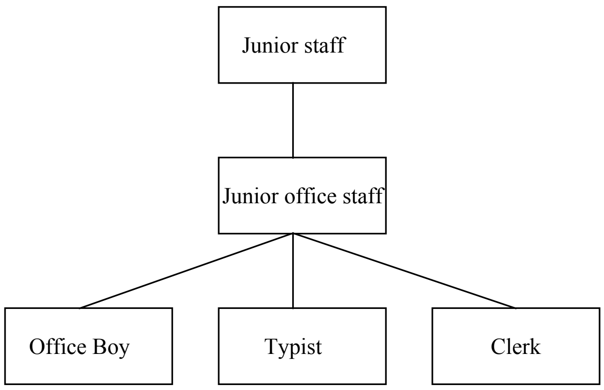

Figure 4 : The Frame Hierarchy

A Junior Staff Frame :

Slot Name	Value	Type	Demons
Job Title	Junior Staff	String	
Office Hours	9am - 5pm	Time	
Qualification Requirement	Five passes in HKCEE	String	
Salary Pay Scale	1 to 10	String	
Department	General Secretariat	String	
Annual leave	21 days	Integer	
Father Frame	-		
Son Frame	Junior Office Staff		

A Junior Office Staff Frame :

Slot Name	Value	Type	Demons
*Job Title	Junior Office Staff	String	
Name		String	
Address and Telephone		String	

HKID#		String	IF possess HKID no. THEN Privilege is Local ELSE Privilege is Overseas.
Privilege		Local/ Overseas	
Sex		M/F	
*Office Hours	9am-5pm	Time	
*Qualification Requirement	Five passes in HKCEE	String	
*Department	General Secretariat	String	
*Salary Pay Scale	1 to 10	String	
Present Salary Point		Integer	Present Salary Point must between 1 to 10 inclusive.
Years of Service		Integer	
*Annual leave	21 days	Integer	
Leave taken		Integer	
Leave balance		Integer	Leave balance = Annual leave - Leave taken
Knowledge of Work		G/M/L (Good, Medium, Low)	
Acceptance of Responsibility		G/M/L	
Organization of Work		G/M/L	
Initiative		G/M/L	
Relations with Colleagues		G/M/L	
Relations with Public		G/M/L	
Expression on Paper		G/M/L	
Oral Expression		G/M/L	
Supervisory Skills		G/M/L	
Leading Skills		G/M/L	
Performance		G/M/L	
Experience		G/M/L	
Ability		G/M/L	
Quality of Services		G/M/L	
Seniority		G/M/L	
Promotion		Yes /Wait /Reject	
Father Frame	Junior Staff		
Son Frame	Clerk, Typist and Office boy		

* denotes slots inherited from parent frame

A Clerk frame is similar to a Junior Office Staff frame except that more detailed information about the various types of Clerk duties are included such as Purchasing Clerk, Book Keeping Clerk, Sales Clerk, Inventory Clerk, Customer Services Clerk, Data Entry Clerk...etc. For the purpose of this modelling exercise, we can treat the Class Junior Office Staff as the common job grade in the organization, and the Class Clerk, Office Boy and Typist as specific job categories all belonging to the same job grade. Any new employment regulations and promotion rules that apply to Junior Office Staff grade will be applicable to all Clerks, Office Boys and Typists in the organization. The major problems of verifying this HES is due to the fact that some rules are applicable to the general class (Super Class : Junior Office Staff) and through inheritance these rules are applicable to specific classes as well (Classes : Clerks, Office Boy and Typists). Anomalies exist whenever rules specifically applied to a class are in conflict with those rules that are applied to their superclass. Furthermore, these rules may be in a subsumbed situation and some of them may be unreachable. We will illustrate how to detect them in the following sections.

First, we model the above example using our proposed methodology described in previous sections. It is noted that a frame is equivalent to a data structure with various types declarations, (or an object with different attributes). Demons are declared as methods or procedures within some frame. In the above expert system example, the two frames are Class frames. Each individual clerk's information is inferred by the creation of a clerk frame instance. The data value of Clerk Name, Sex, Address...etc are input via the user interface. The data values and demons in the slots with a * are inherited from the parent frame; the data value of Privilege and Leave balance are updated by firing the demons in HKID# and Leave balance. The data values for slots between Knowledge of Work and Leading Skills inclusively are input by the individual clerk's supervisor at the beginning of the inference process. The data value of Performance, Experience, Ability, Quality of Services and Seniority are being inferred by the execution of the rules pre-defined earlier by the personnel manager of the organization. The goal is to find out the data value of the slot Promotion, which can be inferred by forward chaining or backward chaining within the rule sets. (Over 100 rules were constructed for the original expert system based on the Multiple Criteria Decision Model). Detail data structure of a clerk token and some typical rules are given as follows:

A clerk token's colour is :

Color AA = string; (all text strings)
Color BB = with Local | Overseas; (colours explicitly specified)
Color CC = with Male | Female;
Color DD = time; (date)
Color EE = integer with 0..10;(between 0&10)
Color FF = integer;
Color GG = with Good | Medium | Low;
Color HH = with Yes | Wait | Reject;
Color II = list AA with 4; (a list of four strings)
Color JJ = list AA with 3;
Color KK = list FF with 5;
Color LL = list GG with 15;
Color MM = with Clerk | Typist | Office Boy;
Color NN = product II * BB * CC * DD * JJ * KK * LL * HH; (all tuples (i,b,c,d,j,k,l,h) where $i \in II$, $b \in BB$,....$h \in HH$)
Color OO = with Yes | No; (for state token, if the value is Yes, it denotes that the predicate is true, else if the value is No, the negation of the predicate is true.)

Var i:II; var b:BB; var c:CC; var d:DD; var j:JJ; var k:KK; var l:LL; var h:HH; var clerk : NN; (var denotes variable declaration which introduces one or more variables. Here we have one variable, clerk, which is with colour NN. We may use var clerk1, clerk2, clerk3 : NN for declaring three different clerks for example.)

Some typical rules are :

Rule 1: IF X is a junior office staff
AND X's quality of service is Good
AND X's seniority is High
THEN X's promotion is Yes.

