FORMAL DESCRIPTION TECHNIQUES FOR THE VERIFICATION OF EXPERT SYSTEMS

SHIU Chi Keung Simon

Ph.D.

THE HONG KONG POLYTECHNIC UNIVERSITY

1997

Formal Description Techniques for the Verification of Expert Systems

Submitted by

SHIU Chi Keung Simon

M.Sc., M.Sc., Teacher Certificate

A thesis submitted in total fulfillment of the requirements for the degree of Doctor of Philosophy

Department of Computing

Hong Kong Polytechnic University

1997

STATEMENT OF AUTHORSHIP

Except where reference is made in the text of the thesis, this thesis contains no material published elsewhere or extracted in whole or in part from a thesis presented by me for another degree or diploma.

No other person's work has been used without due acknowledgement in the main text of the thesis.

This thesis has not been submitted for the award of any other degree or diploma at any other tertiary institution.

SHIU Chi Keung Simon
September 1997

ACKNOWLEDGEMENTS

I would like to express my gratitude to all those who have assisted me and made this thesis possible.

I am deeply indebted to my chief supervisor, Dr. James N. K. LIU, for his invaluable guidance, helpful advice and constructive criticisms. My most sincere appreciation is extended to my co-supervisor, Professor Daniel S. YEUNG, for the countless discussions, suggestions and comments. Without his help, I would not have the courage and determination to complete this work.

In particular, I would like to thank my wife, LI Mei Yee, for her love, consideration and sacrifice she has made towards my research endeavors over the past years. Thanks also to my son, SHIU Pak Wah Ivan, and my daughter, SHIU Yee Shin Janna, for their patience and understanding.

Thanks also to the board of examiners who spent their time and effort in assessing this research work. They are Prof. Y. S. LEE from the Department of Electronic Engineering, Hong Kong Polytechnic University, Prof. K. S. Leung from the Department of Computer Science and Engineering, Chinese University of Hong Kong and Prof. S. S. Tseng from the Department of Computer and Information Science, National Chiao Tung University, Taiwan.

Finally, I would also like to express my indebtedness to the Hong Kong Polytechnic University for supporting me with the Staff Development Programme, and to Department of Computing for providing me with excellent research environment and facilities to complete my thesis.

ABSTRACT

With increasingly complex, sophisticated and changing real-world situations, it has been recognized over recent years that Expert Systems which combine one or more techniques greatly increase the problem solving capability and help overcome some of the shortcomings associated with any single technique. The verification of these Expert Systems requires methods that could tackle the multiple knowledge representation paradigms and integrated inference mechanisms used. This thesis presents a formal description technique for verifying the correctness, consistency, and completeness of Hybrid Expert Systems (HES) that emphasizes an integration of object hierarchy, property inheritance and production rules.

Four important research contributions arise from this investigation: (1) A formal approach based on State Controlled Coloured Petri Nets was developed in modelling and analyzing Hybrid Rule/Frame-based Expert Systems. (2) Errors and anomalies due to the integration of the object-hierarchy and production rules in HES are defined and explained. (3) A set of propositions is formulated to verify errors and anomalies in Rule/Frame-based HES defined in (2), and (4) Rigorous mathematical proofs of all of these propositions are developed.

The main idea of this formal technique 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 in the system are done by constructing and examining the reachability tree spanned by the knowledge inference. This provides a formal basis for automating the deduction process and a means of verifying HES.

A complexity analysis is conducted to investigate the performance of the methodology. The complexity includes the effort to transform the rules and object hierarchy into places and transactions, the calculation of the size of the Occurrence Graphs, and the time required searching such Occurrence Graphs for

anomalies. This is followed by the discussion and suggestions on the potential and direction of the developed model for future research.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION AND MOTIVATION

1.1. Introduction

Expert Systems (ES) incorporate human expertise in computer programs to allow these programs to perform tasks which normally involve human experts. Showing that an ES is 'correct' is a critical task. An incorrect system may make costly errors, or may not perform up to expectations. In either case the decisions generated by the system may be inappropriate or wrong, if used, considerable damage such as financial loss or human suffering may result.

Knowledge verification can be broadly defined (Gupta, U. G., 1993) as the process of analyzing and establishing that an ES is robust, reliable, accurate, complete, and consistent. The process of verifying a knowledge base is made up of three main activities:

- Checking if the knowledge is complete, consistent and correct.
- Determining if the reasoning mechanism accurately and consistently interprets and applies the knowledge in the knowledge base to solve system problems. This process is referred to as certification. Quite frequently, when off-the-shelf shells are used in the development process, inference engine certification is simply assumed.
- Analyzing and grading the performance of the system by comparing it with that of its human counterpart.

Traditionally, attention has been concentrated on using verification techniques to tackle rule-based systems (Gupta, U. G., 1991; Gamble R. F. et al., 1994; Liu N. K. & Dillon T., 1995; Nurrell, S. & Plant, R., 1996). However, these techniques exhibit a limited range of applicability. They could not cope with the kind of Hybrid Expert Systems (HES), e.g. Rule-based plus Frame-based, which many of the current Expert Systems are being developed (Aikins, J. S., 1993; O'Keefe, R. E. & O'Leary, D. E., 1993; Durkin, J., 1994; Vranes, S. & Stanojevic, M., 1995). 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 development software (or shells), such as ADS, ART, EXSYS EL, KAPPA-PC, KBMS, NEXPERT OBJECT, LEVEL5 OBJECT, PRO-KAPPA, REMIND, which combine some sort of Frame-based representation with a Rule-based inference engine.

The verification of these Hybrid Expert Systems requires methods that could tackle the multiple knowledge representation paradigms and integrated inference mechanisms used. This thesis presents a formal description technique based on State Controlled Coloured Petri Nets for verifying the correctness, consistency, and completeness of Hybrid Expert Systems (HES) that emphasizes an integration of object hierarchy, property inheritance and production rules.

1.2. Motivation of the Research

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, J. R. & Schultz, R. D., 1988; Duchessi, P. & O'Keefe, R. M., 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, U. G., 1991, 1993; O'Keefe, R. M. & O'Leary, D. E., 1993; Coenen, F. & Bench-Capon, T., 1993; Liu N. K. & Dillon T., 1995; Nurrell, S. & Plant, R., 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, K. et al., 1991; Preece, A. D., 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:

- 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 thesis 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.

1.3. Aim of the Research

The aim of this research is to develop a formal methodology (i.e. Mathematical Model) for specifying and verifying Hybrid Expert Systems. A broader categorization of anomalies pertaining to knowledge verification is provided. Representation schemes are examined for adequacy of representation, ability to detect anomalies and at reasonable cost. The schemes adopted in this analysis are based on the notion of State Controlled Coloured Petri Nets (SCCPNs). Predicate transitions, object-hierarchy, inheritance relations are formulated to establish correspondence between anomalies in the hybrid knowledge base and their manifestation in the transformed representation. Proofs for these transitions are derived. Algorithms are developed to detect the anomalies listed. The methodology should exhibit the following characteristics:

- provide a graphical representation of the relationships among the object hierarchy, object instances, methods, demons and the production rules.

- allow for the dynamic checking of HES which yields information on how the system achieves its goals.

- provide information about the current state of transition predicates as well as the states of the object instances.

- provide a clear semantics which allow for the formal analysis (i.e. methods which are based on mathematical representations and proof techniques) of the behaviour of the modelled HES.

- has the ability to maintain or update both the state of predicates and slot values of the object instances during transition firings.

- has a potential to tackle situations with relatively higher complexity and variant conditions like temporal space, probabilistic and fuzzy reasoning.

1.4. Scope of Research

This research will focus on the development of 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. If Domain knowledge (concepts) is related by production rules and frame hierarchy, then anomalies may arise among these knowledge (concepts) due to the existence of two mutually independent formalism of relations. The anomalies include the checking of the Correctness, Consistency and Completeness in Hybrid Expert Systems.

Correctness refers to the accuracy of the hybrid knowledge base. It includes the checking of Redundancy, Subsumption, Ambiguity and Circular rule sets that applied to the parent object class and child object classes within the HES. 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 of Contradiction, Deadend and Unnecessary IF Conditions that applied to the parent object class and child object classes within the HES. Completeness refers to the amount of knowledge built into the knowledge base. It includes the checking of Unreachability of the HES.

1.5. Contributions of the Research

(1) A formal approach based on State Controlled Coloured Petri Nets was developed in modelling and analyzing Hybrid Rule/Frame-based Expert Systems. The result was published in (Shiu, S. C. K. et al., 1997;1996b)

(2) Errors and anomalies due to the integration of the object-hierarchy and production rules in HES are defined and explained. The result was published in (Shiu, S. C. K. et al., 1995a,b; 1996a)

(3) A set of propositions is formulated to verify errors and anomalies in Rule/Frame-based HES defined in (2). Our result is to be published in a Journal paper in the *Special Issue on Intelligent Hybrid Systems* of *Expert Systems with Applications.*

(4) Rigorous mathematical proofs of all of these propositions are developed.

1.6. Outline of Thesis

Chapter One describes the traditional methods adopted in verifying Expert Systems which exhibit a limited range of applicability. They could not cope with the kind of Hybrid Expert Systems (HES), e.g. Rule-based plus Frame-based, which many of the current Expert Systems are being developed. In view of the lacking of proper understanding within this subject, i.e. Hybrid Expert System verification, the motivation of this research is to address the issues of knowledge description, formulation and verification of HES.

Chapter Two examines the issues of knowledge, Expert Systems and their verifications. Prior works in the area of knowledge verification are reviewed and their limitations assessed. This is used to guide the search for alternative approaches in modelling and analyzing of HES.

Chapter Three highlights the importance of seeking a formal description technique for modelling knowledge representations in HES. In particular, Coloured Petri Nets paradigm is adopted as the candidate methodology to support a formal description of the anomalies in terms of predicate calculus and object oriented concepts.

Chapter Four introduces a methodology for modelling HES based on State Coloured Coloured Petri Nets (SCCPNs). The general properties of a HES are described and their corresponding representations in the SCCPNs given. A Taxonomy of the anomalies in the HES is defined and explained.

Chapter Five formally presents the formulation and representation schemes for various components in a HES using SCCPNs. The scheme derived is for the purpose of knowledge verification.

Chapter Six applies the formal verification method to a practical personnel selection system and illustrates the strength and potential of the methodology. An algorithm for generating the reachability graph is provided. Through the reachability analysis, various anomalies can be revealed in this personnel selection system.

Chapter Seven formulates a set of propositions concerning verification in the transformed HES. Rigorous mathematical proofs of the correctness, consistency and completeness of HES are developed.

Chapter Eight gives a complexity analysis of the SCCPN methodology. The evaluation criteria and assessment of the utility of the approach are addressed.

Chapter Nine discusses the findings from this thesis. Possible extensions of the methodology include Hybrid Expert Systems involving uncertainty, temporal knowledge, case-based reasoning and common sense reasoning are suggested as the direction for future research work.

1.7. Publications resulted from this research

1.7.1. Refereed Journal Papers

(1). Simon C.K. Shiu, James N.K. Liu and Daniel S. Yeung, "Formal Description and Verification of Hybrid Rule/Frame-based Expert Systems," to appear in the *Special Issue on Intelligent Hybrid Systems* of *Expert Systems with Applications*.

(2). Simon C.K. Shiu, James N.K. Liu and Daniel S. Yeung, "Detection of Anomalies of Hybrid Rule/Frame-based Expert Systems Using Coloured

Petri Nets," *Australian Journal of Intelligent Information Processing Systems*, Vol. 3, No. 3, pp. 59-76, Spring, 1996.

1.7.2. Refereed Conference Papers

(3). Simon C.K. Shiu, James N.K. Liu and Daniel S. Yeung, "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, 1997.

(4). Simon C.K. Shiu, James N.K. Liu and Daniel S. Yeung, "An Approach Towards the Verification of Fuzzy Hybrid Rule/Frame Based Expert Systems". In *Proceedings of ECAI-96 Workshop in Validation, Verification and Refinement of KBS*, Budapest, 12-16 August, pp. 105-113, 1996.

(5). Simon C.K. Shiu, James N.K. Liu and Daniel S. Yeung, "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*, Vancouver, pp. 2257-2262, 1995.

(6). Simon C.K. Shiu, James N.K. Liu and Daniel S. Yeung, "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*, Melbourne, Australia, June 6-8, pp. 525-532, 1995.

CHAPTER 2. LITERATURE SURVEY AND CRITICAL EVALUATION

2.1. What is Knowledge?

According to the Webster's New World Dictionary of the American Language, Knowledge is: "a clear and certain perception of something; understanding; learning; all that has been perceived or grasped by the mind and organized information applicable to problem solving". "Knowledge encompasses the implicit and explicit restrictions placed upon objects (entities), operations, and relationships along with general and specific heuristics and inference procedures involved in the situation being modeled" (Sowa, J. F., 1984). "Knowledge is an abstract term that attempts to capture an individual's understanding of a given subject" (Durkin, J., 1994).

One of the major research areas of Artificial Intelligence has been the study of the nature of knowledge, it's formal properties and it's use in reasoning, planning and interpretation. Another aspect of the research in 'knowledge' concerns the study of particular kinds of knowledge, such as spatial, temporal, uncertain, fuzzy or causal knowledge. A major difficulty in describing knowledge is that an expert's knowledge is largely implicit. There is widespread agreement that the most difficult, time consuming, and expensive task in constructing an Expert System is extracting knowledge (e.g. in the form of production rules) from a human expert and debugging the resulting knowledge base. If noise and/or redundant data are present the problem is even more difficult. As Expert Systems are developed, modelers must provide descriptions of them for many purposes. They use some characterization in terms of properties that appear to be relevant to the knowledge base. Initial descriptions provide a central frame of reference allowing cooperation among designers of different parts of an Expert System. Descriptions also play a role in the verification process. The model must be checked for logical correctness and then implemented for compliance with a set of criteria.

However, when building an Expert System, it is impossible to capture all of the expert's knowledge. Rather, a well-focused topic from the subject area is chosen for modelling and representation. Cognitive psychologists (e.g. Newell, A., 1990) have formed a number of theories to explain how humans solve problems. These works suggest the types of knowledge humans commonly use, how they mentally organize their organization, and how they use it efficiently to solve a problem. According to (Durkin, J., 1994), knowledge can be classified as (1) Procedural knowledge: this type provides direction on how to do something. Rules, strategies, agendas and procedures, are typical type of procedural knowledge. (2) Declarative knowledge: this type describes what is known about a problem. This includes simple statements that are asserted to be either true or false, a list of statements that describes some object or concept. (3) Meta-knowledge: this type describes knowledge about knowledge. It picks other knowledge that is best suited for solving a problem. (4) Heuristic knowledge: this type describes a rule-of-thumb that guides the reasoning process. It is often called shallow knowledge because it is empirical and represents the knowledge compiled by an expert through the experience of solving past problems. If the experts are using fundamental knowledge to solve the problem, such as fundamental laws, functional relationships etc. this knowledge is referred to as deep knowledge. (5) Structural knowledge: this type describes knowledge structures. It describes an expert's overall mental model of the problem. Frames, concepts, subconcepts and objects are typical examples of this type of knowledge.

2.2. Expert Systems

The technology of Expert Systems is one of the few branches of Artificial Intelligence that has transitioned from research laboratories to the world of commercial and industrial applications. Expert Systems incorporate human expertise in a computer program to allow these programs to perform tasks normally requiring a human expert. An Expert System has been defined as "An intelligent computer program that uses knowledge and inference procedures to solve problems that are difficult enough to require significant human expertise for their solution" (Feigenbaum, E. A., 1982). "Expert Systems are systems

which are capable of offering solutions to specific problems in a given domain or which are able to give advice, both in a way and at a level comparable to that of experts in the field" (Lucas, P., 1991), and "A computer program designed to model the problem-solving ability of a human expert" (Durkin, J., 1994).

The first noteworthy Expert System was DENDRAL (Buchanan, B. & Feigenbaum, E. A., 1978). The system was designed to perform chemical analyses of the Martian soil. The success of DENDRAL marked the beginning of the so-called Expert System industry. Later successful systems include: MACSYMA, INTERNIST, CASNET, MYCIN, HARPY, HEARSAY, PUFF, PROSPECTOR and XCON. Starting from the 1980s, the interest in the field gave birth to a large number of companies that marketed Expert System development software – Expert System Shells. Today, Expert Systems 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 (Liebowitz, J., 1991; Zarri, G. P., 1991; Stachowitz, R. A. & Chang, C. L., 1991; Lee, J. K. et al., 1991).

A significant bottleneck that is frequently encountered in the use and application of Expert Systems technology is the lack of a rigorous and unified framework for testing and verifying the correctness, consistency and completeness of the Expert Systems. An incorrect ES may make costly errors, or may not perform up to expectations, may result in lawsuits, and may cause Expert Systems to be viewed as a non-viable technology for critical applications (Brown, D. E. & Pomykalski, J., 1991).

2.3. 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, M. et al., 1982), CHECK (Nguyen, T. A. et al., 1985), ESC (Cragun, B. J. & Steudel, H. J., 1987), COVADIS (Rousset, M. C., 1988), EVA (Chang, C. L. et al., 1990), KB-REDUCER (Ginsberg, A., 1988), COVER (Preece, A. D., 1989), SACCO (Laurent, J. P. & Ayel, M., 1989), NASA MMU-FDIR (Culbert, C., 1994), VALID (Meseguer, P., 1994), JIPDEC (Terano, T., 1994), SYCOJET and SACCO (Ayel, M & Vignollet, L., 1994).

This interest has driven from the need to test the large number of Expert Systems that have been developed since the mid-1980s. It also has 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, U. G., 1991, 1993; O'Keefe, R. M. & O'Leary, D. E., 1993; Coenen F. & Bench-Capon, T., 1993; Liu N. K. & Dillon T., 1995; Nurrell, S. & Plant, R., 1996). While there is controversy over how to define the terms verification and validation, there is 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 T. A. et al., 1987; Preece, A. D., 1991; O'Keefe, R. E. & O'Leary, D. E., 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, U. G., 1991; Gamble R. F. et al., 1994; Liu N. K. & Dillon T., 1995; Nurrell, S. & Plant, R., 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, R. E. & O'Leary, D. E., 1993) and (Coenen, F. & Bench-Capon, T., 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, E. H., 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, E., 1991); Decision Table Methods (Suwa, M. et al., 1982; Nguyen, T. A. et al., 1985); Incidence Matrix Method (Landauer, C., 1990; Agarwal, R. & Tanniru, M., 1991); Knowledge Base Reduction (Ginsberg, A., 1987); Generic Rule Systems (Chang, C. L. et al., 1990; Stachowitz, R. A. & Chang, C. L., 1988; Stachowitz, R. A. & Combs, J. B., 1987; Preece, A. D. & Shinghal, R., 1991a and 1991b); Bayesian Approach (O'Keefe, R. E. & O'Leary, D. E., 1993; O'Leary, D.E., 1995); Statistical Investigations (Landauer, C., 1990; O'Leary, D. E., 1988a); Rule Clustering (Jacob, R. J. K. & Forscher, J. N., 1991; Mehrotra, M., 1991); Using Test Cases, (Cuda, T. V. & Dolan, C. P., 1991) and Petri-Net Systems (Liu, N. K., 1996, 1995, 1993, 1991; Wu, C. H. & Lee S. J., 1995; Scarpelli, H & Gomide, F., 1994a, 1994b; Yao, Y., 1994; Zhang, D. & Nguyen, D., 1994; Nazareth, D. L., 1993; Agarwal, R. & Tanniru, M., 1992; Meseguer, P., 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 thesis, 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.

2.4 Major Approaches for Expert Systems Verification

2.4.1. Normal Form Approach

(Charles, E., 1991) considered that knowledge based systems can be checked at the clausal level. The rules in an ES are translated into clausal form using logical equivalence. The most common types of clausal forms are Disjunctive Normal Form (DNF) and Conjunctive Normal Form (CNF). Checking for anomalies requires comparing the individual clauses. E.g.

Rule 1: IF B OR C THEN A

Rule 2: IF (B AND D) OR E THEN A

These then translate into the following clauses:

Clause 1: $B \Rightarrow A$

Clause 2: $C \Rightarrow A$

Clause 3: $B \wedge D \Rightarrow A$

Clause: 4: $E \Rightarrow A$

Here clause 1 subsumes clause 3. Meta-rules can be used to identity subsumed clauses, e.g. A clause is subsumed by another if all literals in the second clause are also contained in the first, i.e. a clause $(A \wedge B \wedge C)$ will be subsumed by a clause $(A \wedge B)$. If all the clauses derived from a rule are eliminated then the rule is declared to be redundant. If only some clauses are eliminated then the rule is declared to contain redundant premises or conclusions. Although this Normal Form approach is quite straightforward, the decision of how to resolve anomalies detected is based on the view as to how the rule-base should be structured together with expert view of the domain. Another problem of this approach is its inability to deal with complicated Hybrid Expert Systems.