Rule 2: IF X is a clerk
AND X's quality of service is Good
AND X's seniority is High
THEN X's promotion is Yes.

Rule 3: IF X is a clerk
AND X's quality of service is Good
AND X's seniority is High
AND X is a local citizen
THEN X's promotion is Yes.

Rule 4: IF X is a clerk
AND X's year of service is greater than Five
THEN X's seniority is Not High.

Rule 5: IF X is a junior office staff
AND X's year of service is greater than Five
THEN X's seniority is High.

Rule 6: IF X is a clerk
AND X's knowledge of work is Not Good
AND X's English is Not Good
THEN X needs to attain training course.

Rule 7: IF X is a junior office staff
AND X needs to attain training course
THEN X's experience is Low.

Rule 8: IF X is a clerk
AND X is a junior office staff
THEN X is entitled to 14 days annual leave.

Rule 9: IF X is a office boy
AND X needs to attain training course
THEN X is on Probation.

Rule 10: IF X is a junior office staff
THEN X is required to do typing.

Rule 11: IF X is required to do typing
THEN X is a clerk.

Rule 12: IF X is a clerk
THEN X is a junior office staff.

These rules can be rewritten as :

Rule 1:	$A \wedge B \wedge C \Rightarrow X$
Rule 2:	$A1 \wedge B \wedge C \Rightarrow X$
Rule 3:	$A1 \wedge B \wedge C \wedge D \Rightarrow X$
Rule 4:	$A1 \wedge E \Rightarrow \neg C$
Rule 5:	$A \wedge E \Rightarrow C$
Rule 6:	$A1 \wedge \neg F \wedge \neg G \Rightarrow Y$
Rule 7:	$A \wedge Y \Rightarrow H$
Rule 8:	$A1 \wedge A \Rightarrow K$
Rule 9:	$A2 \wedge Y \Rightarrow Z$
Rule 10:	$A \Rightarrow L$
Rule 11:	$L \Rightarrow A1$
Rule 12:	$A1 \Rightarrow A$

Where the meanings of the literals used in the above rules are as follows:

A	= Junior Office Staff
A1	= Clerk
A2	= Office Boy
B	= Quality of service is Good
C	= Seniority is High
$\neg$C	= Seniority is Not High
D	= Local citizen
E	= Years of service is greater than Five
$\neg$F	= Knowledge of work is Not Good
$\neg$G	= English is Not Good
H	= Experience is Low
K	= Entitled to 14 days annual leave
L	= Required to do Typing
X	= Promotion is Yes
Y	= Needs to attain training course
Z	= On Probation

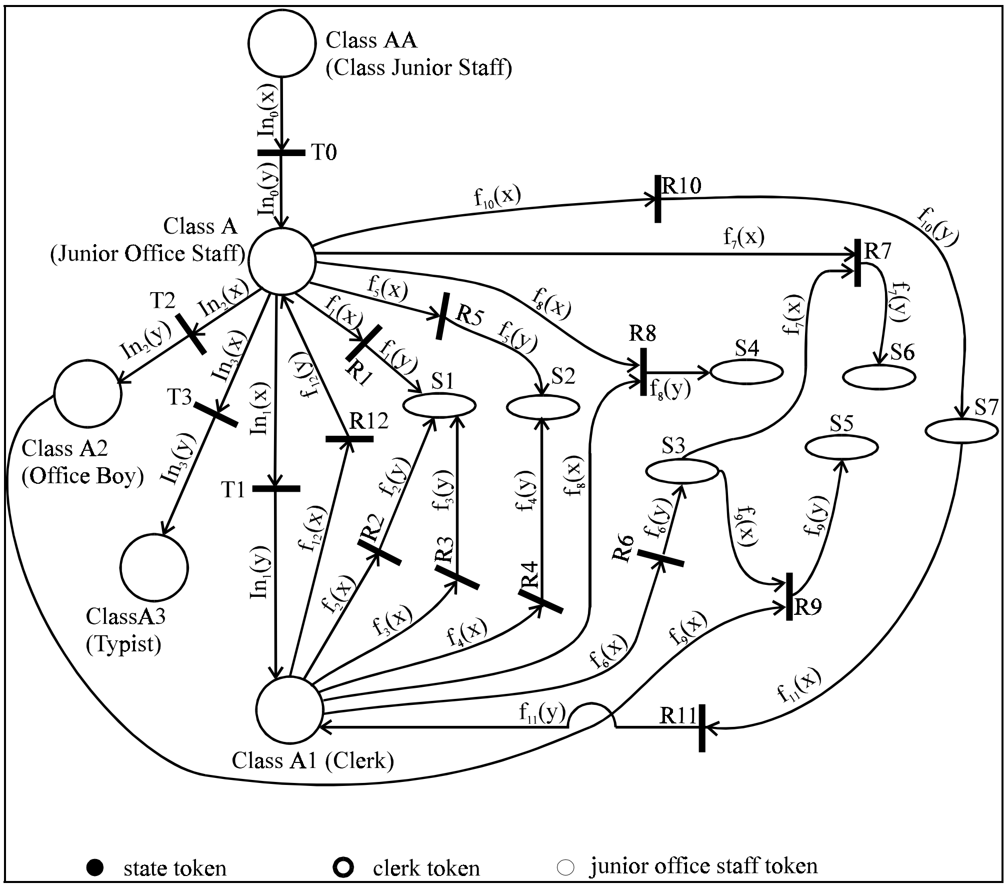

Figure 6 : CPN representation of the given HES

The Hybrid Expert System is represented by a Coloured Petri Net shown in Figure 6, according to the methodology proposed in section three. Note that for simplicity, the self-loop associated with each input place is not shown in the net. The rules are labelled R1 to R12. The inheritance relations are represented by T1 to T3. S1 to S7 represents the predicates of these rules.

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 [13] 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 (i.e. corresponds to some meaningful initial facts) and then check the reachability set for any irregularities of the associated predicate places. The checking of the irregularities and anomalies 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. Therefore, we propose the following algorithm for generating the reachability set of a CPN as follows:

```
Reachability Set = {$M_0$}, where $M_0$ is the initial
  marking
Reachability Graph ={}
UnfiredMarkingList = [$M_0$]
  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 CPN 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 for the generation of the reachability set. The CPN 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 Detection of Errors and Anomalies in HES

6.1 Correctness

6.1.1 Subsumption

Analysis of the network will show the presence of subsumption in the HES (Figure 7a). Suppose we have a Junior Office Staff with good quality of service and high seniority, we want to infer whether he should be promoted or not in our HES. This inference process will be as follow: initially, we have a Junior Office Staff token in the input place Class A (Junior Office Staff), and this token's slot "quality of service is Good" is TRUE and this token's slot "seniority is High" is also TRUE. This enables both R1 and T1 to be fired, as a result, a Clerk token is created in place Class A1 (clerk) by the T1 transition and a Junior Office Staff token is created in S1 by $f_1(y)$. Next, R2 is also enabled since R2's antecedent is the same as R1. After firing the two rules, S1 consists of both a Junior Staff Token and a Clerk token.

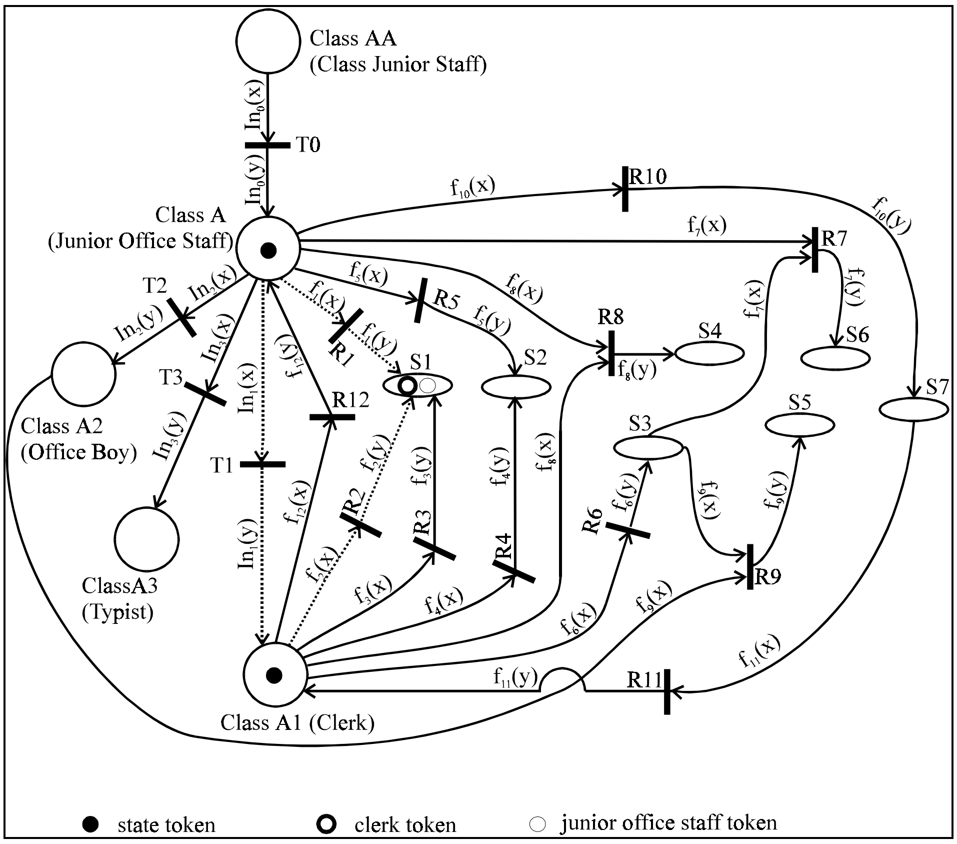

Figure 7a : CPN representation showing the events of subsumption, Case I

Figure 7b represents the reachability graph as the results of the execution of R1 and R2. The graph is a directed graph from which we can see the marking M1, M2, M3, M4 and M5 are reachable from marking M0. In marking M5, both a Clerk token and a Junior Office Staff token is created in S1, by examining the slot "promotion" in this two tokens reveals that they have the same value, ie 'YES'. Since in the place Class A1, the Clerk token inherited all his attributes from the initial Junior Office Staff token, this means that R1 and R2 are using the same set of initial attributes for inference, therefore, we can conclude that R2 subsumes R1 because R2 is just a more specific case of R1. (ie. Clerk is the child of Junior Office Staff).