2.4.2. Decision Table Methods

A decision table is a tabular representation of a procedural decision situation, where the state of a number of conditions determines the execution of a set of actions (Coenen, F. & Bench-Capon, T., 1993). Conditions are given along the X-axis and actions along the Y-axis. The aim is to demonstrate the results of an exhaustive set of mutually exclusive combinations of conditions. More succinctly, decision tables can be said to be a method of organizing and documenting logic in a manner that allows easy inspection and analysis. This approach, widely used in conventional software specification to determine the effect of conditional statements, has been used to facilitate the testing of a set of rules for conditions of ambiguity redundancy and completeness (Cragen, B. J. & Steudel, H. J., 1987; Vanthienen, J., 1991). A number of variations on the decision table approach have also been developed, examples include Suwa's rule checking program (Suwa, M. et al., 1982) and Nguyen's CHECK (Nguyen, T. A. et al., 1985).

The main idea of decision table techniques is as follows:

- Separate rules into sub-tables so they have logical isolation from other rules i.e. the rules in the set have at least one condition (attribute) in common, and no other rules uses that attribute.
- Further separate the sub-table so that no rule in a sub-table allocates the value for a condition in another rule in the same table.
- A master table is created to display all possible combinations for condition parameters and resulting action parameters.
- The master table is used to check for conflicts, redundancies, subsumptions and missing rules. (e.g. A missing rule is identified if there is a possible combination of values of attributes appear in the antecedent set but no corresponding output action.)

The problem of decision table techniques is that in any realistically sized application the table will grow to unmanageable proportions unless some form of partitioning is implemented whereby only a finite number of condition-action

combinations need be considered. A further problem of decision table techniques is that only static checks are involved, i.e. no consideration is given to dynamic checking.

2.4.3. Incidence Matrix Method

The incidence matrix method of Expert System verification and validation involves the construction of matrices to determine the number of rules containing a certain variable or combination of variables or the number of variables common to a set of rules. A number of knowledge base system verification and validation systems exist that are based on the development of incidence matrices (Landauer, C., 1990; Agarwal, R. & Tanniru, M., 1991). Structurally an incidence matrix is very similar to a transposed decision table (i.e. Conditions are given along the Y-axis and Actions along the X-axis).

The main idea of incidence matrix technique is as follows:

- Convert the rules into a incidence matrix by
    1. Assign negative numbers to possible values for attributes in the antecedent of the rules.
    2. Assign positive numbers to possible values for attributes in the consequent of the rules.
    3. Assign zero to attributes not appear in either antecedent or consequent of the rules.

       E.g. The incidence matrix of the following rules is:

       Rule 1: IF A=1 AND B=1 THEN C=1

       Rule 2: IF A=2 THEN C=2

	A=1,	A=2,	B=1,	C=1,	C=2
Rule 1	-1	0	-1	1	0
Rule 2	0	-1	0	0	1

$$= \begin{bmatrix} -1 & 0 & -1 & 1 & 0 \\ 0 & -1 & 0 & 0 & 1 \end{bmatrix}$$

- Multiply the incidence matrix by the transpose of itself. The resulting matrix shows the common elements contained in the rules in the knowledge base.
- If a rule (e.g. Rule X) does not share an element with one other rule in the knowledge base (KB), we can consider it to be redundant. (i.e. If we multiply the incidence matrix of the knowledge base (KB) with the transpose of the incidence matrix for the rule (Rule X), the resulting matrix contains only zeroes).
- Similar steps as above can also check other anomalies such as subsumption and incompleteness.

The problem of incidence matrix techniques is very similar to decision table techniques, i.e. any realistically sized application the matrix will grow to unmanageable proportions unless some form of partitioning is implemented whereby only a finite number of condition-action combinations need be considered. A further problem of incidence matrix techniques is that only static checks are involved (i.e. no consideration is given to dynamic checking).

2.4.4. Knowledge Base Reduction Systems

The concept of knowledge base reduction, as advocated by (Ginsberg, A., 1987) is based upon ideas concerned with solving problems associated with truth maintenance as advanced by de Kleer amongst others, particularly those concepts underlying de Kleer's Assumption-Base Truth Maintenance System (ATMS) (de Kleer, J., 1986). Ginsbery defines KB Reduction as:

> "A technique whereby for every assertion H that a KB can make one has calculated all possible logically independent and minimal sets of inputs under which the KB will be led to assert H."

The technique requires that the rules in a KB form a hierarchical network that can be partitioned into levels starting at the leaf nodes. Labels are then computed for every hypothesis and default hypothesis in the KB. A hypothesis is any literal in the consequent of a rule and default hypothesis is any literal in the antecedent of a rule that negates a hypothesis. A label is thus the set of all minimal logically consistent inputs to establish a hypothesis. This set of inputs is referred to as an environment and the individual inputs as findings. The latter are defined as antecedent literals that are not negated in the consequent of any rule. By comparing labels we can then identify anomalies such as inconsistency, redundancy, auxiliary rules and subsumptions. For example, consider the following four rules:

Rule 1: IF P OR Q THEN R
Rule 2: IF A THEN B
Rule 3: IF B AND R THEN S
Rule 4: IF NOT S AND C THEN T

Findings: P, Q, A, C
Hypotheses: R, B, S, T
Default Hypothesis: $\neg S$

The labels are then:
Label R: $P \vee Q$
Label B: A
Label S: $(A \wedge P) \vee (A \wedge Q)$
Label T: $\neg A \wedge C \vee (\neg P \wedge \neg Q \wedge C)$
Label $\neg S$: $\neg A \vee (\neg P \wedge \neg Q \wedge C)$

If the rule label R consists solely of inconsistent environments (e.g. $P \wedge \neg P$) the rule R can be eliminated because it can never be fired. If the rule label is implied by the current partial label of H, (e.g. Every environment of H is a super-set of some environment in the rule label of R) this may suggest the existence of a subsumed rule.

The basic advantage of the KB reduction is its ability to treat multiple, mutually contradictory, states at once. In addition, it has an advantage when dealing with problems that require many solutions, since it avoids all dependency directed backtracking and context switching. The advantage of the KB reduction is less clear when there is only one solution. It all depends on how much dependency-directed backtracking is required to find the single solution. The more there are, the better the KB reduction is likely to be. For some problems with many solutions, the basic KB reduction may contribute to inefficiency because all solutions will be explored when only one or a few may be necessary.

2.4.5. Generic Rule Systems

Generic rule systems are designed to allow knowledge based systems, using any representation, to be verified and validated by first translating the rules or frames into a generic representation. Two outstanding examples are the EVA (Chang, C. L. et al., 1990; Stachowitz, R. A. & Chang, C. L., 1988; Stachowitz, R. A. & Combs, J. B., 1987) and COVER (Preece, A. D. & Shinghal, R., 1991a and 1991b). EVA is written in PROLOG and consists of a wide range of validation tools that enable the user to check the redundancy, consistency, completeness and correctness of a KBS. EVA can be viewed as a metashell consisting of a unifying architecture that uses a single inference strategy, a single meta-KB and a common language for specifying requirements, constraints and models for domain knowledge. This makes EVA independent of any specific shell. COVER is implemented in PROLOG and C, and runs on SUN workstations. COVER checks for a number of types of anomaly such as deficiency, ambivalence, redundancy and circularity.

The major criticism of using Generic Rule Systems is that specific languages used in particular Expert System shells must be translated into this generic form, which is not a trivial task.

2.4.6. Bayesian Approach

In systems which attempt to measure uncertainly or strength of association, using certainty factors, Bayesian probabilities or any other method, it is also important to verify that the weights are consistent, complete, correct and not redundant. (O'Keefe, R. E. & O'Leary, D. E., 1993; O'Leary, D. E. 1995). This can be done by ensuring that each rule that is supposed to have a weight does have one and that the weights are developed in concert with the theory on which they are based. For example, given the following rule:

Rule 1: IF E THEN H (to degree S, N)

where S and N are numeric values that represent the strength of association between E and H. S is a sufficiency factor, since a large value of S means that a high probability for E is sufficient to produce a high probability for H, and N is a necessity factor, since a small value of N indicates that a high probability of E is necessary to produce a high probability of H. S and N can be specified directly, or they can be developed by establishing the likelihood ratios. The relationships between these ratios could take any of a number of functional forms, including linear, quadratic, etc. Anomalies exist if these are violated.

Finding anomalies in the weights in an ES is a process that has received limited attention, probably due to the limited number of implemented ES that make extensive use of uncertainty measures.

2.4.7. Statistical Investigations

Statistical methods can be a useful static or dynamic verification test. (Landauer, C., 1990; O'Leary, D. E., 1988a) suggest that various aspects of rules, such as attributes and conclusions, be analyzed statistically as part of the verification process. This can be done statically or dynamically. The frequency that rules are fired or paths are traversed can be statistically analysis to reveal some anomalies. For example, a priori, it may be expected that a particular rule or sequence of rules should fire frequently. If analysis of actual or simulated use of the system provides data that indicates that this is not the case, then it would be appropriate

to examine those rules in more detail. In applying the statistical technique, the biggest problem is the identification of criteria for analysis. This may require expert's input, again, it is time consuming and error prone.

2.4.8. Rule Clustering

Grouping rules can be performed by: (1) measuring the distance between two rules based on their relatedness, and (2) clustering rules with a minimum distance. Rule grouping has been advocated for improving the modularity of rule bases, consequently enhancing their maintainability (Jacob, R. J. K. & Forscher, J. N., 1991; Mehrotra, M., 1991). In terms of knowledge base verification, by decomposing the rule base into a number of meaningful units, the pair-wise comparisons among rules within the groups can be minimized. The main idea is as follows:

- Calculate the distance metric between rules using the formula

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

  where $D(r_i,r_j)$ is the distance metric
- Rules are clustered with a minimum distance
- Construct a rule connection graph in each cluster of rule sets
- Represent the rule sets using adjacent matrix
- Apply pair-wise comparisons and detect for anomalies

The advantage of this method is the reduction of the total number of pair-wise comparisons with the assumption that errors will occur between pairs or limited sets of rules. Consequently, this approach fails to recognize that some knowledge systems are not simple classification systems, but could involve a network of inferences, and errors in chained inference are a definite possibility, even if pairwise comparisons indicate no errors.

2.4.9. Using Test Cases

Using test cases may be considered as an informal method, however, some authors (Cuda, T. V. & Dolan, C. D., 1991) claim that many of the properties checked by formal means may be better dealt with by relatively informal techniques based on a particular method of constructing a knowledge base. A test case consists of a set of inputs for a particular problem and the correct outputs. The knowledge base is given the inputs of a test case, and the outputs it generated are checked against the known correct outputs. A test case descriptor allows a possible range of values to be entered for each input parameter and then forms cases by taking the cross product.

An unfulfillable goal is detected if, as a result of running the test cases a goal does not have a value. Unreachable attribute values and if-conditions are detected in a similar way. Illegal input sets are identified by presenting the expert with input sets produced by the test case descriptors. Illegal output sets are checked for by running the widest possible range of test cases and checking the system's output against them. Although using test cases is straightforward and easy to use, it requires expert's evaluation and interpretation of the tests' results. Again, it is time consuming and error prone.

2.4.10. Petri-Net Systems

Suggestions for the use of Petri Nets (Liu, N. K., 1996, 1995, 1993, 1991; Wu, C. H. & Lee S. J., 1995; Scarpelli, H & Gomide, F., 1994a, 1994b; Yao, Y., 1994; Zhang, D. & Nguyen, D., 1994; Nazareth, D. L., 1993; Agarwal, R. & Tanniru, M., 1992; Meseguer, P., 1990) to model the interaction and temporal relationships between individual events represented in production-based Expert Systems appear promising. The model's behaviour can be expressed in Algebraic form, thus supplying the basis for automating algorithms, capable of proving properties of the modelled system.

Petri-net models are abstract, formal representations of information flow. They describe the input/output relationship between objects using a graphical representation. Using Petri nets for verification purposes, each rule is translated into a transition by allocating a place to each condition and each action in the

rule. The detection and analysis of the anomalies in the system are done by constructing and examining the reachability tree spanned by the knowledge inference. A more detailed description and evaluation of Petri Nets systems as the appropriate methodology for this research project will be discussed in the next Chapter.

2.5. Summary

A number of issues about the description and verification of knowledge being applied in Expert Systems has been discussed. To obtain a conceptual basis for the theme of verification in this thesis, we have defined the process of verification as the checking of the appropriateness of a model. This involves the checking of correctness, consistency and completeness in Expert Systems. A number of major approaches to the verification of Expert Systems have been reviewed. These include the use of Normal Form Approach, Decision Table Methods, Incidence Matrix Method, Knowledge Base Reduction, Generic Rule Systems, Bayesian Approach, Statistical Investigations, Rule Clustering, Using Test Cases, and Petri-Net Systems. However, these techniques exhibit a limited range of applicability. They could not cope with the kind of Hybrid Expert Systems (HES), e.g. Rule-based plus Frame-based, which many of the current Expert Systems are being developed. The verification of these Hybrid Expert Systems requires methods that could tackle the multiple knowledge representation paradigms and integrated inference mechanisms used.

CHAPTER 3. CHOICE OF METHODOLOGY

3.1. Logic-based Techniques

This set of techniques includes the use of Normal Form Approach (Charles, E., 1991); Decision Table Methods (Suwa, M. et al., 1982; Nguyen, T. A. et al., 1985); Incidence Matrix Method (Landauer, C., 1990; Agarwal, R. & Tanniru, M., 1991); Knowledge Base Reduction (Ginsberg, A., 1987); Generic Rule Systems (Chang, C. L. et al., 1990; Stachowitz, R. A. & Chang, C. L., 1988; Stachowitz, R. A. & Combs, J. B., 1987; Preece, A. D. & Shinghal, R., 1991a and 1991b). The major problem of logic-base technique is that it does not provide explicit and complete interrelationship of knowledge structure. Therefore, it cannot reflect a network of possible inference in a knowledge base. Consequently, anomalies due to any semantic gap will unlikely be detected and verified. Furthermore, logic-based techniques is incapable of supporting the investigation of any concurrent and dynamic behaviour exhibiting in the knowledge systems. Lastly, logic-based techniques cannot support the descriptions of complex data types such as object-oriented concepts.

3.2. Statistics-based Techniques

This set of techniques includes the use of the Bayesian Approach (O'Keefe, R. E. & O'Leary, D. E., 1993; O'Leary, D. E., 1995); Statistical Investigations (Landauer, C., 1990; O'Leary, D. E., 1988a); Rule Clustering (Jacob, R. J. K. & Forscher, J. N., 1991; Mehrotra, M., 1991). In systems that attempt to measure uncertainty or strength of association, statistical methods can be a useful static or dynamic verification test. Nevertheless, these techniques employ meta-knowledge from the domain to examine the statistical results from the tests. Thus, it is not easy to identify a generic approach in representing anomalies based on statistics. Furthermore, the assumptions made when carrying out the statistics has to be examined together with the results. Finally, this kind of techniques is only good at

proving 'the system is good as long as not proven bad'. The correctness, consistency and completeness of the system cannot be formally established.

3.3. Test Cases-based Techniques

This set of techniques mainly use Test Cases (Cuda, T. V. & Dolan, C. P., 1991). The problems of using test cases includes: (1) Difficulty in establishing criteria for testing; (2) Difficulty in comparing results generated from the test cases; (3) How to maintain Objectivity; (4) How to determine the reliability of the Expert Systems if only test cases are used, and (5) The availability of the test cases. Another major problem with using test cases is an assumption that the expert against which the system is being compared is always correct, i.e. if the system differs from the expert then it is 'wrong'. Using synthetic cases is dangerous, and demands considerable objectivity on behalf of the developers. Finally, there is always a temptation to make the test cases reflect the known strengths of the system.

3.4. Petri Nets-based Techniques

Petri Nets based techniques (Liu, N. K., 1996, 1995, 1993, 1991; Wu, C. H. & Lee S. J., 1995; Jensen K., 1995, 1996; Scarpelli, H & Gomide, F., 1994a, 1994b; Yao, Y., 1994; Zhang, D. & Nguyen, D., 1994; Nazareth, D. L., 1993; Agarwal, R. & Tanniru, M., 1992; Meseguer, P., 1990) is now in widespread use for many different practical purposes. The main reason for the great success of these kinds of net models is the fact that they have a graphical representation and a well-defined semantics allowing formal analysis. The simplest Petri nets are those without colours and called Place/Transition Nets (PTN). In PTNs there is only one kind of token and this means that the state of a place is described by an integer or by a Boolean value (e.g. 1 or 0). In high level nets, such as Coloured Petri Nets (CPNs), each token carries complex information or data which may be used to describe the entire state of a process. A Petri net model is a description of the state and action of a system – it gives an explicit description of both the states and actions of the system. This allows

the user to determine freely whether, at a given moment of time, he wants to concentrate on states or on actions. In a typical Petri Net diagram (Figure 3.1.), states of the system are indicated by means of cycles (or ellipses) which are called places. Each place may contain a dynamically varying number of small black dots, which are called tokens. An arbitrary distribution of tokens on the places is called a marking. An initial distribution of tokens on the places is called the initial marking and it is usually denoted by M_0. The actions of the system is indicated by means of rectangles, which are called transitions. The places and transitions of a Petri net are collectively referred to as the nodes. The Petri net also contains a set of directed arrows, which are called arcs. Each arc connects a place with a transition or a transition with a place – but never two nodes of the same kind. Each arc may have an expression attached to it (e.g. A positive integer), this expression is called an arc expression. The above gives the syntax of a Petri net.

With reference to the semantics of a Petri Net, each transition represents a potential move. A move is possible if and only if each input place of the transition contains at least the number of tokens prescribed by the arc expression of the corresponding input arc. If this happens, the transition is enabled. When a transition is enabled the corresponding move may take place, this means the transition occurs. The effect of an occurrence is that tokens are removed from the input places and added to the output places. The number of removed/added tokens is specified by the arc expression of the corresponding input/output arc. Tokens are removed from the input places, and completely new tokens are added to the output places. This means that there is no relationship between the token removed and the token added. The execution of a transition T transforms marking M_0 to the marking M_1, therefore, M_1 is reachable from M_0 by T. If two or more transitions are concurrently enabled in a marking M, this means the enabled transitions may occur at the same time (or occur in parallel). A transition may even occur concurrently to itself, if there are sufficient tokens deposited in its input places.

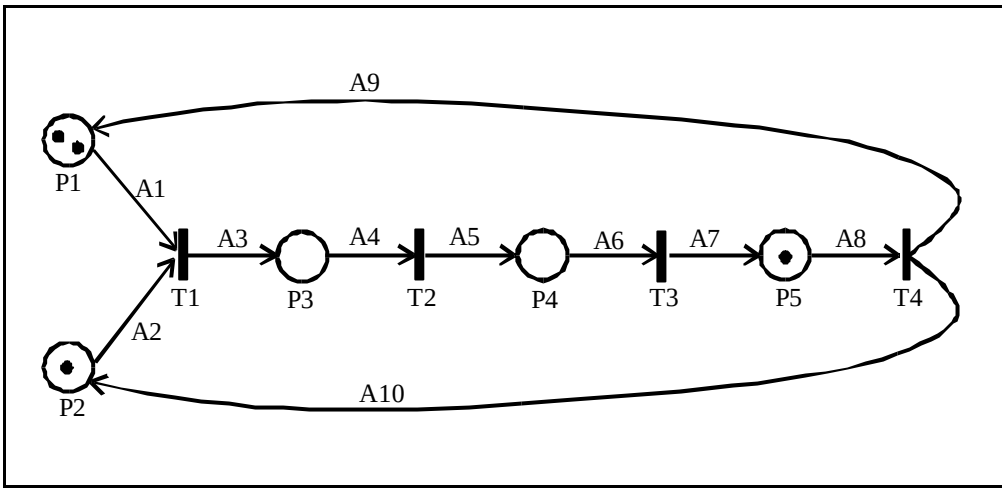


Figure 3.1. A Petri Net

In Figure 3.1., there are five places, P1 to P5; four transitions, T1 to T4 and ten arcs, A1 to A10. There are two tokens in place P1, one token in place P2 and another token in P5.

3.4.1. Coloured Petri Nets

In order to describe complex systems in a manageable way, the development of high level Petri Nets constitutes a very significant improvement in this respect. Coloured Petri Nets belong to the class of high level nets. The main advantages of Coloured Petri Nets (CPNs) over PTNs is the introduction of the data type concept. In CPN, each token is attached a data value, called the token colour. The data value may be of arbitrarily complex type. (e.g. A record where the first field is a real number, the second is a text string, while the third is a list of integer pairs). For a given place all tokens must have token colours that belong to a specified type. This type is called the colour set of the place. Colour sets determine the possible values of tokens analogously to the way in which types determine the possible values of variables and expressions in programming languages.