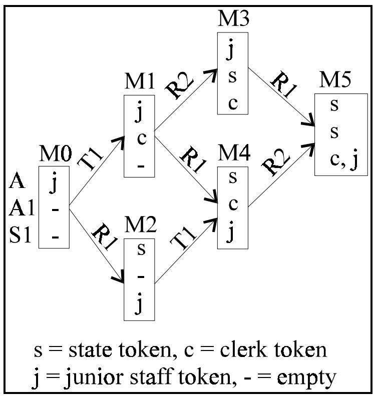
M0
M1
M2
M3
M4
M5
A
A1
S1
T1
R1
R2
s = state token, c = clerk token
j = junior staff token, - = empty

Figure 7b : Reachability graph due to the firing of R1 and R2

In general, if we have two rules:

Rule X : $A \wedge B \Rightarrow C$
Rule Y : $A' \wedge B \Rightarrow C$

If the value of slot A inherits to slot A' (i.e. A is the parent and A' is the child), then Rule Y subsumes Rule X because Rule Y is just a more specialized case of Rule X. (i.e. whenever Rule Y succeeds, Rule X 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.

Next, we consider a more complicated subsumption situation as in Figure 8a. Suppose initially, we have a junior office staff token in the input place Class A (Junior Office Staff), with slot "quality of service is Good" is TRUE, slot "seniority is High" is TRUE and slot "local citizen" is also TRUE. This enables both R1 and T1 to be fired, as a result, a Clerk token is created in place Class A1 (Clerk) by the T1 transition and a Junior Office Staff token is created in S1 by $f_1(y)$. Next, R2 and R3 are also enabled. After firing either one of the two rules, S1 consists of both a Junior Staff Token and a Clerk token.

Figure 8b represents the reachability graph as the results of the execution of Rule1 followed either by R2 or R3. Since M5 is reachable from M4 either by R2 or R3, by examining the slot "promotion" in the Clerk token and Junior Office Staff Token reveal that they have the same value, ie 'YES'. Therefore, these two rules must be in a subsumption relationship because the two transitions R2 and R3 can be enabled in A1 and their final marking is the same.

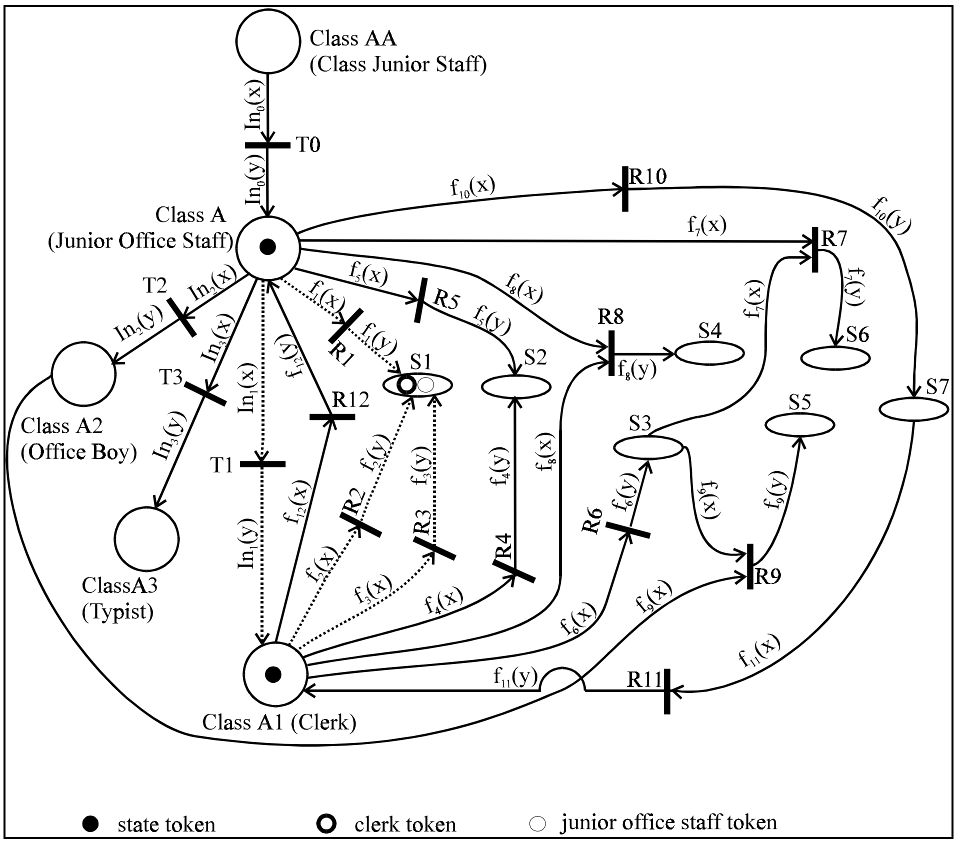

Figure 8a : CPN representation showing the events of subsumption, Case II

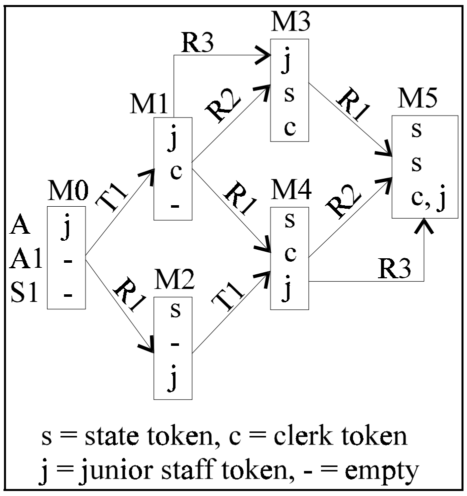
M3
R3
j
s
c
M1
R2
R1
M5
j
c
-
s
s
c, j
M0
T1
R1
M4
R2
A
A1
S1
j
-
-
s
c
j
M2
R1
T1
R3
s
-
j
s = state token, c = clerk token
j = junior staff token, - = empty

Figure 8b : Reachability graph due to the firing of R1, R2 and R3

6.1.2 Cyclicity

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

Rule X :	B⇒C
Rule Y :	C'⇒B

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

In our example, Rule 10, Rule 11 and Rule 12 will form such a cyclicity. In Figure 9a, if we have a Junior Office Staff token in Class A then R10 is enabled and fired, this will further enable R11 and a Clerk token is deposited in A1 (Clerk). As a result, R12 will be enabled and a Junior Office Staff token will be deposited in Class A. This process will continue within a loop with no end. Reachability analysis will show that there exists an infinite tree which has the branching pattern repeated after four levels. (Marking M7, M13 and M12 are repeated)

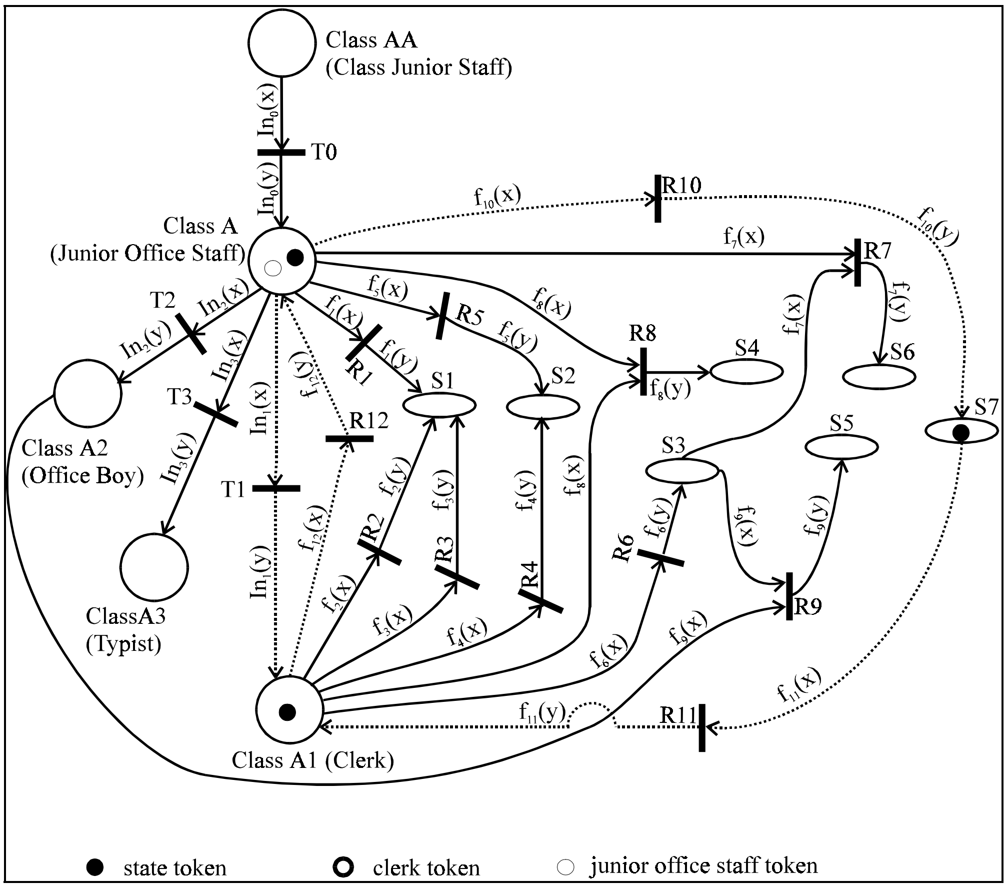

Figure 9a : CPN representation showing the events of cyclicity.

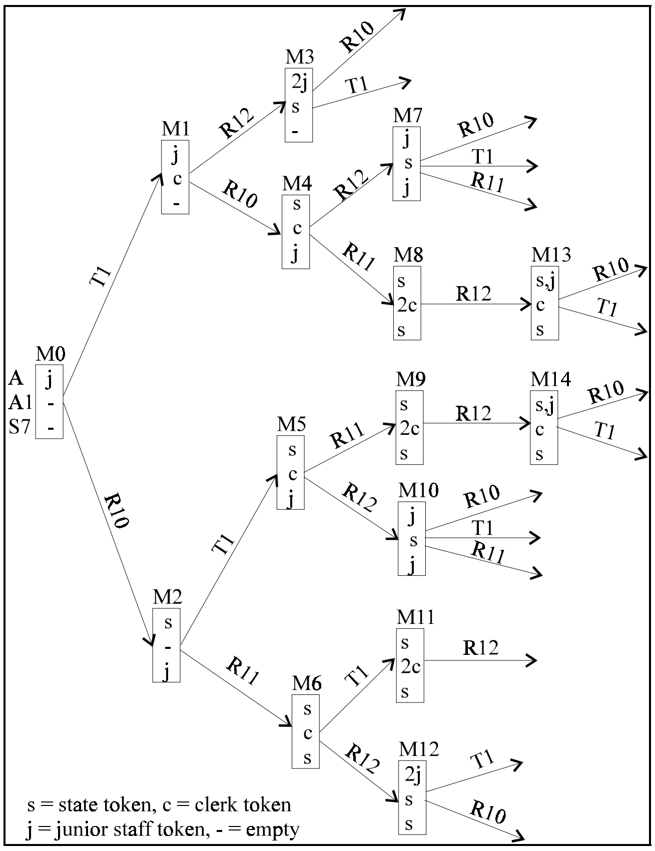

Figure 9b : Reachability graph due to the firing of R10, R11 and R12

6.2 Consistency

6.2.1 Contradiction

If two rules have duplicated antecedents but in the consequence a clause is both affirmed and denied, we refer the situation as inconsistency. The following two rules are in conflict.

Rule X : $A \wedge B \Rightarrow C$
Rule Y : $A' \wedge B' \Rightarrow \neg C$

Since both A' and B' are slots values inherited from his parent A and B, Rule X is in conflict with Rule Y. In practical expert system development, this problem is dealt with by the concepts of overriding.(ie. Rule Y overrides Rule X). This overriding behaviour is normally considered as an anomaly unless it is with the expert's true intent.

In our example, Rule 4 and Rule 5 are in conflict. In Figure 10a, if we have a Junior Office Staff token to start off in Class A with "year of service greater than five years", after firing Rule 4, then his seniority is High. A token clerk will be created in Class A1 with the same attributes, but this time after firing Rule 5, his seniority is Not High. This situation is revealed when we check the reachability graph in Figure 10b. Marking M5 is reachable from M0. In M5, we got both a Clerk token and a Junior Office Staff token in S2. When examining the state of S2 in these two tokens, we could see one is confirmed and the other is denied. This reflects that we have two conflicting rules applied to two different Object Classes.

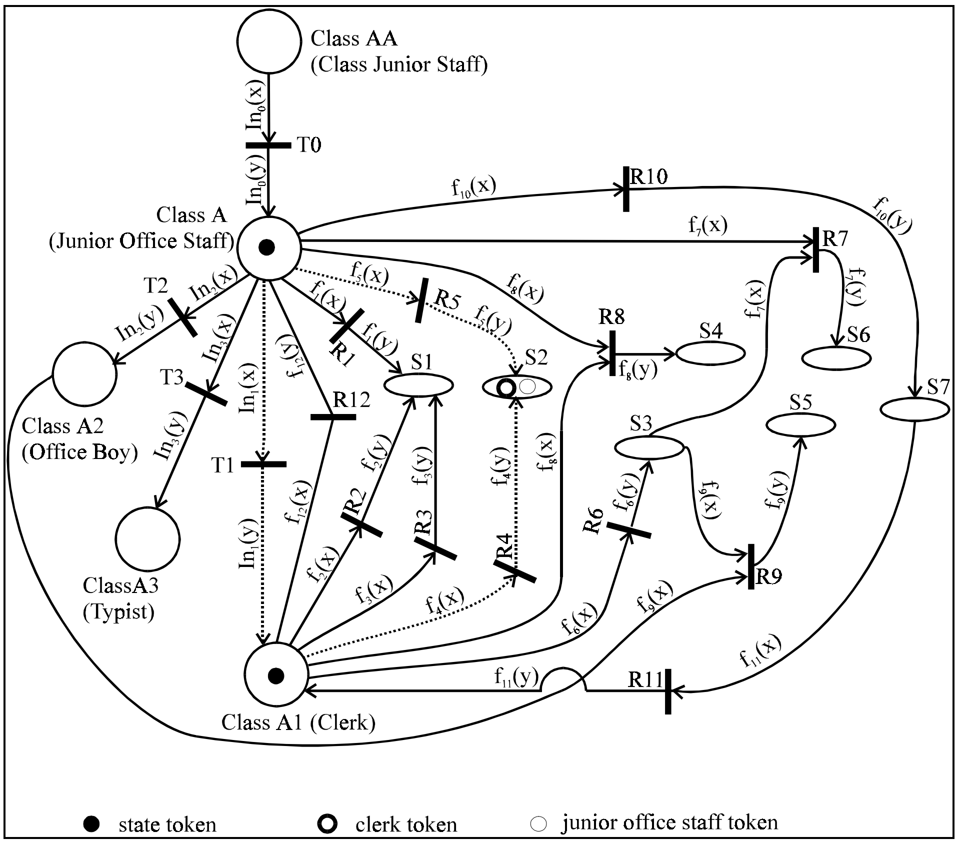

Figure 10a : CPN representation showing the events of contradiction

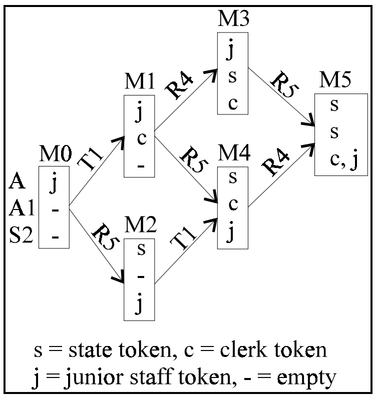
M0
M1
M2
M3
M4
M5
A
A1
S2
T1
R4
R5
s = state token, c = clerk token
j = junior staff token, - = empty

Figure 10b : Reachability graph due to the firing of R4 and R5

6.2.2 Unnecessary IF condition

If we have two rules which contain the same conclusion but with conflicting conditions, then this situation is referred to as having unnecessary IF conditions in the knowledge base. Eg. consider the following two rules

Rule X : $A \wedge B \Rightarrow C$
Rule Y : $A \wedge \neg B \Rightarrow C$