Attaching a colour to each token and a colour set to each place allows the use of fewer places than would be needed in a PTN. Intuitively, the introduction of colours has allowed the folding of places into a single place without losing the ability to

distinguish between various states and actions of the system. Similarly, to get transitions which can represent many different actions, the arc expressions surrounding a given transition can contain a number of variables. This variable can be bound to different values and this means that the expressions evaluate to different values. Therefore, in order for the transition to occur, the variables in the arc expressions surrounding the given transition have to be bound to colours (data) of the corresponding type (data type). When a transition is enabled for a certain binding, it may occur, and it then removes tokens from its input places and adds tokens to its output places.

The introduction of colours into PTNs have the following advantages:

- Description and analysis of systems become more compact and manageable.
- It is possible to describe data manipulations in a direct way.
- It becomes easier to see the similarities and differences between similar systems.
- It is possible to create hierarchical descriptions.

3.4.2. Choice of Coloured Petri Nets to model Hybrid Expert Systems

3.4.2.1. Requirements of the modelling language

In the design and analysis of Expert Systems, questions of correct behaviour is very important since they may be used to control traffic systems, telecommunication systems, medical diagnostic systems, etc. Their incorrect behaviour could lead to disasters. Moreover, dynamic analysis of Expert Systems is more difficult to understand due to their combinatorial complexity. Therefore, a proper formal methodological framework for the verification of knowledge bases is needed.

Formal approaches to software specification and development have been a topic of active research for a long time. Formal methods are introduced (Graigen, D. et al, 1993) as "mathematically based techniques, often supported by reasoning tools, that can offer a rigorous and effective way to model, design and analyze computer systems." At the specification level, a formal method provides a notation for software specification and development with some mathematical meaning which each specification is associated a mathematical entity. Moreover, there is a formal deduction system which makes it possible to perform some symbolic computations or proofs. This formal system is consistent with the mathematical meaning. One of the main interests of formal techniques is the possibility to perform proofs. Such proofs have different aims: (1) they can be used to verify a specification, (i.e. by verifying that some properties are consequences of the specification, or by refuting some other properties which correspond to undesirable situations; (2) They can be used to verify that a design step is correct, (i.e. that a detailed specification is compatible with a less detailed one; (3) They can be used to check that a system satisfies a specification, (i.e. by proofing the properties of the system).

A formal description technique (Broy, M., 1991) comprises of the following two components:

- Syntax: the forms of descriptions are precisely defined, this can be done by graphical forms as well as by textual forms or mixtures of both of them.

- Semantics: the meaning of the syntactic forms has to be uniquely defined, this can be done by mapping the syntactic forms onto appropriately chosen semantic models as well as by logical calculi.

Without a mathematically properly defined semantics, a description technique cannot be called formal but at most semiformal. Semantic models are helpful in the understanding of the concepts of a formal description technique. Logical calculi are

of methodological importance when developing, transforming, and verifying systems descriptions. In general, an available formal framework serves two important purposes. First of all, it gives a proper foundation such that it is clear what is meant that a system is correct or that it can be verified. Second, support tools that should give substantial support aid have to be based on formal methods. This is why formal description methods get more and more into practical use, at least, if systems with high reliability are required.

As the major aim of this Ph.D. research is to develop a formal methodology for specifying and verifying Hybrid Expert Systems, the most important requirement is whether the modelling language used has a sound, solid and well defined semantics for formal analysis. In addition, the methodology chosen should exhibit the following potentials:

- provide a graphical representation of the relationships among the object hierarchy, object instances, methods, demons and the production rules in the Hybrid Expert Systems.

- allow for the dynamic checking of HES which yields information on how the system achieves it goals.

- provide information about the current state of transition predicates as well as the states of the object instances.

- provide a clear semantics which allow for the formal analysis of the behaviour of the modelled HES.

- has the ability to maintain or update both the state of predicates and slot values of the object instances during transition firings.

- has a potential to tackle situations with relatively higher complexity and variant conditions like temporal space, probabilistic and fuzzy reasoning.

3.4.2.2. Reasons of choosing Coloured Petri Nets as the modelling language

First of all, a CPN model is a description of the modelled system, and it can be used as a specification of a system which is to be built, or a representation of a system which we want to understand and communicate with others. Secondly, the behaviour of a CPN model can be analyzed, either by means of simulation or by means of formal analysis method. Detail reasons for using it for this research project are as follows:

3.4.2.2.1. Graphical Representation

It is extremely easy to understand and grasp the meaning of the modelled systems by CPN because of its graphical representation. This is due to the fact that CPN diagrams resemble many of the informal drawings which designers and engineers make while they construct and analyze a system. The notion of states, actions and flow are particularly appealing, when they are used to model the states of the predicates of the rules, the inference mechanisms in the Expert Systems. Many concepts in Expert Systems can be represented by places, transitions, and arcs of CPN directly.

3.4.2.2.2. Well Defined Semantics

It is the presence of the semantics which makes it possible to implement simulators for CPNs, and it is also the semantics which form the foundation for the formal analysis methods that this research project seeks to develop.

3.4.2.2.3. Concurrent Systems

Transitions in CPN can be concurrently enabled and occurred, they can be used to model systems which required descriptions of concurrently behaviours. This means that the notions of conflict, concurrency and causal dependency can be defined in a very natural and straightforward way. In Hybrid Expert Systems, the dynamic behaviour (e.g. Inference strategies, inheritance among the object classes in the hierarchy) can be modelled explicitly using these concurrently enabled transitions.

3.4.2.2.4. Few, but Powerful Primitives

The definition of CPN is rather short and it builds upon standard concepts which are based on mathematics and programming languages. This means that it is relatively easy to learn to use CPN. In addition, the small number of primitives also means that it is much easier to develop strong analysis methods.

3.4.2.2.5. Explicit Description of both States and Actions

Since CPN is a system description language which explicitly describe both states and actions, it is easy for the user to change the point of focus from state to actions, or from actions to states. In the case of HES modelling and analysis, at some instance, it may be convenient to concentrate on the states of the predicates, and the states of the object instances while at other instances it may be more convenient to concentrate on the inferences or inheritance of the object properties.

3.4.2.2.6. Hierarchical Descriptions

This means that large CPN can be constructed from relating smaller CPNs in a well defined manner. Therefore, the modelling of very large systems can be carried out in a manageable and modular way.

3.4.2.2.7. Data Manipulation

This means that from a CPN, it can be seen what the environment, enabling conditions and effects of an action are. The data manipulation is carried out by the net expressions, which may be built from a number of variables, constants, operations and functions. The manipulation is similar to applying the operation and functions to the binded variables. The operations and functions take a number of arguments and return a result.

3.4.2.2.8. Formal Analysis Techniques

Formal analysis techniques are available for CPNs, such as the construction and analysis of occurrence graphs (representing all reachable markings); calculation and interpretation of system invariants (place and transition invariants); reductions (which shrink the net without changing a certain selected set of properties) and checking of structural properties (which guarantee certain behavioural properties).

3.5. Summary

As the major aim of this Ph.D. research is to develop a formal methodology for specifying and verifying Hybrid Expert Systems, the most important requirement is whether the modelling language used has a sound, solid and well defined semantics for formal analysis. In addition, the method chosen should be able to model both the Frame-based and Rule-based knowledge representation characteristics. Such a technique is chosen as a possible candidate among logic-based, statistics-based, test-cases based and Petri nets-based methods. The analysis of these choices suggests the use of the Petri Nets paradigm as the candidate methodology for modelling knowledge representations in Hybrid Expert Systems. The distinguished network characteristics and the concept of coloured tokens can be used to establish formal description and verification of Hybrid Expert Systems. This will require semantic extensions of the nets to provide sufficient descriptive and expressive power for the purpose of verification of hybrid knowledge bases.

CHAPTER 4. MODELLING AND VERIFICATION PROBLEMS IN RULE/FRAME-BASED HYBRID EXPERT SYSTEMS (HES)

4.1. 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 (French, S. W. & Hamilton, D., 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 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 (Shiu, S. C. K. et al., 1995a, 1995b), 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 the 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.

Usually, Object class has a set of attributes, demons and methods. Each attribute is 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.

A Demon is 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 linking up different Object Classes.

- Forward Chain with Inheritance: Data directed search with inheritance as one of the means to establish the rule chains linking up different Object Classes.

Other important inference strategies include: Pattern Matching, Unification, Resolution and Heuristic Search.

4.2. Modelling Hybrid Expert System using State Controlled Coloured Petri Nets (SCCPNs)

4.2.1. 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. (i.e. "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 identifies which object instance had fired from which transition. (i.e. var a : OO, var a1 : OO and var s : text string)

4.2.2. Rules with Inheritance

In SCCPN, the transition operations are represented by the arc expression functions. By defining the arc expression functions differently, it can help us model different events in the HES. Therefore, places in the SCCPN 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 SCCPN 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.

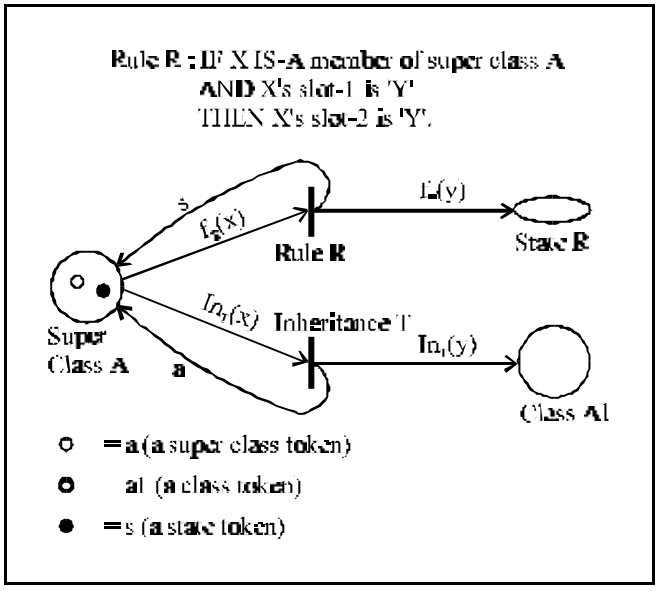


Figure 4.1a. Rule R with Inheritance (before firing) with an input token "a" & "s" in Super Class A.

The transition operations are represented by the arc expression functions. (e.g. A Rule R can be represented in SCCPN as shown in Figures 4.1a, 4.1b and 4.1c)

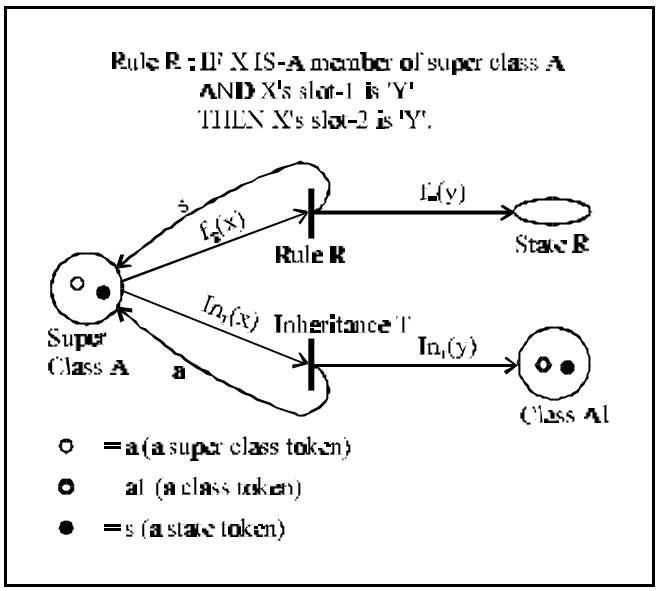


Figure 4.1b. Rule R with Inheritance (after firing Inheritance T) with an input token "a" & "s" in Super Class A.

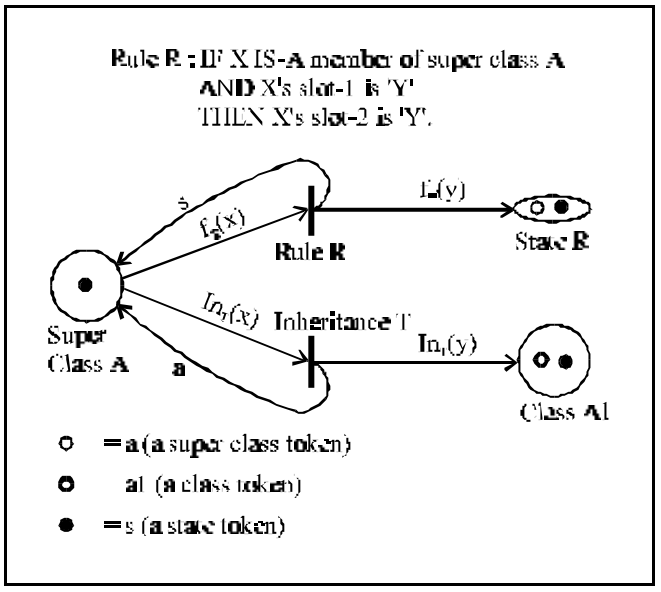


Figure 4.1c. Rule R with Inheritance (after firing both Rule R and Inheritance T) with output token "a" & "s" in State R and output token "a1" & "s" in Class A1. A state token "s" is also created in Super Class A.

Super Class A is a SCCPN Place with colour set that was used to represent the data structure of all object instances in Super Class A. Class A1 is a SCCPN Place with colour set that was used to represent the data structure of all object instances in Class A1. Rule R is a SCCPN 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 SCCPN 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. (i.e. 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 SCCPN network. Moreover, we can also examine the contents of these 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. (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.)

4.2.3. Rules with Message Passing

Places in the SCCPN 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.

4.2.4. Rules with Demons

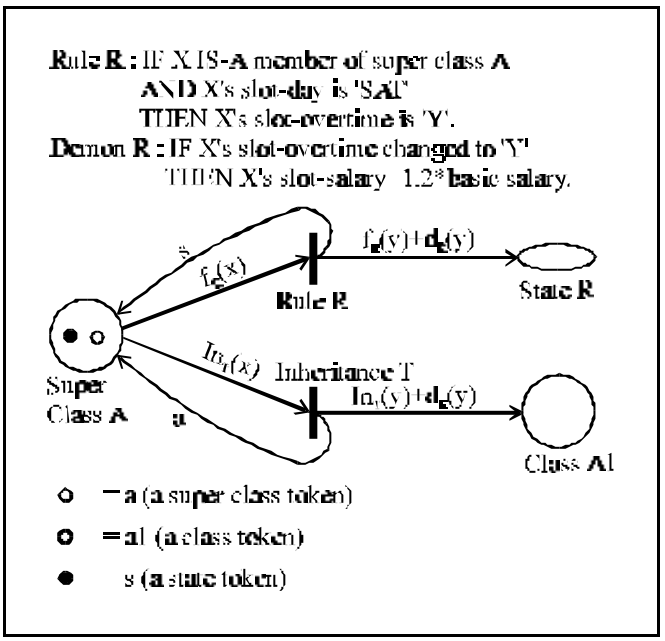


Figure 4.2a. Rule R with Demon (before firing) with an input token "a" and a state token "s" in Super Class A.

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 Figures 4.2a and 4.2b.

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

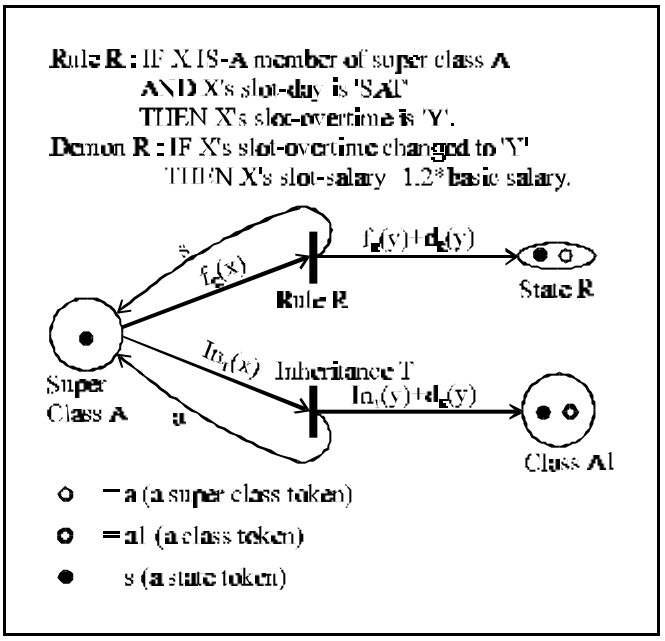


Figure 4.2b. Rule R with Demon (after firing) with output token "a" & "s" in State R and output token "a1" & "s" in Class A1. A state token "s" is also created in Super Class A.

4.2.5. Rules with Methods

Methods are procedures attached to an Object class, they can be represented by the Functions and Operations declarations in SCCPN. 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 SCCPN. 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 statements; 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 SCCPN as follows (Figures 4.3a-4.3.d, the self-loops are omitted for clarity reason)

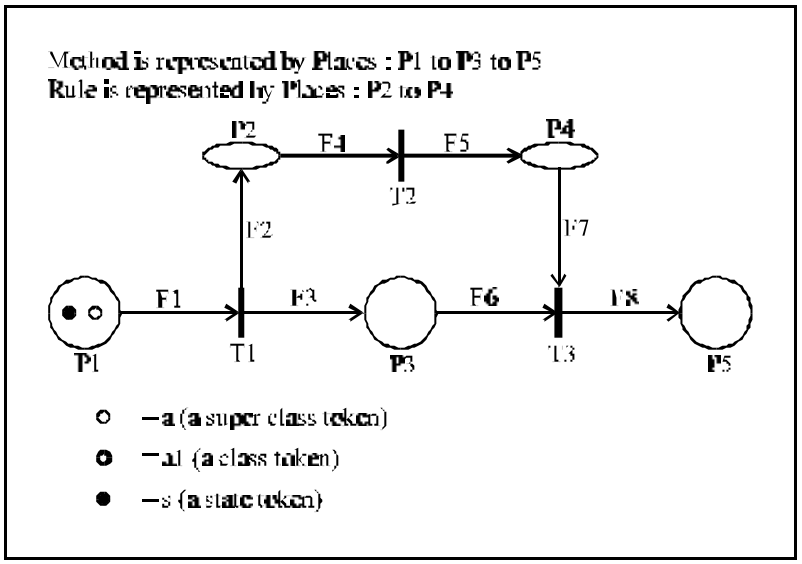


Figure 4.3a. Rule with Method (before firing) with an input token "a" and a state token "s" in P1.

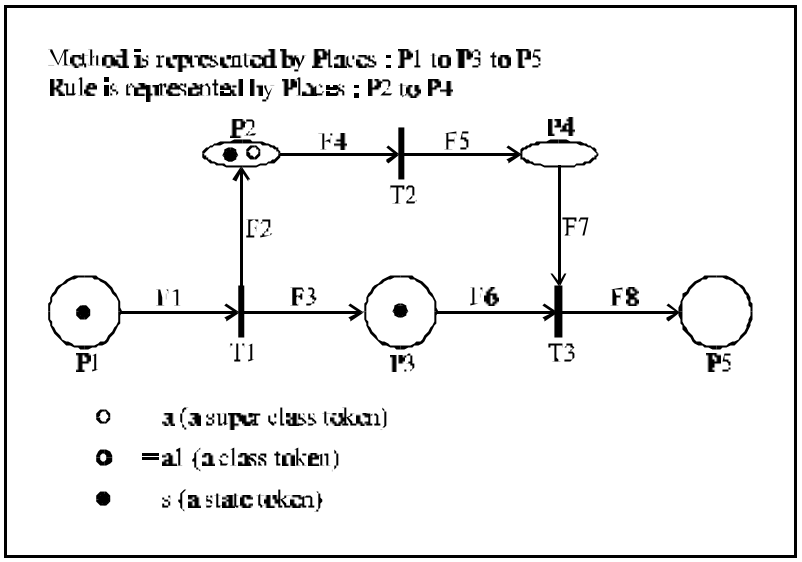


Figure 4.3b. Rule with Method (Rule is called by the Method). The token "a" was passed to P2 and a state token "s" was created in P1, P2 and P3 respectively.

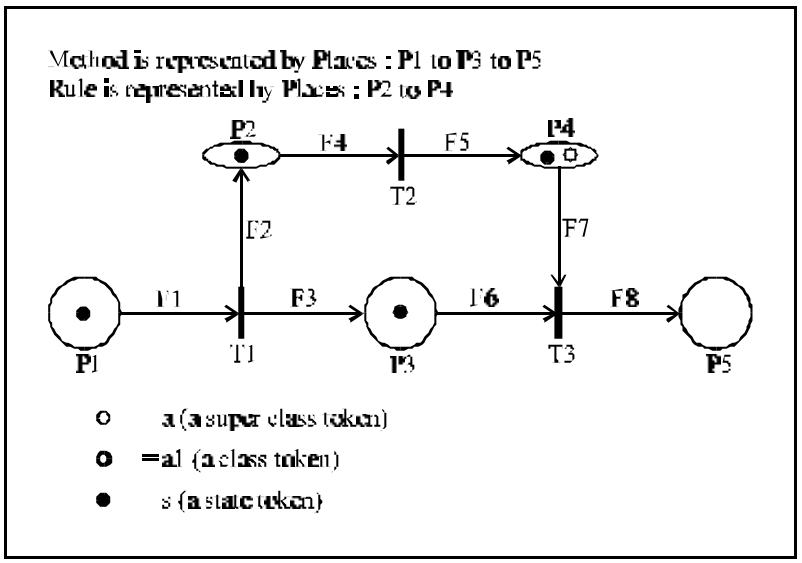


Figure 4.3c. Rule with Method (After firing). The token "a" is in P4 and a state token "s" in P1, P2 and P3 and P4 respectively.

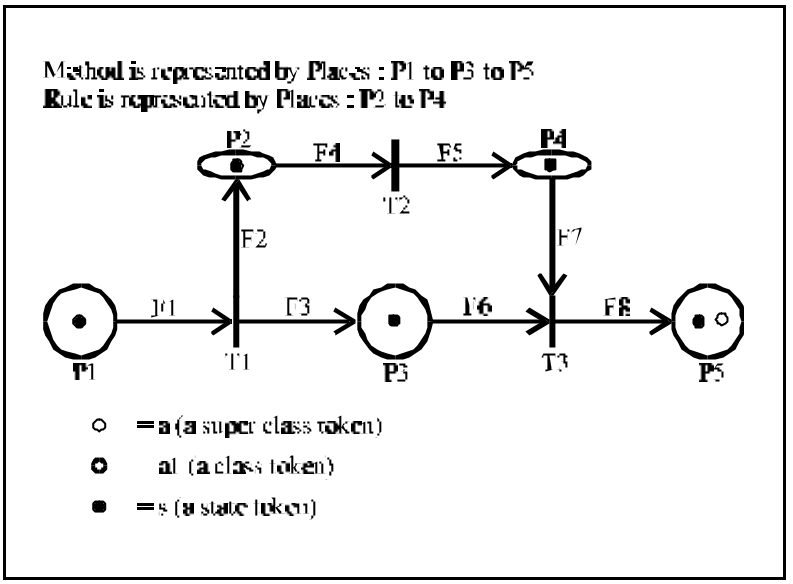


Figure 4.3d. Rule with Method (Method resumes control). The token "a" was passed to P5. A state token "s" was subsequently created in P1, P2, P3, P4 and P5 respectively.

The modelling of methods is divided into two parts. First the state of the method: (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 Figures 4.3a-4.3d, P1 to P3 to P5 represents three states of the Method described above. P2 to P4 represents 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 represents 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.

4.2.6. Rules with Instances

This is represented in SCCPN 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.

4.3. A Taxonomy of Anomalies

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. In a pure Rule-based system, errors and anomalies could include redundancy, dead-end rules, subsumption, duplication, circular rule sets, unsatisfiable conditions, missing rules..etc. Their verification are well documented in the literature (Gupta, U., 1991; Coenen, F. & Bench-Capon, T., 1993; Gamble R. F. et al., 1994; Liu N. K. & Dillon T., 1995; Nurrell, S. & Plant, R., 1996).

In a pure Frame-based system, errors 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 the inheritance of object properties.

Given that in a closed world situation in which a common concept is derived by a HES. The anomalies that are relevant to the correctness, consistency, and completeness of the HES, take the following forms:

4.3.1. Correctness

4.3.1.1. Redundancy

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

In the case of rules which have identical conditions and actions applied both to the parent object class and child object classes, this implies the existence of redundant rules.

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

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

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

Case II. Chained inference

Rule 3 : $A \Rightarrow C$

Rule 4 : $A' \Rightarrow B'$

Rule 5 : $B' \Rightarrow C'$

In the case of a chained inference, some rules could become redundant if the same result could be inferred by alternative transitions even the same input facts are given. (A'=A and C'=C because of inheritance and B' is not ascertainable through other rules). Rule 3 could become redundant as C' could be inferred by an alternative transition, Rule 5, via Rule 4.

4.3.1.2. Subsumption

Case I. Rule 6 is subsumed by Rule 7 (Condition part) between Parent Class and Child Classes.

Rule 6 : $A \wedge B \Rightarrow C \wedge D$

Rule 7 : $A' \Rightarrow C' \wedge D'$

Case II. Rule 8 is subsumed by Rule 9 (Action part) between Parent Class and Child Classes.

Rule 8 : $A \wedge B \Rightarrow C \wedge D$

Rule 9 : $A' \wedge B' \Rightarrow C'$

Case III. Rule 10 is subsumed by Rule 11 (Both Condition and Action) between Parent Class and Child Classes.

Rule 10 : $A \wedge B \Rightarrow C \wedge D$

Rule 11 : $A' \Rightarrow C'$

In a complex frame hierarchy which allows for multiple inheritance, checking for subsumption becomes more difficult because the problem becomes what characteristics the child inherits, and from which parent? The HES has to follow some sort of default orderings in inheritance, and this may lead to sets of conflicting traits which are even more complicated to verify.

4.3.1.3. Ambiguity

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

Rule 12: A IS-A member of ClassX $\vee$ A IS-A member of ClassY $\Rightarrow$ B

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

Rule 13: B $\Rightarrow$ A IS-A member of ClassX $\vee$ A IS-A member of ClassY

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 degree of strategy evaluation.

4.3.1.4. Circular Rule Sets

If a circular loop can occur when a set of rules among different object classes are fired, then these rules are considered as a circular rule set within the object hierarchy.

Case I. Self-reference rule

Rule 14: $A' \Rightarrow A \wedge B$

Case II. Self-reference chain of inference

Rule 15: $A \Rightarrow B \Rightarrow \bullet \bullet \bullet \bullet \bullet \bullet \Rightarrow P$

Rule 16: $P' \Rightarrow A$

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.

4.3.2. Consistency

4.3.2.1. Contradiction

If two rules have duplicate antecedents but in the consequents a clause is both affirmed and denied, we refer this as inconsistency. In an object hierarchy,

inconsistency may occur if a rule applied to the Parent object class but denied to the Child object classes.

Case I. Self-contradictory rule

Rule 17: $A \Rightarrow \neg A'$

Case II. Self-contradictory chain of inference

Rule 18: $A \Rightarrow B \Rightarrow \bullet \bullet \bullet \bullet \bullet \bullet \Rightarrow \neg A'$

Case III. Contradictory pairs of rules

Rule 19: $A \wedge B \Rightarrow C$

Rule 20: $A' \wedge B' \Rightarrow \neg C$

Case IV. Contradictory chains of rules

Rule 21: $A \Rightarrow B \Rightarrow \bullet \bullet \bullet \bullet \bullet \bullet \Rightarrow P$

Rule 22: $A' \Rightarrow \neg P$

4.3.2.2. 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.

Rule 23: $A \Rightarrow B$

A is not defined in the slot of the class hierarchy.

4.3.2.3. Unnecessary IF condition

Rule 24: $A \wedge B \Rightarrow C$

Rule 25: $A' \wedge C \Rightarrow D$

When rule 25 is backward chained to rule 24, (i.e. in order that C is true, we have to check whether A is true and B is true). Rule 25 is equivalent to the testing of A', A and B, (Rule 26):

Rule 26: $A' \wedge A \wedge B \Rightarrow D$

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

4.3.3. Completeness

4.3.3.1. Unreachability

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

Rule 27: ClassA $\wedge$ ClassA' $\Rightarrow$ C

Rule 28: ClassB $\wedge$ ClassC $\Rightarrow$ D

In Rule 27, if Class A is the Parent and Class 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 28, Class B and Class C are both children of Class A, it is not possible for any object instance to be both belonging to two different mutually exclusive classes.

Case II. Mutually exclusive classes chains

Rule 29: Class$A \wedge B \wedge C \Rightarrow P$

Rule 30: Class$A' \wedge P \Rightarrow Q$

If rule 30 is backward chained to rule 29, this causes an unreachable condition because rule 29's condition part and rule 30's condition parts are having mutually exclusive class instantiation.

A broad categorization of anomalies pertaining to knowledge verification of HES was given. This was classified in terms of sub-problems related to correctness, consistency and completeness in hybrid knowledge base. The anomalies take the form of Redundancy, Subsumption, Ambiguity, Circular Rule Sets, Contradiction, Deadend, Unnecessary IF condition and Unreachability. It is noted that significant effect stems from the chained inference in the hybrid knowledge base. Consequently, a mechanism for the detection and location of these anomalies appears to be essential, which is part of the subjects for knowledge verification.

4.4. Summary

An informal description of the method to model Hybrid Expert Systems is described. (The formal definitions of the method will be given in the next chapter). This is based on the notion of State Controlled Coloured Petri Nets (SCCPNs). The object classes are represented by the compound colour set; the production rules' transition operations are represented by the arc expression functions; the inheritance of the properties from classes are represented by another type of transitions operations; message passing is modelled by defining arc expression such that they directly read and write data to token's data slots; Demon is represented by a Places/Transition tuple and Methods are procedures attached to an Object class, they are represented

by the Functions and Operations declarations in SCCPN. In HES, there should be a set of reasoning strategies. Two common ones are: (1) Backward Chain with Inheritance (i.e. Goal directed search with inheritance as one of the means to establish the rule chains linking up different Object Classes); (2) Forward Chain with Inheritance (i.e. Data directed search with inheritance as one of the means to establish the rule chains linking up different Object Classes).

CHAPTER 5. A FORMAL METHODOLOGY FOR MODELLING RULE/FRAME-BASED HES USING STATE CONTROLLED COLOURED PETRI NETS (SCCPNs)

5.1. Fundamental Principles

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. 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 the 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 follows.

DEFINITION 5.1. A HES is defined as a tuple given by: HES = (C, A, D, M, I, H, 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 data type.

D = a finite set of demon functions. Each function is defined from A into an expression such that: $\forall a \in A: D(a) \in A$. (This means the demon functions can only change a slot's value within the same object instance. Besides, this demon function: D(a) generates only one output from each given input "a").

M = a finite set of methods. Each method is defined as a function which takes a number of arguments from an object$\in$ C and returns a result to the object$\in$ C.

I = a specific object element from an object class C.

H = an inheritance relation. It is defined from the partially ordered relations in C.

R = The rules are composed of predicates which are used as functions that map object arguments into TRUE, FALSE values represented by binary truth values 1,0, respectively. (One of the predicates is the IS-A predicate which is used to specify the class of objects which a particular rule can be applied). All literals used in both the condition and action predicates must come from the attribute set A.

S = a finite set of reasoning strategies. The two common HES reasoning strategies are: Backward Chain with Inheritance and Forward Chain with Inheritance.

Explanations: 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, integer or

real. Each specific object element is called an instance of the Object Class and will have different attribute values of the variables. 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, the problem becomes what characteristics the child inherits, and from which parent? The HES has to follow some sort of default orderings on inheritance (Dori, D. & Tatcher, E., 1994; Willis, C.P., 1996), and this may lead to sets of conflicting traits which are even more complicated to verify. Therefore, our present analysis is concentrated on simple inheritance only.

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 functions attached to some Object Class, that will be executed whenever a signal is passed through. Each method is defined as a function which takes a number of arguments and return a result.

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. The two common ones are :

- Backward Chain with Inheritance: Goal directed search with inheritance as one of the means to establish the rule chains linking up different Object Classes.

- Forward Chain with Inheritance: Data directed search with inheritance as one of the means to establish the rule chains linking up different Object Classes.

As HES is modelled by SCCPN, a mapping between the two structures is necessary, and is given in Table 5.1.

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

Table 5.1. Conceptual interpretation of HES in SCCPNs.

As shown in Table 5.1 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 thesis is an extension of State Controlled Petri Nets proposed by (Liu, N. K. & Dillon T., 1995), and Coloured Petri Nets proposed by (Jensen, K., 1995, 1996) and is specified as follows.

DEFINITION 5.2. A SCCPN can be defined as a 10-tuple given by = (*S*, *P*, *T*, *D*, *F*, *A*, *N*, *C*, *E*, *I*), where satisfying the requirements below:

$\Sigma = \{ w_1, w_2, ..., w_i \}$, 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 or 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 output 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.

$C : P \rightarrow S$, 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 $\forall a \in A : [Type(E(a)) = C(p(a))_{MS} \wedge Type(Var(E(a))) \in S]$ 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 5.3. For each transition $t_j \in T$ in a net N,

$\tilde{O}_s(t_j) \cap \ddot{O}_s(t_j) \neq \emptyset$,

$\tilde{O}_c(t_j) \cap \ddot{O}_c(t_j) = \emptyset$,

$\tilde{A}_c(t_j) \cap \ddot{A}_c(t_j) \neq \emptyset$,

$\tilde{A}_s(t_j) \cap \ddot{A}_s(t_j) = \emptyset$,

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 5.4. 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 5.5. 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 5.6. 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 5.7. 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.

DEFINITION 5.8. 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 5.9. A finite occurrence sequence is a sequence of markings and steps: $M_1[Y_1>M_2[Y_2>M_3\ldots\ldots M_n[Y_n>M_{n+1}$ such than n ∈ Natural Number and $M_i[Y_i>M_{i+1}$ for all $i\in 1\ldots..n$. 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 5.10. 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 $Y_1, Y_2 \ldots . Y_n$ such that: $M_1[Y_1>M_2[Y_2>M_3 \ldots \ldots Y_n>M''$. M'' is reachable from M' in n steps. A firing or occurrence sequence is denoted by $\sigma=(Y_1, Y_2 \ldots \ldots Y_n)$

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

DEFINITION 5.11. The full occurrence graph of a SCCPN is the directed graph OG=(V, A, N) where:

a. $V=[M_0>$

b. $A=\{(M_1,b,M_2) \in V x B E x V | M_1[b>M_2\}$.

c. $\forall a=(M_1,b,M_2) \in A: N(a)=(M_1,M_2)$.

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.

5.2. Description and Properties

The logical predicate becomes true by the presence of a state token and the transition associated with this predicate will become active by the presence of the corresponding object class token (instance) and provided that the slots attributes in the object class instance satisfies the transition condition. The transition is enabled and is ready for firing. For simplicity reasons, without taking any transition conditions or transition operations into consideration, we can *minimally* enable a specific transition and then check the reachability set for any irregularities of

predicate places. In this representation, a marking M is composed of M_c that depicts the marking for the class places and M_s that depicts the marking for the state places in the SCCPN. A transition t_j is represented by a t-vector. For verification purposes, we define that:

DEFINITION 5.12. A transition t_j is minimally active if

$$M_C = \begin{Bmatrix} 1 & if\ p_{ci} \in (\tilde{A}_c(t_j) \cup \tilde{O}_c(t_j)) \\ 0 & otherwise \end{Bmatrix}$$

DEFINITION 5.13. A transition t_j is minimally enabled if t_j is both minimally active and that

$$M_S = \begin{Bmatrix} 1 & if\ p_{si} \in (\tilde{A}_s(t_j) \cup \tilde{O}_s(t_j)) \\ 0 & otherwise \end{Bmatrix}$$

and

$$\sum E(p_i, t_j) < b > \leq (M_c(p_{ci}) \cup M_s(p_{si}))$$

DEFINITION 5.14. T_k that contains a group of transitions $\{t_n\}$ is said to be minimally active if $\forall j=1,2,..n$, $t_j \in T_k$, $\exists\ p_i \in (\tilde{A}_c(t_j) \cup \tilde{O}_s(t_j)) \subseteq (\tilde{A}_c(T_k) \cup \tilde{O}_s(T_k))$, such that

$$M_C = \begin{cases} 1 & if\ p_{ci} \in (\tilde{A}_c(t_j) \cup \tilde{O}_c(t_j)) \\ & and\ p_{ci} \notin (\ddot{A}_c(t_j) \cup \ddot{O}_c(t_j)) \\ 0 & otherwise \end{cases}$$

Note that the self-loop arc corresponding to each input place does not cause a repeated firing of transitions. In the absence of any self-reference rule, the set of

input places and that of output places with respect to the transition in SCCPN are always disjointed.

DEFINITION 5.15. T_k that contains a group of transitions $\{t_n\}$ is said to be minimally enabled if $\forall j=1,2,..n$, $t_j \in T_k$, $\exists\ p_i \in (\tilde{A}_c(t_j) \cup \tilde{O}_s(t_j)) \subseteq (\tilde{A}_c(T_k) \cup \tilde{O}_s(T_k))$, such that

$$M_s = \begin{cases} 1 & if\ p_{si} \in (\tilde{A}_s(t_j) \cup \tilde{O}_s(t_j)) \\ & and\ p_{si} \notin (\ddot{A}_s(t_j) \cup \ddot{O}_s(t_j)) \\ 0 & otherwise \end{cases}$$

and

$$\sum E(p_i, t_j) < b > \leq (M_c(p_{ci}) \cup M_s(p_{si}))$$

5.3. Modelling HES with SCCPNs

It is important to understand how to make use of State Controlled Coloured Petri Nets to model the knowledge structure and inference in HES. SCCPNs inherit most of the mathematical properties from Coloured Petri Nets (Jensen, K., 1995, 1996), which enable the storage and recall of past and present states of machine problem solving processes and the precalculation of results. Storage of states helps to implement high level problem solving by allowing the system to back-track through its problem solving process, resolve goal conflicts, and then resume the process from the last successfully completed design step. Precalculation of results allows the expert system to bypass some or all of the steps required during problem solving when it comes across previously encountered or predetermined situations.

In addition to the representation advantages, SCCPNs can provide a clear indication of data dependencies, which would make it possible for us to exploit parallelism in the problem domain. The analysis of SCCPNs can reveal bottlenecks or other

possible anomalies in the procedural flow associated with the hybrid knowledge base. The following schema is used to represent some typical rules which attached to the object hierarchy. The self-loop arc is omitted for clarity reason.

5.3.1. Correctness

5.3.1.1. 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 & 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 Figure 5.1.

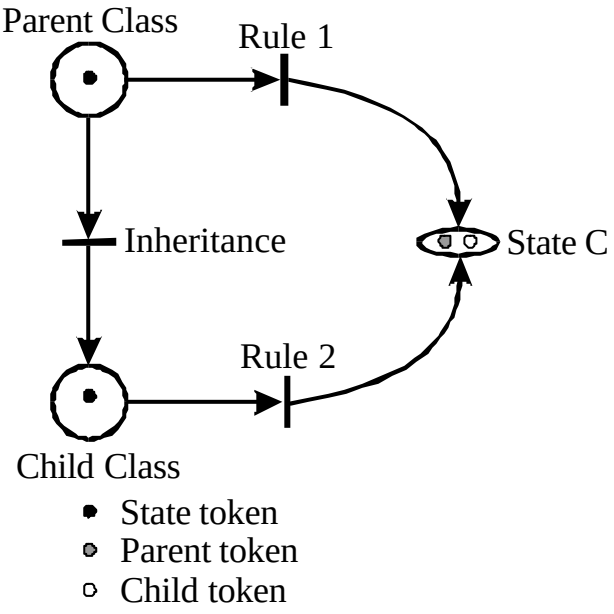


Figure 5.1. SCCPN showing Redundancy Case I

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⇒C

Rule 2 : A'⇒B'

: :

Rule N : N'⇒C'

In general the chain inference can be represented by the following SCCPN in Figure 5.2.

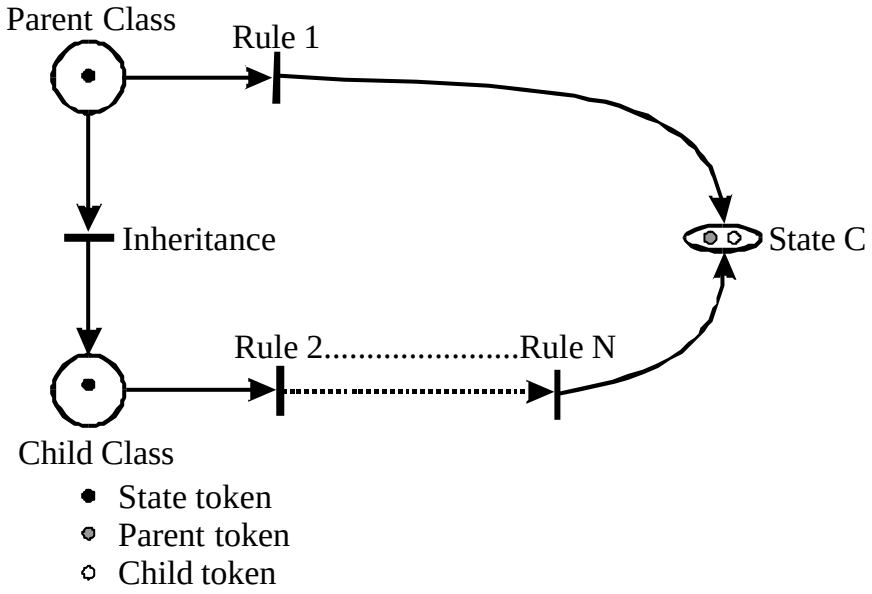


Figure 5.2. SCCPN showing Redundancy Case II

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 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.