These two rules can be combined to form a simple rule :

Rule X : $A \Rightarrow C$

The second IF condition becomes unnecessary. In our example HES (Figure 11a), additional unnecessary conditions can occur when an action in one rule becomes a condition of another rule and these two rules' condition parts are in an inheritance relationship (ie. Rule 6 and Rule 7).

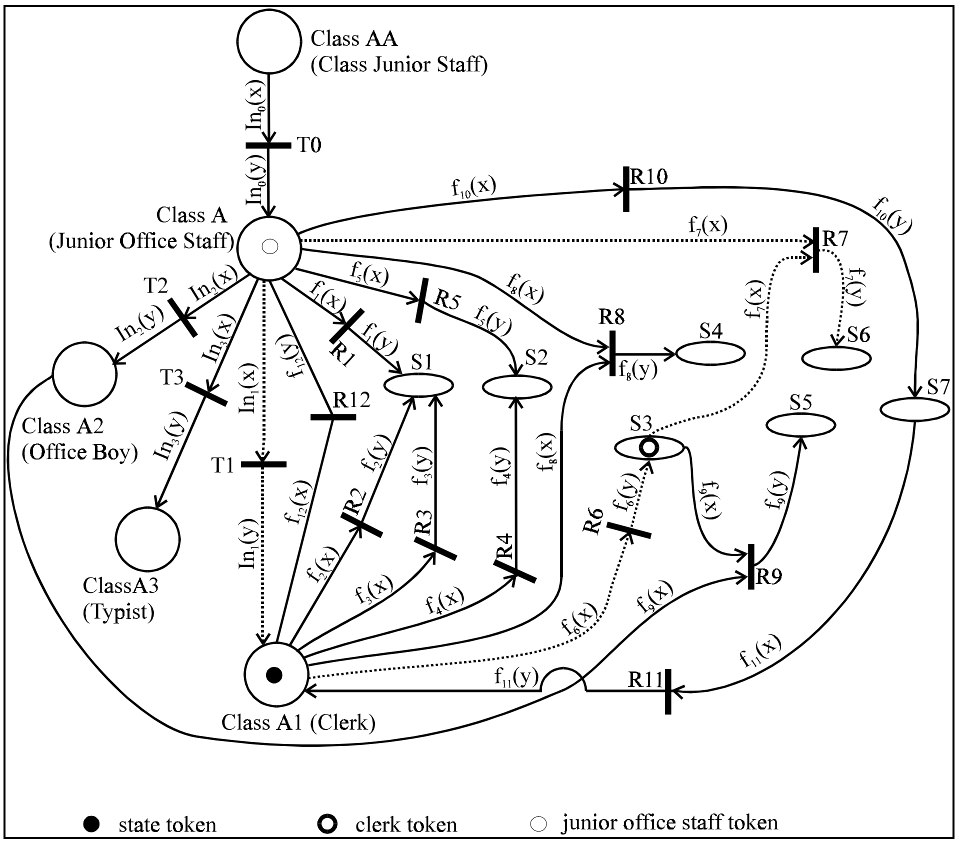

Figure 11a : CPN representation showing the events of unnecessary IF condition

Consider the following two rules

Rule X : $A \wedge B \Rightarrow C$
Rule Y : $A' \wedge C \Rightarrow D$

When Rule Y is backward chained to Rule X, (i.e. inorder that C is true, we have to check whether A is true and B is true.) Rule Y is equivalent to the testing of A', A and B:

Rule Y : $A' \wedge (A \wedge B) \Rightarrow D$

Since, A' and A are in inheritance relation, we may want to remove either the condition IF A' or IF A.

Refer to our example, when we check the reachability graph generated by the initial Junior Office Staff token in Class A, we only have three markings which S6 never gets inferred with any token. The reason is because R6 and R7 are indirectly asking the variable X to be instantiated, both to Junior Office Staff and Clerk simultaneously. Therefore, we have an unnecessary IF condition for X. (ie. IF X is a Junior Office Staff AND IF X is a Clerk.)

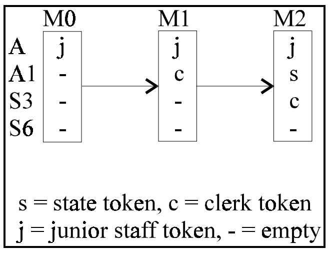
M0
M1
M2
A
A1
S3
S6
j
-
-
-
j
c
-
-
j
s
c
-
s = state token, c = clerk token
j = junior staff token, - = empty

Figure 11b : Reachability graph due to the firing of R6 & R7

6.3 Completeness

6.3.1 Unreachability, Case I

When a rule requires an object instance to be bound with two mutually exclusive classes, or two classes in an inheritance hierarchy. This rule cannot be fired. Eg.

Rule X : $A \wedge A' \Rightarrow C$
Rule Y : $A1 \wedge A2 \Rightarrow C$

In Rule X, if A is the parent and A' is the Child, it is not possible for an object instance to be both belonging to Class A and Class A'. Similarly, in Rule Y, A1 and A2 are both children of A, it is not possible for an object instance to both belonging to two different mutually exclusive classes. Referring to our example (Figure 12a), Rule 8 is found to be in this situation. Examining the reachability tree (Figure 12b), no token is ever deposited in S4 in all reachability Markings from M0.

Furthermore, if the antecedent part of a 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, then this case also leads to unreachability.