5.3.1.2. 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 : $A' \Rightarrow C' \wedge D'$

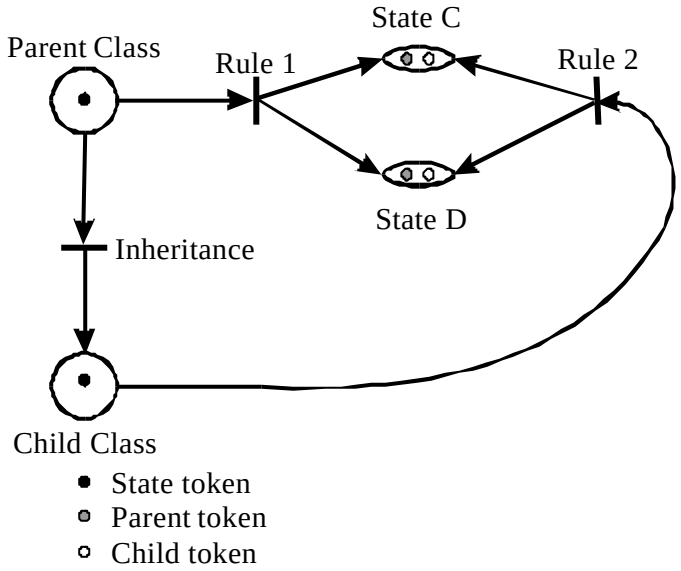


Figure 5.3. SCCPN showing Subsumption Case I

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 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 : $A \wedge B \Rightarrow C \wedge D$

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

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

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

Rule 2 : $A' \Rightarrow C'$

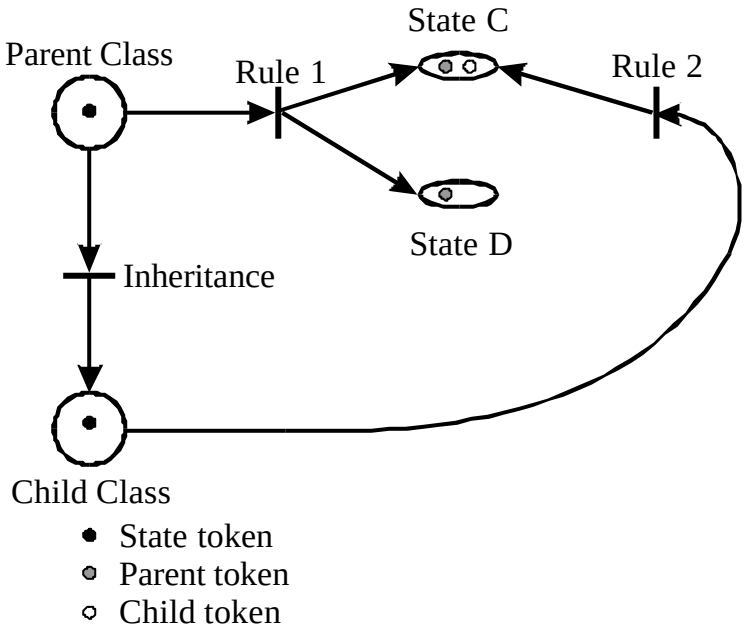


Figure 5.4. SCCPN showing Subsumption Case II and III

Subsumption Case II and Case III are both represented by Figure 5.4. 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' 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.

5.3.1.3. Ambiguity

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

Rule 1 : A IS-A member of ClassX $\vee$ A IS-A member of ClassY $\Rightarrow$ C

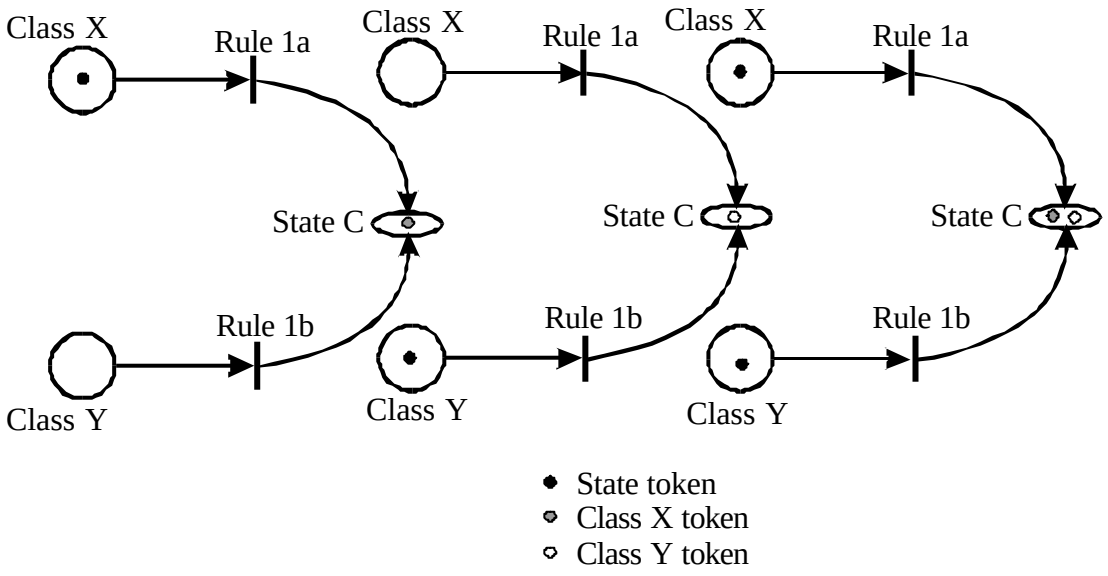


Figure 5.5. SCCPN showing Ambiguity Case I

Rules with inclusive disjunction of IS-A conditions from different Object Classes can be represented in a slight different fashion. In Figure 5.5, 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 ClassX $\Rightarrow$C

Rule 1b : A IS-A member of ClassY $\Rightarrow$C

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

Rule 1 : C$\Rightarrow$A IS-A member of ClassX $\vee$ A IS-A member of ClassY

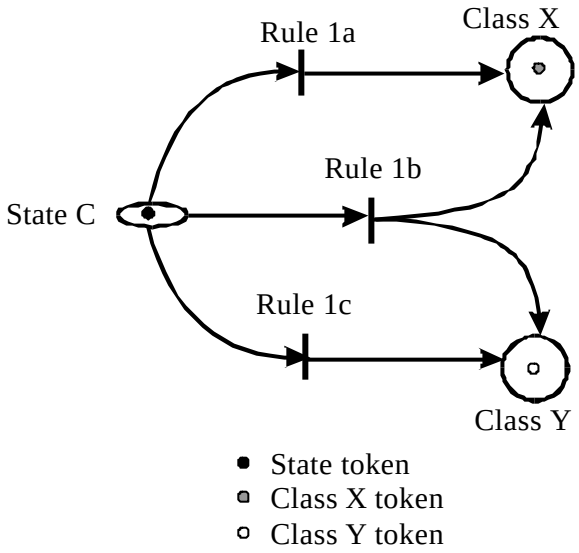


Figure 5.6. SCCPN showing Ambiguity Case II

Rules with inclusive disjunction of IS-A actions from different Object Classes can be represented by the alternative sets of marking as shown in Figure 5.6. 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 degree of strategy evaluation. i.e.

Rule 1a : C$\Rightarrow$A IS-A member of ClassX

Rule 1b : C$\Rightarrow$A IS-A member of ClassX $\wedge$ A IS-A member of ClassY

Rule 1c : C$\Rightarrow$A IS-A member of ClassY

5.3.1.4. Circular Rule Sets

Case I. Self-reference rule

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

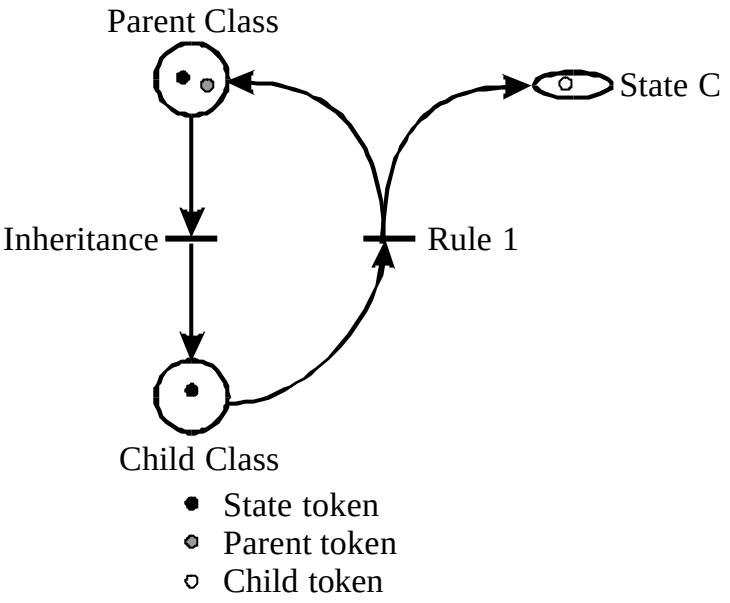


Figure 5.7. SCCPN showing Circular Rule Sets Case I

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 Figure 5.7. 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

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 following SCCPN in Figure 5.8.

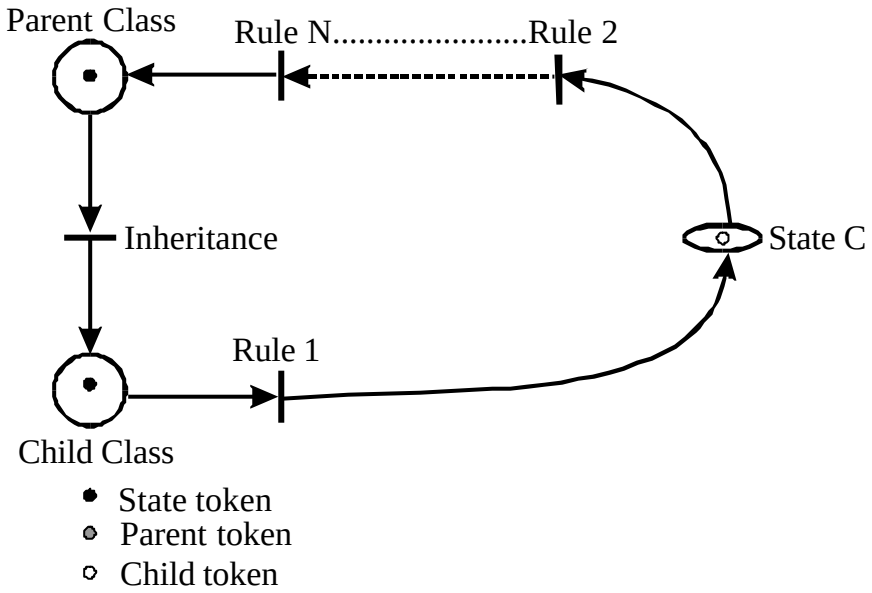


Figure 5.8. SCCPN showing Circular Rule Sets Case II

5.3.2. Consistency

5.3.2.1. Contradiction

Case I. Self-contradictory rule

Rule 1 : A⇒C

Rule 2 : A'⇒¬C

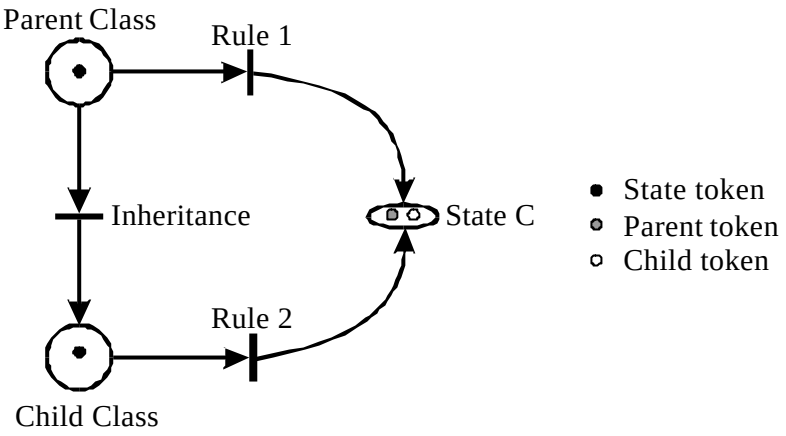


Figure 5.9. SCCPN showing Contradiction Case I

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.

Case II. Self-contradictory chain of inference

Rule 1: $B' \Rightarrow \neg C$

Rule 2 : $C' \Rightarrow D$

⋮ ⋮

Rule N: $N' \Rightarrow B$

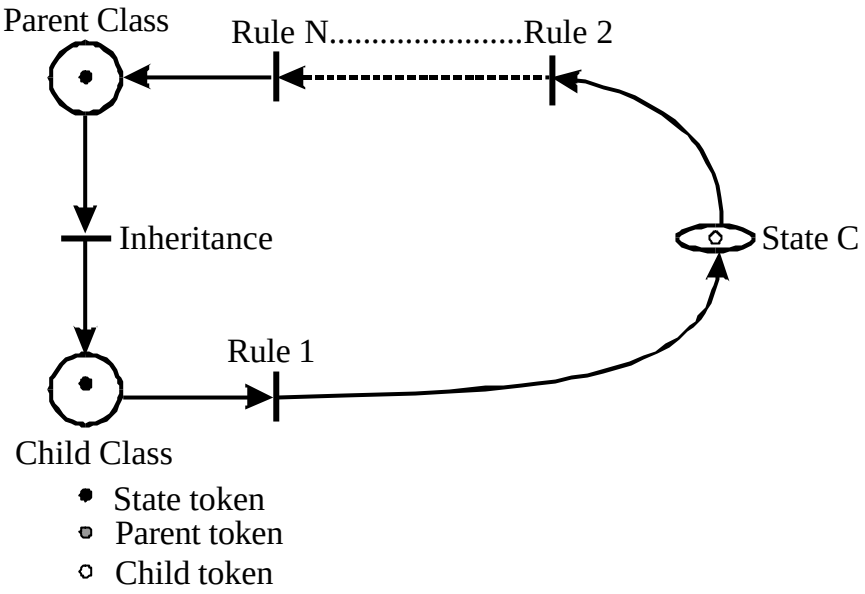


Figure 5.10. SCCPN showing Contradiction Case II

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 : $A' \wedge B' \Rightarrow \neg C$

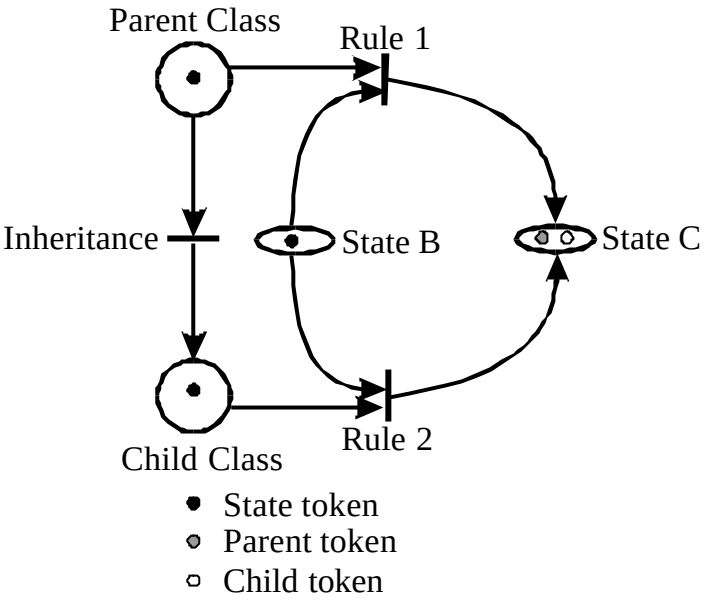


Figure 5.11. SCCPN showing Contradiction Case III

If we have a Parent token in Parent Class with A is TRUE, and a State token in State B indicating State B is 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 : $N \Rightarrow P$

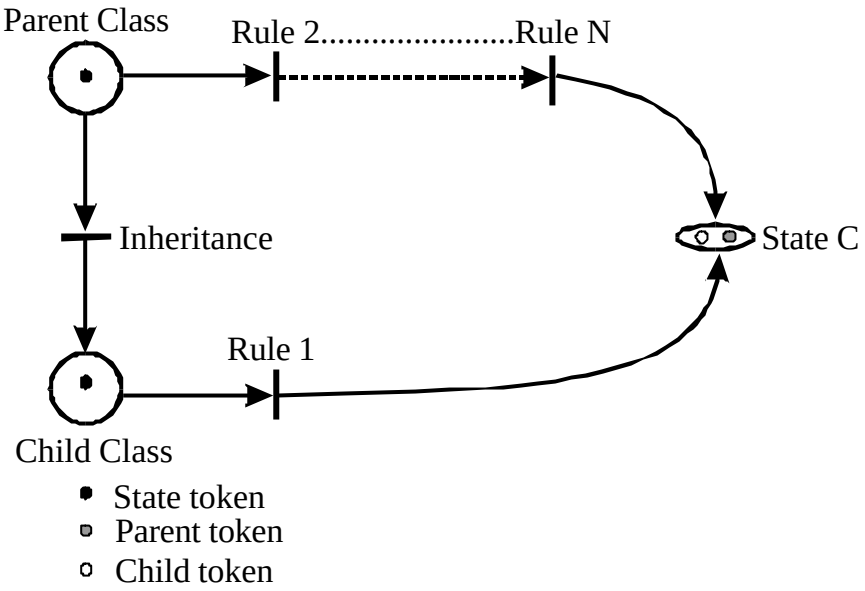


Figure 5.12. SCCPN showing Contradiction Case IV

5.3.2.2. 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. (Figure 5.13.)

Rule 1 : A⇒B

A is not defined in the slot of the class hierarchy.

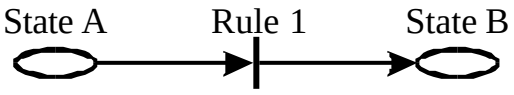


Figure 5.13. SCCPN showing Deadend

Since A is not defined, no tokens will be created in State A.

5.3.2.3. 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 : $X' \wedge X \wedge A \Rightarrow C$

Since X' and X are in inheritance relation, we may want to remove either the condition IF X' or IF X. (Figure 5.14.)

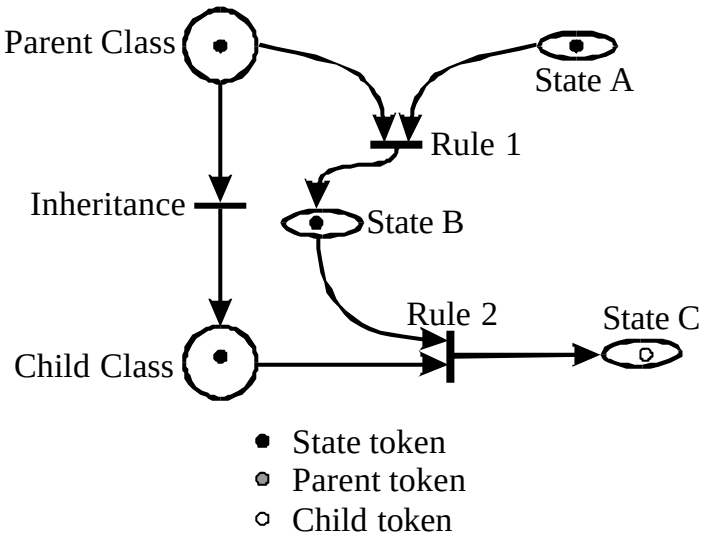


Figure 5.14. SCCPN showing Unnecessary IF condition

5.3.3. Completeness

5.3.3.1. Unreachability

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

Rule 1 : ClassA $\wedge$ ClassA'$\Rightarrow$C (applied to Parent Class)

Rule 2 : ClassA $\wedge$ ClassA'$\Rightarrow$C (applied to Child Class)

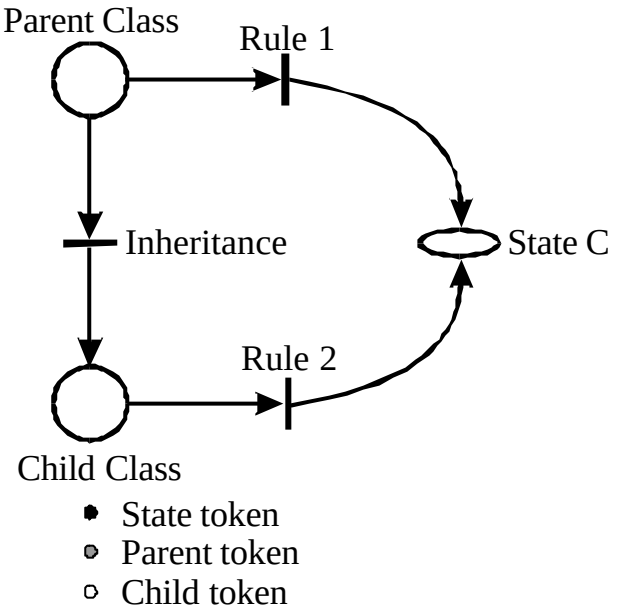


Figure 5.15. SCCPN showing Unreachability Case I.a

Rules with mutually exclusive classes can be represented by the alternative sets of rules in Figure 5.15. 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 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.

Rule 1 : ClassA $\wedge$ ClassB$\Rightarrow$C (applied to Class A)

Rule 2 : ClassA $\wedge$ ClassB$\Rightarrow$C (applied to Class B)

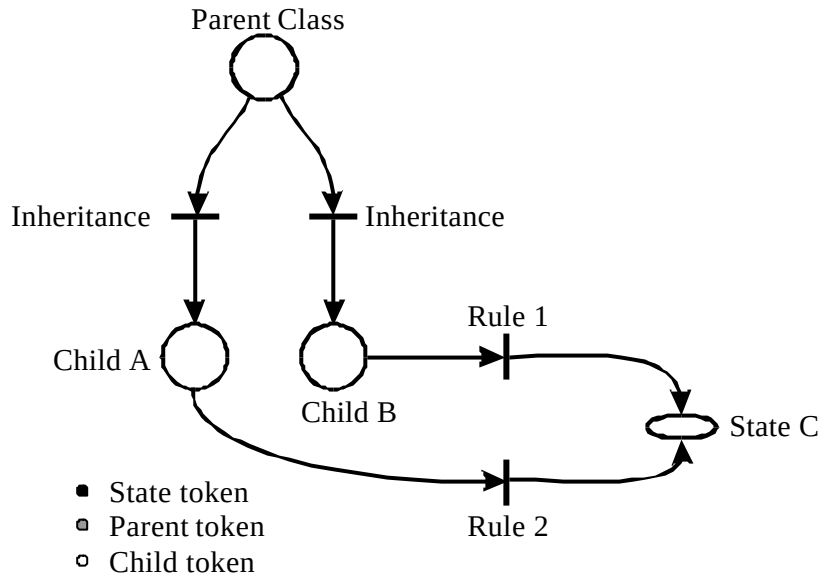


Figure 5.16. SCCPN showing Unreachability Case I.b

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 two different mutually exclusive classes.

Case II. Mutually exclusive classes chains

Rule 1 : ClassX$\wedge$A$\wedge$B$\Rightarrow$C

Rule 2a : ClassX'$\wedge$C$\Rightarrow$D (applied to Class X')

Rule 2b : ClassX'$\wedge$C$\Rightarrow$D (applied to State C)

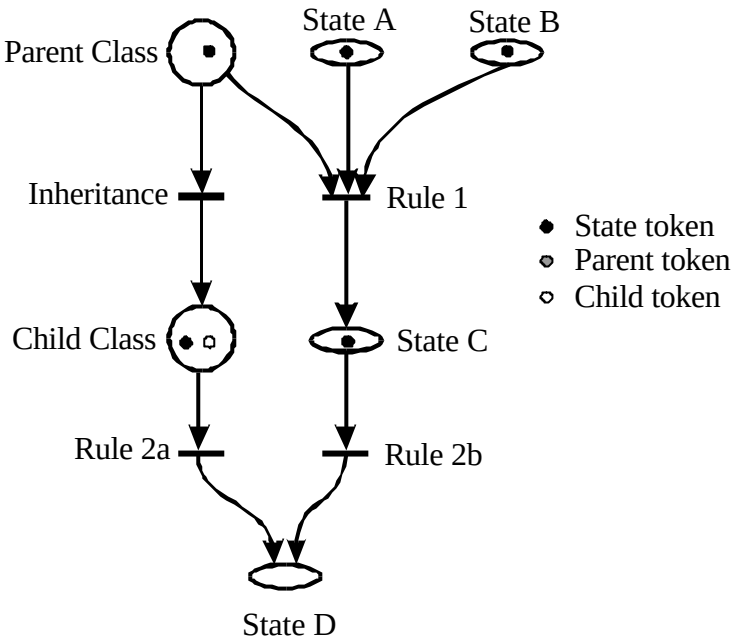


Figure 5.17. SCCPN showing Unreachability Case II

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.

5.4. Knowledge Inference in SCCPN Modelling

Basically, the methods for knowledge inference comprise of event driven and goal driven reasoning. The reasoning strategy for dynamic knowledge inference in SCCPN is event driven reasoning, because the reasoning process is based on the occurrences of events. The goal of the reasoning for SCCPN is to determine the subsequent events (activities) based on current events.

The initial marking and colours of the net determines the initial state of the system. Subsequent markings and colours of the tokens contribute to a reachability set which can reflect the degree of inference at different level, stemming from the initial event. The transitions of SCCPN model are structured to be in one direction only with the exception for the self-loop that is associated with each input places. When a

transition t (either representing a rule or a inheritance relation) becomes active and fired, the inference proceeds in a forward direction. Subsequently, SCCPNs support a forward chaining (events driven) paradigm of inference. In order to model and allow for simulating the backward chaining (goal driven) behaviour, a concept for backward enabled transition (Liu, N.K., 1991) have been introduced. It is defined that a transition t for P→Q is backward enabled if its inference proceeds in a backward direction as if it were for P←Q. Input places and output places are interchanged accordingly to accomplish the changes.

5.5. Summary

The factual and inference knowledge in a HES can be formulated in SCCPNs. State tokens are used to indicate the validity of a fact which is maintained by the presence of a self-loop in the net. A methodology for modelling a variety situations including Redundancy, Subsumption, Ambiguity, Circular Rule Sets, Contradiction, Deadend, Unnecessary IF Condition and Unreachability of rule sets attached to the object hierarchy is given. This allows for the checking of alternative markings at any level of inference. SCCPN is a event driven inference paradigm. This is to facilitate the generation and analysis of the knowledge inference in a HES being modelled by the net.

CHAPTER 6. AN APPLICATION OF THE FORMAL VERIFICATION METHOD

6.1. A Personnel Selection Hybrid Expert System

To illustrate the HES modelling by our proposed SCCPN methodology, we adopt a simplified version of a Personnel Selection Expert System currently being used in Hong Kong (Huen, H.S.M., 1993). 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 6.1 and details of relevant frames in the hierarchical structure are given below.

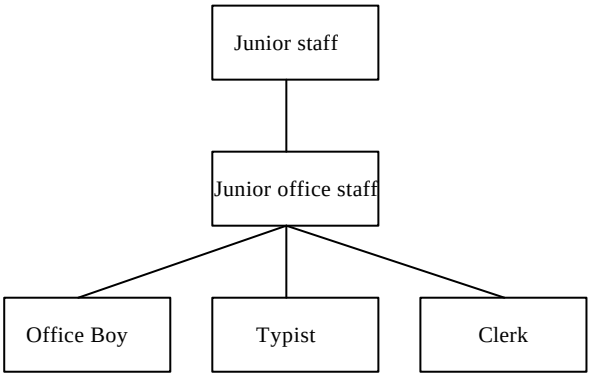


Figure 6.1. The Frame Hierarchy

A Junior Staff Frame:

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

Table 6.1. A Junior Staff Frame

A Junior Office Staff Frame:

Slot Name	Value	Type	Demon
Job Grade	Junior Office Staff	String	
Name		String	
Address and Telephone		String	
Hong Kong Identity Card Number (HKID)		String	IF possess HKID THEN Privilege is Local ELSE Privilege is Overseas
Privilege		Local / Overseas	
Sex		M/F	
*Office Hours	9 am - 5 pm	Time	
*Qualification Requirement	Five passes in HKCEE	String	
*Department	General Secretariat	String	
*Salary Pay Scale	Point 1 to Point 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		

* denote slots inherited from parent frame

Table 6.2. A Junior Office Staff 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 subsumed 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 chapters. It is noted that a frame is equivalent to a data structure with various type 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∈ II, b∈ BB,....h∈ HH)
Color OO =	with Yes | No;

Color PP = product OO * KK; (for state token, the first variable o∈OO is the state of the predicate, (i.e. 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. The second variable k∈KK is to record which class object has fired the rule))

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 an 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: **AÙBÙC Þ X**

Rule 2: **A1ÙBÙC Þ X**

Rule 3: **A1ÙBÙCÙD Þ X**

Rule 4: **A1ÙE Þ ØC**

Rule 5: **AÙE Þ C**

Rule 6: **A1ÙØFÙØG Þ Y**

Rule 7: **AÙY Þ H**

Rule 8: **A1ÙA Þ K**

Rule 9: **A2ÙY Þ Z**

Rule 10: **A Þ L**

Rule 11: **L Þ A1**

Rule 12: **A1 Þ 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

ØC = Seniority is Not High

D = Local citizen

E = Years of service is greater than Five

ØF = Knowledge of work is Not Good

ØG = 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

The Hybrid Expert System is represented by a State Controlled Coloured Petri Net shown in Figure 6.2, according to the methodology proposed in the previous chapters. 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 represent the predicates of these rules.

To illustrate the application of our formal methodology, the net in Figure 6.2 are representing the followings:

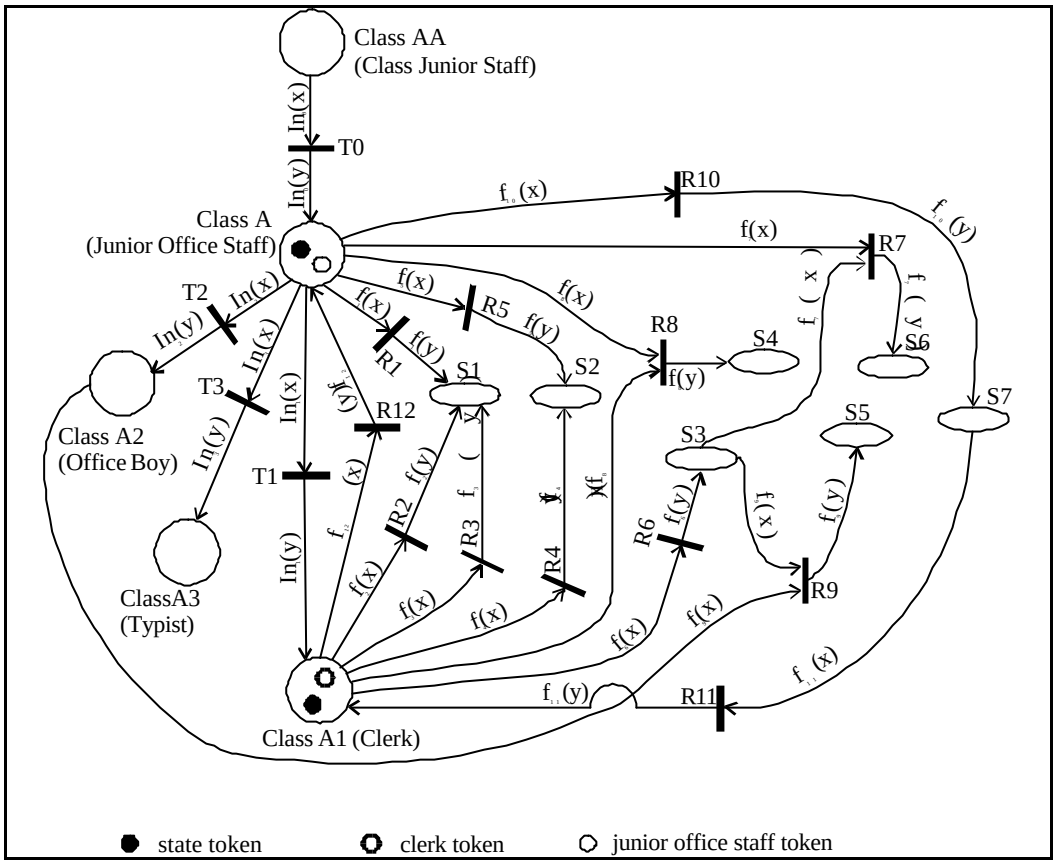


Figure 6.2. SCCPN representation of the given HES

Σ = { *Color A, Color B, …Color P*}, sixteen colour sets used,

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

P_c = {Junior Staff, Junior Office Staff, Office Boy, Typist and Clerk }, five places that model the classes of the HES,

P_r = { ClassA, ClassA1, S1, S2, S3,…S7}, nine places that model the predicates of the production rules,

$P_c \cap P_r$: the intersection of $P_c \cap P_r$ ={ClassA, ClassA1}, 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 = {*T0, T1, T2, T3*}, four transitions that are connected to and from *class places*,

T_r = { R1, R2, R3, ….R12}, twelve transitions that are connected to and from *predicate places*,

$T_c \cap T_r = \varnothing$,

$D = \{ d_1, d_2, ..., d_n \}$, 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

$\tilde{A}_c(t)$= *{ClassAA if T0, ClassA if T1, ClassA if T2 and ClassA if T3}*

$\ddot{A}c(t)$= *{ClassA if T0, ClassA1 if T1, ClassA2 if T2 and ClassA3 if T3}*

$\tilde{A}s(t)$= *{ClassAA if T0, ClassA if T1, ClassA if T2 and ClassA if T3}*

$\ddot{A}s(t)$= *{ClassA if T0, ClassA1 if T1, ClassA2 if T2 and ClassA3 if T3}*

Predicate : {$\tilde{O}_c$, $\ddot{O}_c$, $\tilde{O}_s$, $\ddot{O}_s$} where

$$\tilde{O}_c(t)=\begin{cases} ClassA\ if\ R1, ClassA1\ if\ R2, ClassA1\ if\ R3, \\ ClassA1\ if\ R4, ClassA\ if\ R5, ClassA1\ if\ R6, \\ ClassA + S3\ if\ R7, ClassA + ClassA1\ if\ R8, \\ ClassA2 + S3\ if\ R9, ClassA\ if\ R10, \\ S7\ if\ R11,\ A1\ if\ R12 \end{cases}$$

$$\ddot{O}_c(t)=\begin{cases} S1\ if\ R1, S1\ if\ R2, S1\ if\ R3, \\ S2\ if\ R4, S2\ if\ R5, S3\ if\ R6, \\ S6\ if\ R7, S4\ if\ R8, S5\ if\ R9, S7\ if \\ R10, ClassA1\ if\ R11, ClassA\ if\ R12 \end{cases}$$

$$\tilde{O}_s(t)=\begin{cases} ClassA\ if\ R1, ClassA1\ if\ R2, ClassA1\ if\ R3, \\ ClassA1\ if\ R4, ClassA\ if\ R5, ClassA1\ if\ R6, \\ ClassA + S3\ if\ R7, ClassA + ClassA1\ if\ R8, \\ ClassA2 + S3\ if\ R9, ClassA if\ R10, \\ S7\ if\ R11, A1\ if\ R12 \end{cases}$$

$$\ddot{O}_s(t)=\begin{cases} S1\ if\ R1, S1\ if\ R2, S1\ if\ R3, \\ S2\ if\ R4, S2\ if\ R5, S3\ if\ R6, \\ S6\ if\ R7, S4\ if\ R8, S5\ if\ R9, S7\ if \\ R10, ClassA1\ if\ R11, ClassA\ if\ R12 \end{cases}$$

$C : P \rightarrow S$, a *colour function*, it maps each place into a colour set, {Color N + Color P} for all places

$E(a) = \{In_0(x), In_0(y), ..In_3(x), In_3(y);$ *and* $f_1(x), f_1(y),\ldots f_{12}(x), f_{12}(y)\}$

$I =$ The initial junior staff token and state token in ClassA.