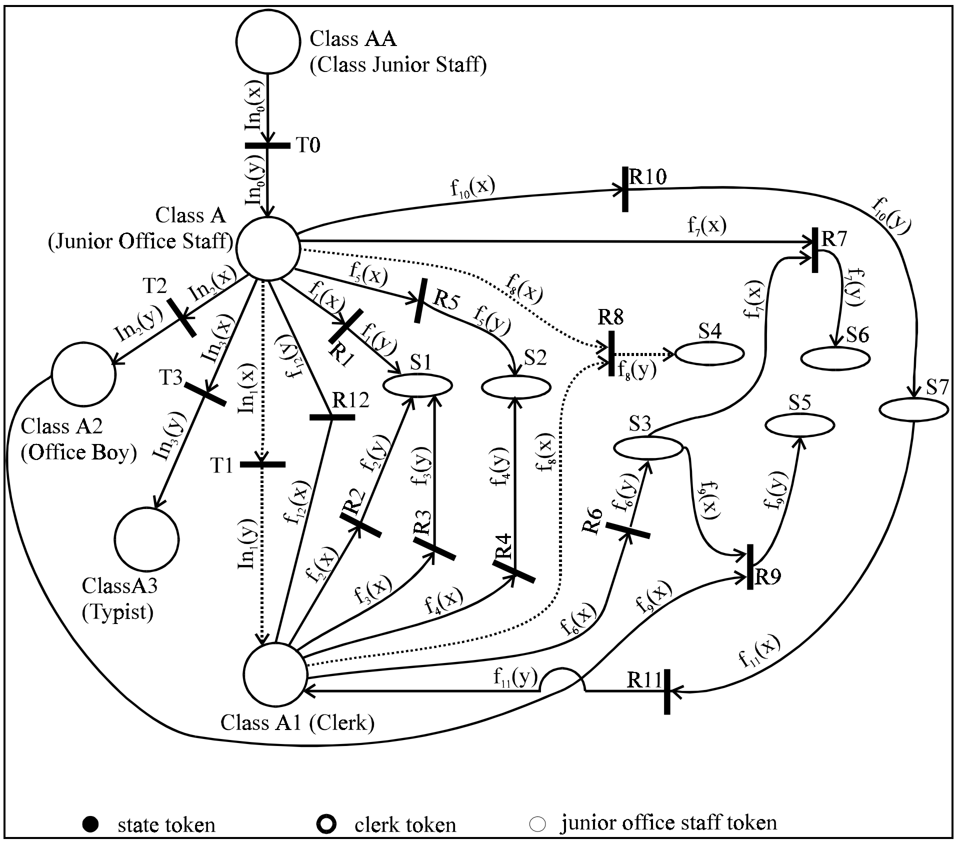

Figure 12a : CPN representation showing the events of unreachability

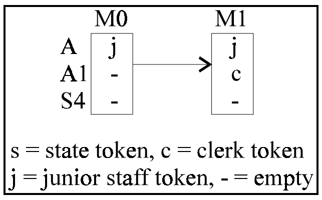
M0
M1
A
A1
S4
j
-
-
j
c
-
s = state token, c = clerk token
j = junior staff token, - = empty

Figure 12b : Reachability graph due to the firing of R8

6.3.2 Unreachability, Case II

Consider a more complicated situation with involves chain rules (Figure 13a), Rule 6's action part will forward chain to Rule 9's condition part. Now this causes an unreachable condition because Rule 6's condition part and Rule 9's condition part are having mutually exclusive class instantiation.

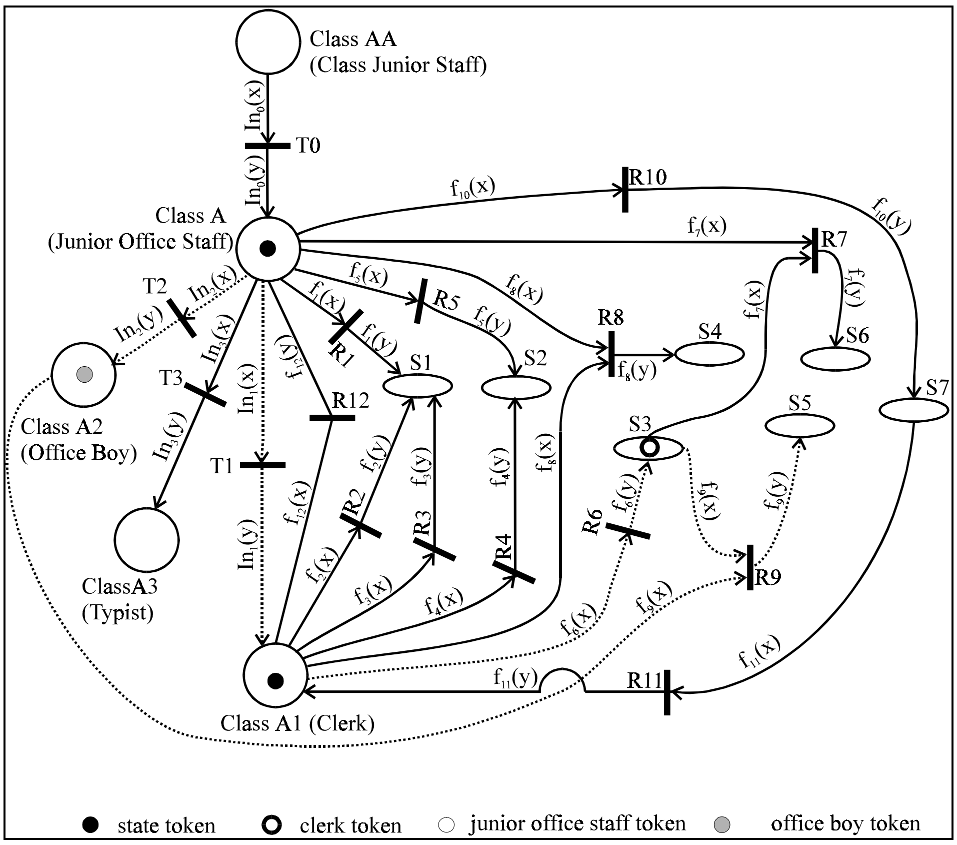

Figure 13a : CPN representation showing the events of unreachability

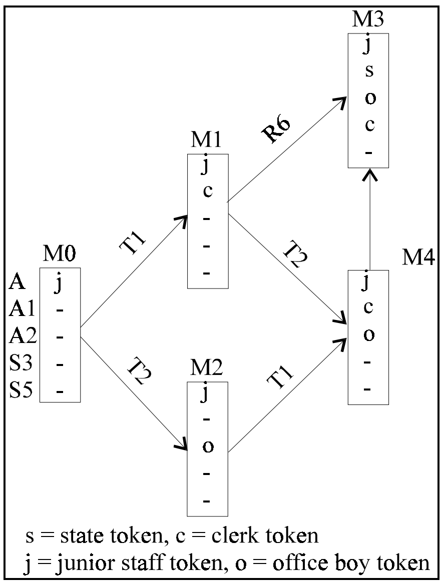
M0
A j
A1 -
A2 -
S3 -
S5 -
T1
T2
M1
j c - - -
M2
j - o - -
R6
T2
T1
M3
j s o c -
M4
j c o - -
s = state token, c = clerk token
j = junior staff token, o = office boy token

Figure 13b : Reachability graph due to the firing of R6 & R9

By examining the reachability graph in Figure 13b it shows that S5 never has any token reached from Marking M0. This means this rule is unreachable.

7 Conclusion and Discussion

In this paper, we have described an approach to model hybrid (rule- and frame-based) expert systems using Coloured Petri Nets and the concept of controlled state tokens. We have defined the additional anomalies caused by the integration of the object hierarchy, property inheritance and production rules. The detection and analysis of the anomalies of proposed model is done by constructing and examining the reachability tree spanned by the knowledge inference. An algorithm is also given to generate such a reachability set of the nets. Our approach allows for formal verification of the correctness, consistency, and completeness of the hybrid knowledge base.

Future work will include formalizing our approach and developing of algorithms and proof 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.

8 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] Chang, C.L., Combs, J.B. and Stachowitz, R.A. “A Report on the Expert Systems Validation Associate (EVA)”. Experts Systems with Applications, Vol. 1, No. 3, pp219-230, 1990.

[3] Charles, E. “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. 1991.

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

[5] Cragen, B.J. and Steudel, H.J. “A Decision Table Based Processor for Checking Completeness and Consistency in Rule-Based Expert Systems”. International Journal of Man-Machine Studies, Vol 26, pp633-648, 1987.

[6] de Kleer, J. “An Assumption-Based TMS”. Artificial Intelligence 28, pp127-162, 1986.

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

[8] 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.

[9] 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.

[10] Huen, H.S.M. A Prototype Decision Support System for Assessing The Claims for Promotion of Clerical Officers in the Hong Kong Civil Service. M.Sc. Dissertation, Department of Computing, Hong Kong Polytechnic University, 1993.

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

[12] Landauer, C. “Correctness Principles for Rule-Based Expert Systems”. Expert Systems with Applications, Vol 1, No. 3, pp291-317, 1990.

[13] 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.

[14] 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.

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

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

[17] 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.

[18] Liu, N.K. and Dillon T.S. “Formal Description and Verification of Production Systems”, International Journal of Intelligent Systems, Vol. 10, No.4, pp.399-442, 1995.

[19] 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.

[20] 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.

[21] Preece, A.D. and Shinghal, R. “DARC: A Procedure for Verifying Rule-Based Systems”. In Expert Systems World Congress Proceedings, Liebowitz, J. (ed) Vol. 2, Pergamon Press, pp971-979, 1991.

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

[23] 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.

[24] Shiu, S.C.K., Liu, J.N.K. and Yeung, D.S. “An Approach Towards the Verification of Hybrid Rule/Frame-based Expert Systems using Coloured Petri Nets”, In Proceedings of 1995 IEEE International Conference on Systems, Man and Cybernetics, pp.2257-2262, Vancouver, Canada, October 22-25,1995.

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

[26] 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.

[27] Vanthienen, J. "Knowledge Acquisition and Validation Using a Decision Table Engineering Workbench". In Expert Systems World Congress Proceedings, Liebowitz, J. (ed), Pergamon Press, Vol 3, pp1861-1868, 1991.

[28] Vranes, S. and Stanojevic, M. "Integrating Multiple Paradigms within the Blackboard Framework". IEEE Transactions on Software Engineering, Vol. 21, No. 3, pp.244-262, March 1995.

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