6.2. Analysis of the Personnel Selection System using SCCPNs

The major analysis technique, within the context of Expert System verification, is the use of reachability tree which represents the reachability set of the SCCPN (or occurrence graph in (Jensen, K., 1995,1996)'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 SCCPN. However, recent research (Li, X. et al., 1993; Christensen, S. & Petrucci, L., 1995; Kemper, P. 1996; Kondratyev, A. 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. 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 SCCPN as follows:

```
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 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.

6.2.1. Detection of Errors and Anomalies in HES

6.2.1.1. Correctness

6.2.1.1.1. Subsumption

Analysis of the network will show the presence of subsumption in the HES (Figure 6.3a). 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.

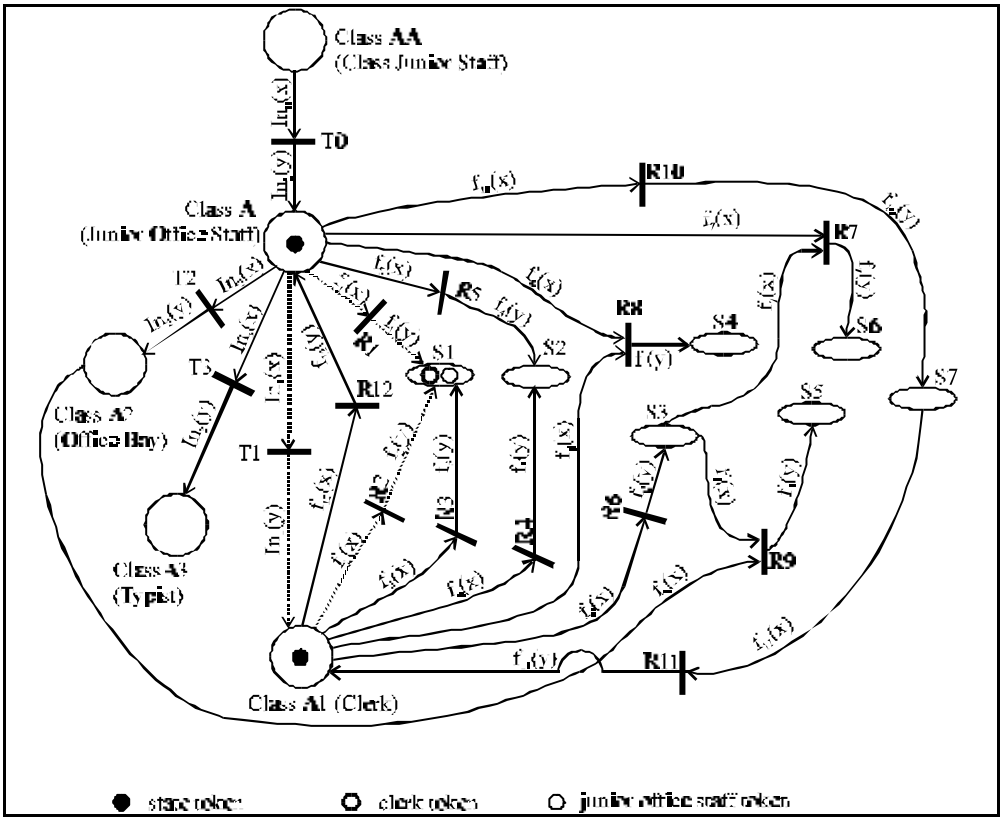


Figure 6.3a. SCCPN representation showing the events of subsumption, Case I

Figure 6.3b 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 markings 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, i.e. '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 is subsumed by R1 because R2 is just a more specific case of R1. (i.e. Clerk is the child of Junior Office Staff).

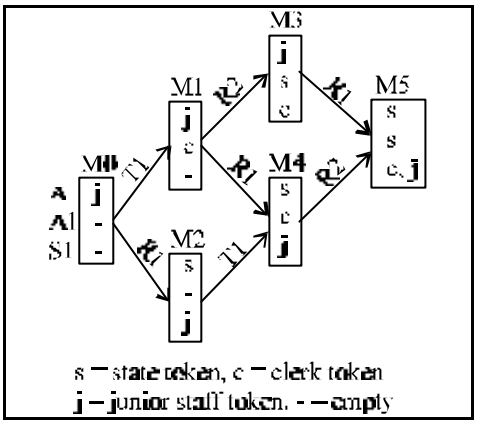


Figure 6.3b. Reachability graph due to the firing of R1 and R2

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 is subsumed by Rule X because Rule Y is just a more specialized case of Rule X. (i.e. whenever Rule X succeeds, Rule Y 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 6.4a. 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.

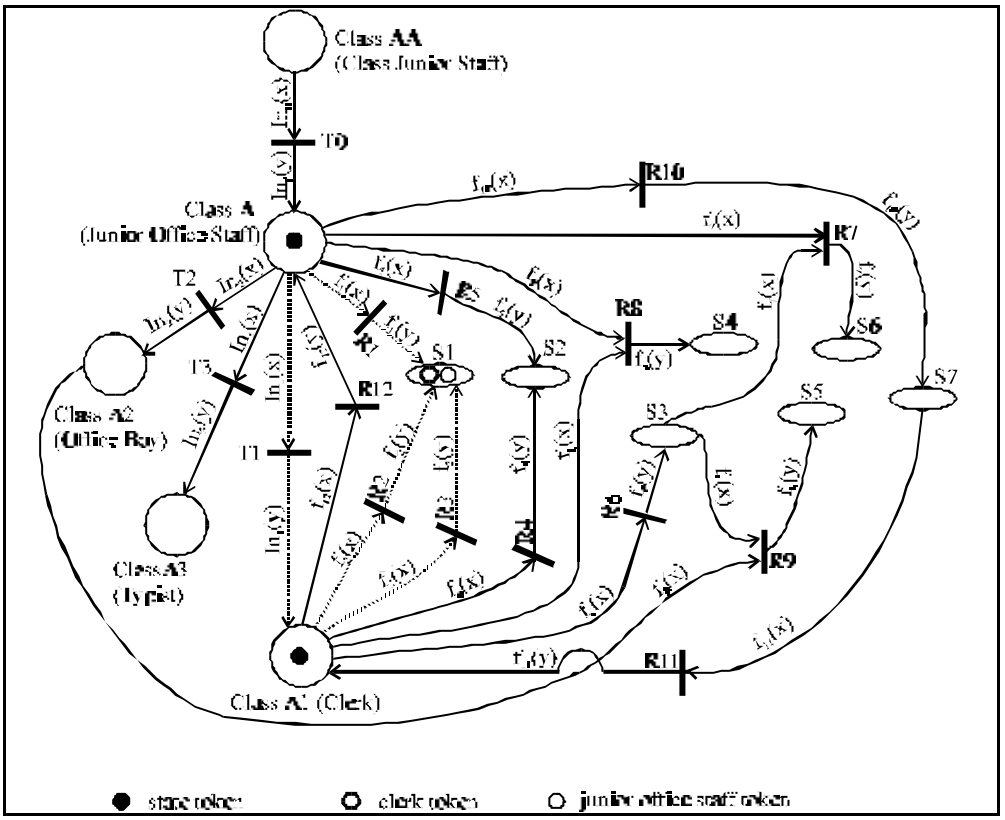


Figure 6.4a. SCCPN representation showing the events of subsumption, Case II

Figure 6.4b 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, i.e. 'YES'. Therefore, Rule 3 is subsumed by Rule 2 because the two transitions R2 and R3 can be enabled in A1 and their final marking is the same.

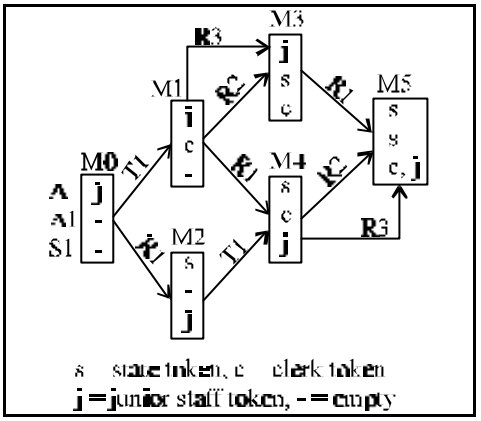


Figure 6.4b. Reachability graph due to the firing of R1, R2 and R3

6.2.1.1.2 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.

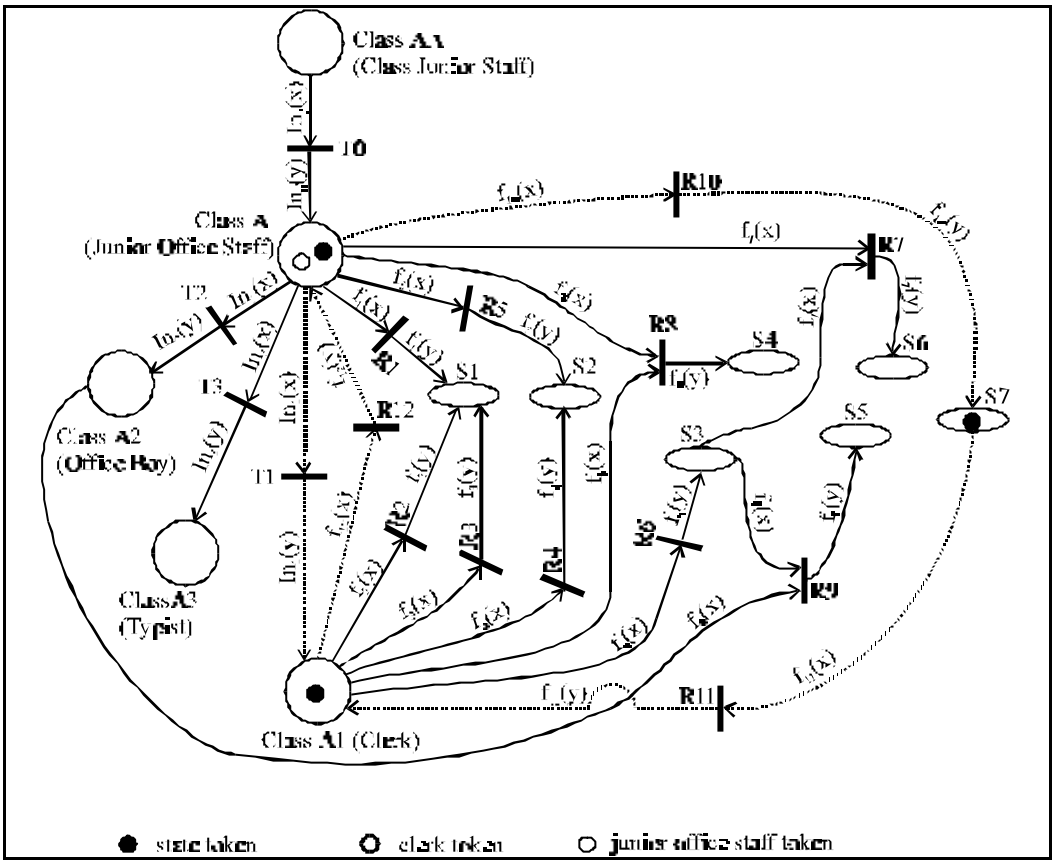


Figure 6.5a. SCCPN representation showing the events of cyclicity.

In our example, Rule 10, Rule 11 and Rule 12 will form such a cyclicity. In Figure 6.5a, 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 (Figure 6.5b) will show that there exists an infinite tree which has the branching pattern repeated after four levels. (Markings M7, M13 and M12 are repeated in cycles)

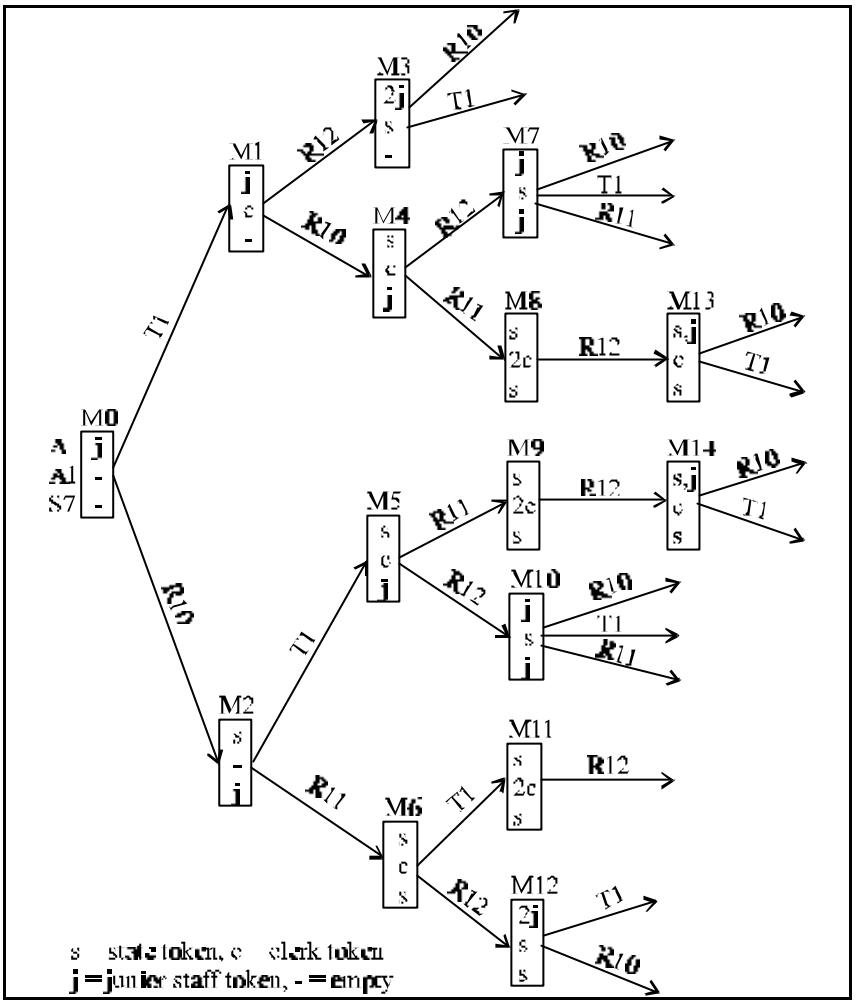


Figure 6.5b. Reachability graph due to the firing of R10, R11 and R12

6.2.1.2. Consistency

6.2.1.2.1 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, Rule X is in conflict with Rule Y. In practical Expert System development, this problem is dealt with by the concepts of overriding (i.e. 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 6.6a, 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 6.6b. 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.

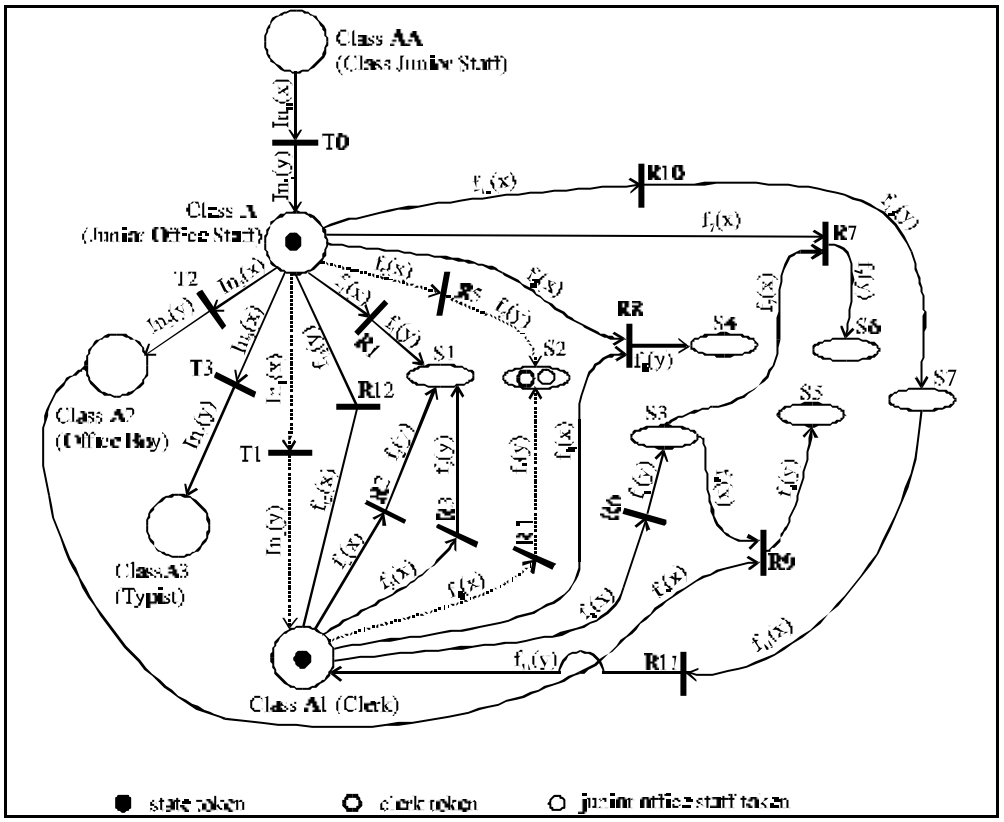


Figure 6.6a. SCCPN representation showing the events of contradiction

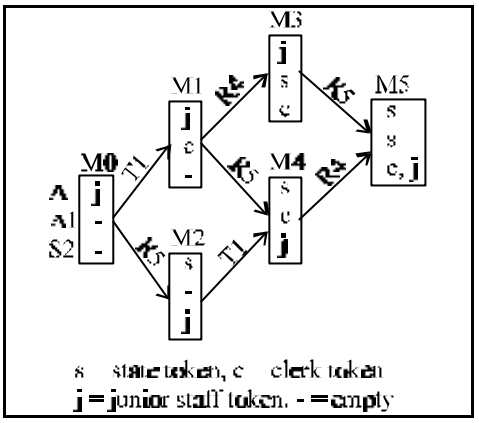


Figure 6.6b. Reachability graph due to the firing of R4 and R5

6.2.1.2.2. 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. E.g. 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 : $A \Rightarrow C$

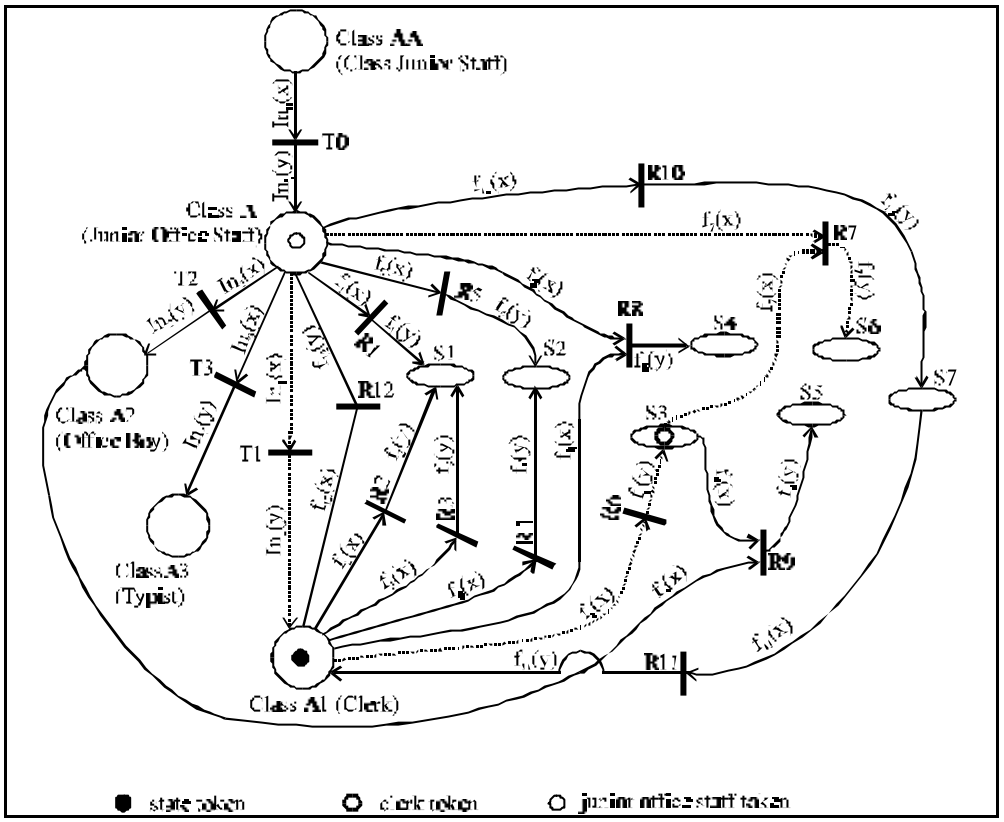


Figure 6.7a. SCCPN representation showing the events of unnecessary IF condition

The second IF condition becomes unnecessary. In our example HES (Figure 6.7a), 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 (i.e. Rule 6 and Rule 7).

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'∧(A∧B)⇒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 (Figure 6.7b) generated by the initial Junior Office Staff token in Class A, we only have three markings which S6 never gets inferred with any token. It 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. (i.e. IF X is a Junior Office Staff AND IF X is a Clerk.)

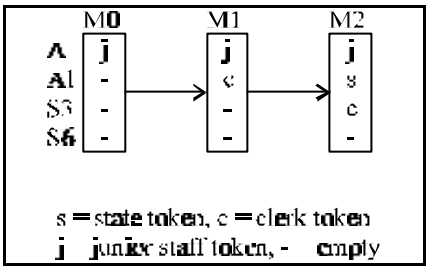


Figure 6.7b. Reachability graph due to the firing of R6 & R7

6.2.1.3 Completeness

6.2.1.3.1 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. E.g.

Rule X : A∧A'⇒C

Rule Y : A1∧A2⇒C

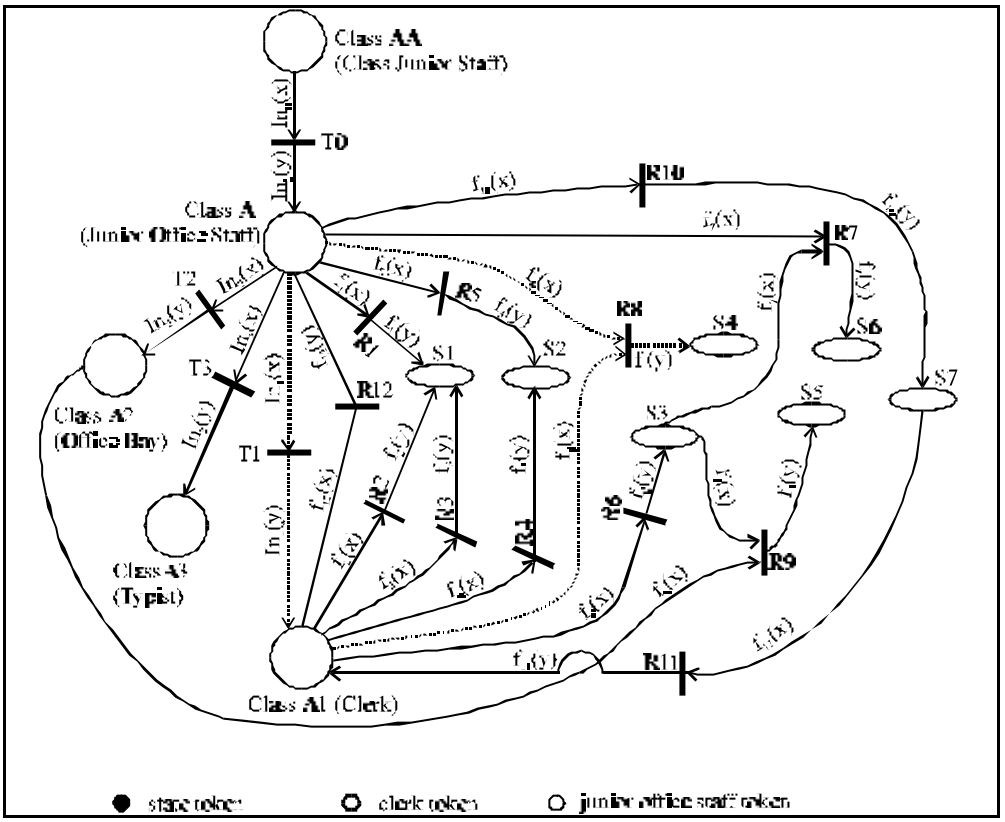


Figure 6.8a. SCCPN representation showing the events of Unreachability

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 6.8.a), Rule 8 is found to be in this situation. Examining the reachability tree (Figure 6.8b), no token is ever deposited in S4 in all reachability Markings from M0.

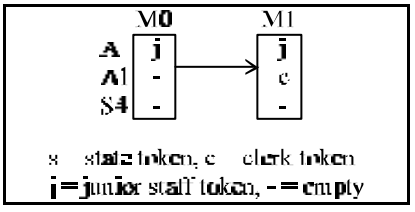


Figure 6.8b. Reachability graph due to the firing of R8

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.

6.2.1.3.2 Unreachability, Case II

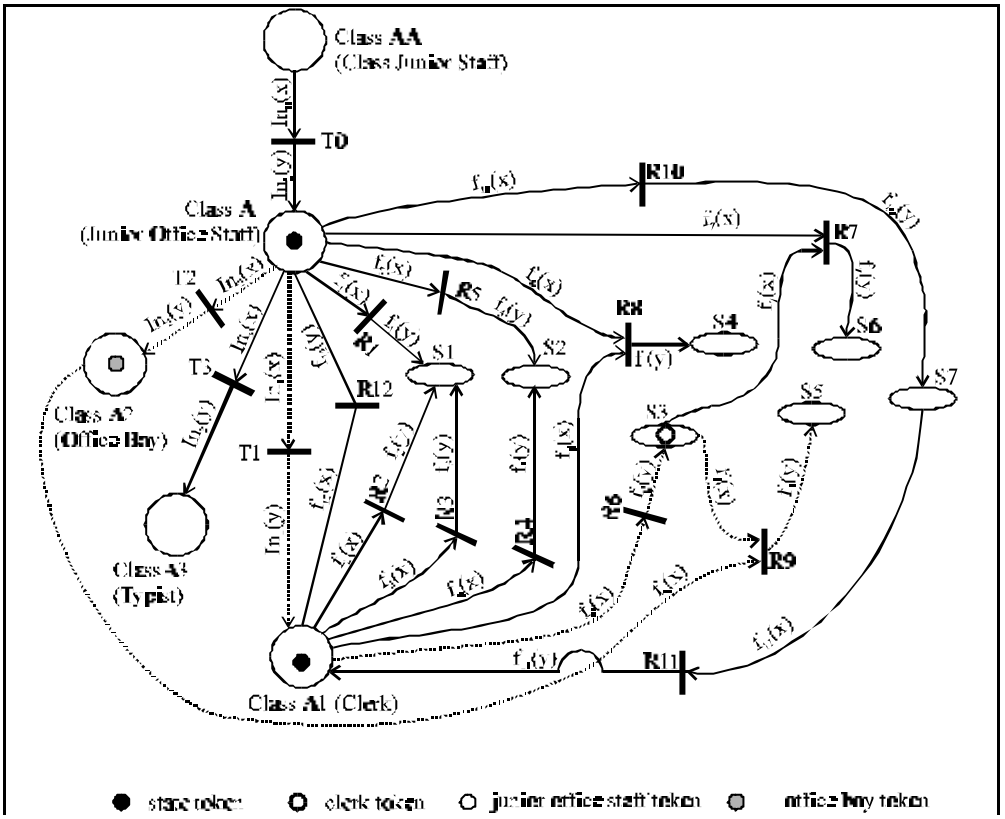


Figure 6.9a. SCCPN representation showing the events of Unreachability

Consider a more complicated situation which involves chain rules (Figure 6.9a), 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 parts are having mutually exclusive class instantiation.

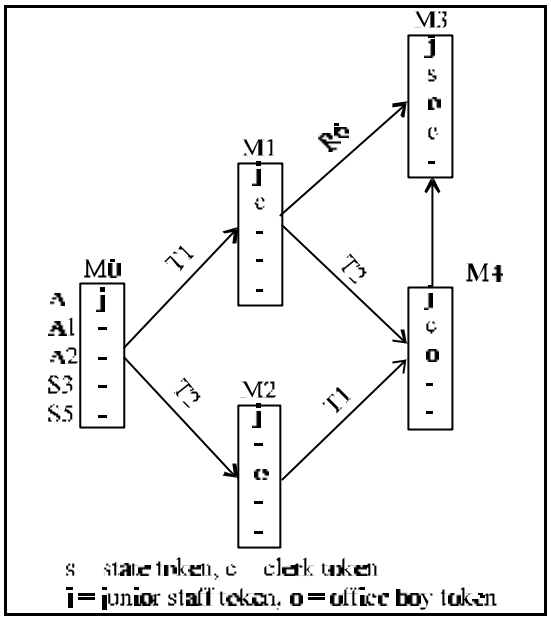


Figure 6.9b. Reachability graph due to the firing of R6 & R9

By examining the reachability graph in Figure 6.9b it shows that S5 never has any token reached from Marking M0. This means this rule is unreachable.

6.3. Time and Space Complexity of the SCCPN Methodology

When the SCCPN methodology is used to detect anomalies of the above Personnel Selection Expert System, some measurements are necessary to assess the real performance of the methodology. Two of the most important considerations are how much memory (space complexity) it will use to construct the full Occurrence Graph and how long (time complexity) it will take to search for a particular marking in the nodes of the Occurrence Graph? Other important issues include the effort used to transform the rules and object hierarchy into places and transactions and the effects of choosing different search strategy.

6.3.1. Derivation of the Occurrence Graphs

Given a SCCPN, the derivation of the Occurrence Graph depends on a number of parameters, such as the number of places and transactions, the arc expressions, the number of token types and the initial markings. Refer to the Personnel Selection Expert System, as can be seen from Figure 6.3 to Figure 6.9, the size and the shape of the Occurrence Graph only depend on the initial markings of the SCCPN. (i.e. the characteristics of the Junior Office Staff). It is because there are no changes of the production rules and the object hierarchy used.

The derivation of the Occurrence Graphs of the Personnel Selection Expert System can be divided into the following two steps: (1) Calculate the total number of Occurrence Graphs generated from all possible initial markings; (2) For each initial marking, calculate the efforts required to derive that particular Occurrence Graph. Therefore, using the worst-case analysis, the total number of Occurrence Graphs generated in this example is equal to the total number of possible combination of slot values between the Slot Knowledge of Work and Leading Skills inclusively.

= (Possible values each slot could have) to the power of (total number of slots)
= 39
= 19,683

Therefore, there are totally 19,683 different Junior Office Staff tokens for initial markings, which corresponds to 19,683 Occurrence Graphs being generated. In order to reduce the number of Occurrence Graphs being examined, we can use only those meaningful Junior Office Staff tokens. (e.g. those with at least six "GOOD"s between the Slot Knowledge of Work and Leading Skills. This reduced the number of initial markings to:

= (six "Good") + (seven "Good") + (Eight "Good") + (Nine "Good")
= 23*C(9,6) + 22*C(9,7) + 2*C(9,8) + 1
= 672 + 144 + 18 + 1
= 835

Therefore, using the above meaningful tokens as initial markings, we can reduce over 95.7% of our efforts in generating and examining the Occurrence Graphs.

Secondly, the efforts required to derive the Occurrence Graph of the Personnel Selection Expert System, using the worst-case analysis, is as follows:

We convert the SCCPN in Figure 6.2. into two matrices, Di, and Do, which are used to represent the input and output functions for the class tokens and state tokens respectively.

The Di of the Personnel Selection Expert System SCCPN is:

	T0	T1	T2	T3	R1	R2	R3	R4
ClassAA	-	-	-	-	-	-	-	-
ClassA	[j,s]	-	-	-	-	-	-	-
ClassA1	-	[c,s]	-	-	-	-	-	-
ClassA2	-	-	[o,s]	-	-	-	-	-
ClassA3	-	-	-	[t,s]	-	-	-	-
S1	-	-	-	-	[j,s]	[c,s]	[c,s]	-
S2	-	-	-	-	-	-	-	[c,s]
S3	-	-	-	-	-	-	-	-
S4	-	-	-	-	-	-	-	-
S5	-	-	-	-	-	-	-	-
S6	-	-	-	-	-	-	-	-
S7	-	-	-	-	-	-	-	-

Table 6.3a. Input functions for the class tokens and control tokens (T0-R4)

f = Junior Staff j = Junior Office Staff c = Clerk o = Office Boy
t = Typist

	R5	R6	R7	R8	R9	R10	R11	R12
ClassAA	-	-	-	-	-	-	-	-
ClassA	-	-	-	-	-	-	-	[c,s]
ClassA1	-	-	-	-	-	-	[j,s]	-
ClassA2	-	-	-	-	-	-	-	-
ClassA3	-	-	-	-	-	-	-	-
S1	-	-	-	-	-	-	-	-
S2	[j,s]	-	-	-	-	-	-	-
S3	-	[c,s]	-	-	-	-	-	-
S4	-	-	-	[j/c,s]	-	-	-	-
S5	-	-	-	-	[c/o,s]	-	-	-
S6	-	-	[j/c,s]	-	-	-	-	-
S7	-	-	-	-	-	[j,s]	-	-

Table 6.3b. Input functions for the class tokens and control tokens (R5-R12)

Detection of any form of error in the Personnel Selection Expert System will require the generation of a reachability tree for close examination. All markings that are reachable from a given marking will need to be stored for examination. Given an initial marking M0, the effort, in the worst case, to derive the next marking will involve the following operations:

Identify enabled transitions requires comparison between M0 and the Markings in Di.

= (Number of tokens compared) * (Number of slots in each token)
= (16 class tokens) * (16 slots) + (16 state tokens) * (2 slots)
= 256+32
= 288 comparisons

Similarly, the creation of the next state marking requires substitutions of token colours with the values in the output matrix Do.

The Do of the Personnel Selection Expert System SCCPN is:

	T0	T1	T2	T3	R1	R2	R3	R4
ClassAA	[f,s]	-	-	-	-	-	-	-
ClassA	-	[j,s]	[j,s]	[j,s]	[j,s]	-	-	-
ClassA1	-	-	-	-	-	[c,s]	[c,s]	[c,s]
ClassA2	-	-	-	-	-	-	-	-
ClassA3	-	-	-	-	-	-	-	-
S1	-	-	-	-	-	-	-	-
S2	-	-	-	-	-	-	-	-
S3	-	-	-	-	-	-	-	-
S4	-	-	-	-	-	-	-	-
S5	-	-	-	-	-	-	-	-
S6	-	-	-	-	-	-	-	-
S7	-	-	-	-	-	-	-	-

Table 6.4a. Output functions for the class tokens and control tokens (T0-R4)

f = Junior Staff j = Junior Office Staff c = Clerk o = Office Boy

t = Typist

	R5	R6	R7	R8	R9	R10	R11	R12
ClassAA	-	-	-	-	-	-	-	-
ClassA	[j,s]	-	[j,s]	[j,s]	-	[j,s]	-	-
ClassA1	-	[c,s]	-	[c,s]	-	-	-	[c,s]
ClassA2	-	-	-	-	[o,s]	-	-	-
ClassA3	-	-	-	-	-	-	-	-
S1	-	-	-	-	-	-	-	-
S2	-	-	-	-	-	-	-	-
S3	-	-	[c,s]	-	[c,s]	-	-	-
S4	-	-	-	-	-	-	-	-
S5	-	-	-	-	-	-	-	-
S6	-	-	-	-	-	-	-	-
S7	-	-	-	-	-	-	[j,s]	-

Table 6.4b. Output functions for the class tokens and control tokens (R5-R12)

Therefore, the number of substitutions required is:

= (Number of tokens substituted) * (Number of slots in each token)

= (19 class tokens) * (16 slots) + (19 state tokens) * (2 slots)

= 304 + 38

= 342 substitutions

The total number of comparisons and substitutions required in each derivation of the next state markings are 342 + 288 = 630 in the worst-case.

6.3.2. Transformation of Rules and Object hierarchy to SCCPN

Since there are 12 rules having a total of 26 conditions and 12 actions, the maximum number of predicate places is (26+12) = 38 (storage spaces). There are 5 object classes in the hierarchy, therefore, we need another 5 object places. The total number of storage spaces for the predicate and class places of this Personnel Selection System are 43.

The storage spaces required for the tokens are calculated as follows:

= (Number of classes) * (Number of Places) * (Number of slots in each token)
= 5 * 43 * 16
= 3,440 (storage spaces).

Therefore the total storage spaces required are 3,483.

6.3.3. Evaluation function for particular marking

After generation of the Occurrence Graph, each node will be evaluated by a function. The purpose of this function is to check for the existence of a particular marking within a node. For a function that searches for Subsumption within the HES, it is as follows:

```
SearchSubsumption (SearchArea, StartNode)
Begin
    Result:=StartNode; Found:=FALSE
        For all nodes∈ SearchArea Do
        Begin
                If ParentToken and ChildToken Exists in Result THEN
                Begin
                Compare Slots Value
                IF Parent.Slots.Value = Child.Slots.Value THEN Found:=TRUE
                End
        End
End.
```

SearchArea specifies the part of the Occurrence Graph that should be searched. It is often the subset that is minimally enabled by the meaningful initial markings.

6.3.4. Complexity measures for the Personnel Selection Expert System SCCPN

6.3.4.1. Correctness

6.3.4.1.1. Subsumption

Refer to the Occurrence Graph for Subsumption Case I, Figure 6.3b, the space required is:

= (Total no. of nodes)*(Storage spaces for each node)

= 6*3,483

= 20,898

The computation effort required for comparisons and substitutions is:

= (Total no. of comparisons and substitutions) * (Steps in comparison + substitution)

= 7*630

= 4,410

Refer to the Occurrence Graph for Subsumption Case II, Figure 6.4b, the space required is:

= (Total no. of nodes)*(Storage spaces for each node)

= 6*3,483

= 20,898

The computation effort required for comparisons and substitutions is:

= (Total no. of comparisons and substitutions) * (Steps in comparison + substitution)

= 9*630

= 5,670

6.3.4.1.2. Cyclicity

Refer to the Occurrence Graph for Cyclicity, Figure 6.5b, the space required is:

= (Total no. of nodes)*(Storage spaces for each node)

= 15*3,483

= 52,245

The computation effort required for comparisons and substitutions is:

= (Total no. of comparisons and substitutions) * (Steps in comparison + substitution)

= 29*630

= 18,270

6.3.4.2. Consistency

6.3.4.2.1. Contradiction

Refer to the Occurrence Graph for Contradiction, Figure 6.6b, the space required is:

= (Total no. of nodes)*(Storage spaces for each node)

= 6*3,483

= 20,898

The computation effort required for comparisons and substitutions is:
= (Total no. of comparisons and substitutions) * (Steps in comparison + substitution)
= 7*630
= 4,410

6.3.4.2.2. Unnecessary IF Condition

Refer to the Occurrence Graph for Unnecessary IF Condition, Figure 6.7b, the space required is:
= (Total no. of nodes)*(Storage spaces for each node)
= 3*3,483
= 10,449

The computation effort required for comparisons and substitutions is:
= (Total no. of comparisons and substitutions) * (Steps in comparison + substitution)
= 2*630
= 1,260

6.3.4.3. Completeness

6.3.4.3.1. Unreachability

Refer to the Occurrence Graph for Unreachability Case I, Figure 6.8b, the space required is:
= (Total no. of nodes)*(Storage spaces for each node)
= 2*3,483
= 6,966

The computation effort required for comparisons and substitutions is:
= (Total no. of comparisons and substitutions) * (Steps in comparison + substitution)

= 1*630

= 630

Refer to the Occurrence Graph for Unreachability Case II, Figure 6.9b, the space required is:

= (Total no. of nodes)*(Storage spaces for each node)

= 5*3,483

= 17,415

The computation effort required for comparisons and substitutions is:

= (Total no. of comparisons and substitutions) * (Steps in comparison + substitution)

= 6*630

= 3,780

From the above calculations, the total effort involved in finding the anomalies of the Personnel Selection Expert System requires 149,769 storage spaces and 38,430 computations.

6.3.5. Worst Case Analysis of Occurrence Graphs

The above space and time complexity analysis is based on the concept of worst-case analysis. As can be seen from the above calculations, the amount of work done cannot be described by a single number because the number of steps performed is not the same for all inputs. According to (Baase, S., 1988), worst case analysis of an algorithm is defined as:

$$W(n) = \max \{t(I) \mid I \in Dn\}$$

where W(n) is the maximum number of basic operations performed by the algorithm on any input of size n. Dn is the set of inputs of size n for the problem under

consideration, and I be an element of Dn. t(I) is the number of basic operations performed by the algorithm on input I.

Refer to the Occurrence Graphs of the Personnel Selection Expert System, Dn is the sets of meaningful Junior Office Staff tokens, I is a particular Junior Office Staff token, t(I) is the number of comparison and substitutions required on input I and W(n) is the maximum computations generated over the input set I. (i.e. W(n) = the detection of Cyclicity which requires 52,245 storage spaces and 18,270 computations.)

In practical applications of Occurrence Graphs analysis, as reported by (Jensen, K., 1995,1997), the time and space complexity could be significantly reduced since a lot of the markings in an Occurrence Graph will be almost identical. The solution is to avoid duplication of identical parts by representing each marking as a set of pointers, as shown in Figure 6.10. This means each multi-set only appears once - even though it may appear in many different marking nodes.

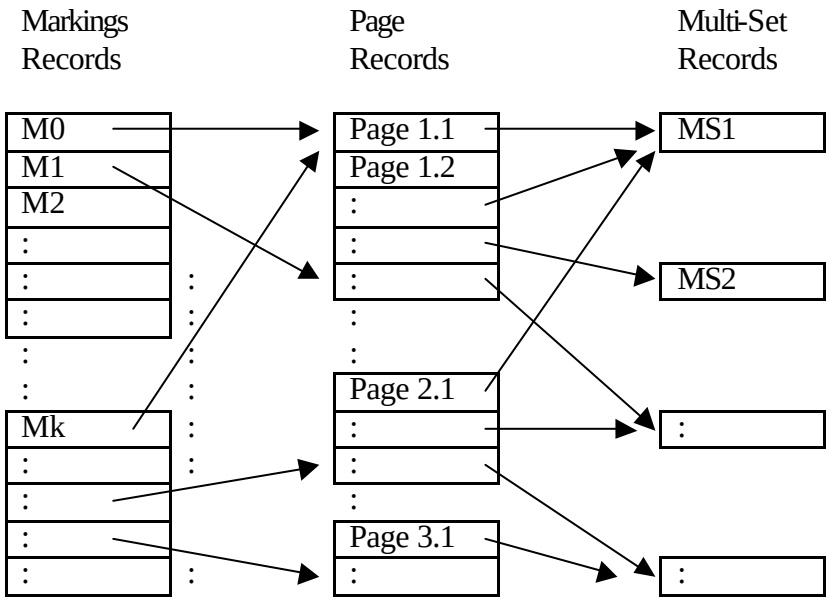


Figure 6.10. Representation of a set of markings

The above method saves a lot of space, as well as the time required to evaluate a particular marking. Nevertheless, the full exploration of the time and space reduction methodologies of Occurrence Graph analysis is beyond the scope of this research, therefore, our current analysis only concentrated on the efforts required for building the full Occurrence Graphs of the Personnel Selection Expert System for anomalies detection.

6.4. Summary

In this Chapter, we have applied our SCCPN approach to model a practical Personnel Selection Hybrid Rule- and Frame-based Expert Systems. The detection and analysis of the anomalies of system 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. A complexity analysis is conducted to investigate the performance of the methodology. The complexity includes the effort to transform the rules and object hierarchy into places and transactions, the calculation of the size of the Occurrence Graphs, and the time required searching such Occurrence Graphs for anomalies. Our approach allows for formal verification of the correctness, consistency, and completeness of the hybrid knowledge base.

CHAPTER 7. FORMAL DESCRIPTION AND VERIFICATION OF RULE/FRAME-BASED HES

It has been shown that Hybrid Expert Systems can be modelled by State Controlled Coloured Petri Nets (SCCPNs). Consequently, we have been able to dynamically simulate the propagation of rule inference and inherence of object properties in the hybrid knowledge base and identify some defined anomalies through the analysis of the reachability tree. To allow for accurate detection of these anomalies, a more formal definition and discussion of the properties of these anomalies will be given. Altogether, eight Propositions are derived, each Proposition represents a dynamic property of the SCCPN, namely: (1) Redundancy, (2) Subsumption, (3) Ambiguity, (4) Cyclicity, (5a) Contradiction I, (5b) Contradiction II, (6) Deadend, (7) Unnecessary IF and (8) Unreachability. A set of Occurrence Graph (*c.f.* Definition 5.11.) properties is defined for each Proposition. These properties act as the necessary and sufficient conditions for the existence of the corresponding dynamic properties in the SCCPN.

Therefore, if we want to formally verify whether a given Hybrid Expert System consists of, for example, Redundancy or not, we only have to investigate the HES's corresponding SCCPN, and thus we shall consider those Occurrence Graph properties under the Redundancy Proposition. If those properties exist, we can deduce that the SCCPN consists of Redundancy. Since the SCCPN is the model of the given Hybrid Expert System, therefore, we have verified the HES.

7.1. Correctness: Forward Case Proof

The problems of correctness about a rule set applied to an object hierarchy might involve redundancy, subsumption, ambiguity, and cyclicity as described in terms of predicate formulae in Chapter 4. These are observable either between a pair of rules

applied to an object hierarchy or rules that represent chains of inference in the object hierarchy.

7.1.1. Redundancy

Proposition 7.1. 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 predicate k of this two class tokens are the same.

If there exists incorrect rules applied to the object hierarchy of the following cases:

Case (I): Conditions and Actions identical between Parent Class and Child Class.

Let $E(\Phi)$ be the arc expression function of the predicate IS-A member of Parent Class and
Let $E(\phi)$ be the arc expression function of the predicate IS-A member of Child Class.

In SCCPN representation, there should exist t_{r0}, t_{r1} and t_c such that

$E(\tilde{O}_c(t_{r0}),t_{r0}) - E(\Phi)=E(\tilde{O}_c(t_{r1}),t_{r1}) - E(\phi)$
$E(t_{r0},\ddot{O}_c(t_{r0}))=E(t_{r1},\ddot{O}_c(t_{r1}))$

$\tilde{O}_c(t_{r0})=\tilde{A}_c(t_c)$, $\ddot{A}_c(t_c)=\tilde{O}_c(t_{r1})$, $\ddot{O}_c(t_{r0})=\ddot{O}_c(t_{r1})$,
$\tilde{O}_s(t_{r0})=\tilde{A}_s(t_c)$, $\ddot{A}_s(t_c)=\tilde{O}_s(t_{r1})$, $\ddot{O}_s(t_{r0})=\ddot{O}_s(t_{r1})$

Choose M_0 with a class token element (p_{r0},c_{c0}) and a state token (p_{r0},c_{s0}) s.t. t_{r0} is minimally enabled,

then $M_{sk}=1$ if $p_{sk}\in\tilde{O}_s(t_{r0})$, 0 otherwise.
And $M_{ck}=1$ if $p_{ck}\in\tilde{O}_c(t_{r0})$, 0 otherwise.

Since $\tilde{O}_s(t_{r0})=\tilde{A}_s(t_c)$ and $\tilde{O}_c(t_{r0})=\tilde{A}_c(t_c)$, t_c is enabled (Definition 5.7.).

Since t_c is enabled, the new marking in $\ddot{A}_c(t_c)=1$ and has a colour of (p_{r1},c_{c1}) which is inherited from (p_{r0},c_{c0}). Where $E(p_{r0},t_c) - E(\Phi)=E(t_c,p_{r1}) - E(\phi)$ (Definition 5.8.).
Since $E(\tilde{O}_c(t_{r1}),t_{r1})=E(\tilde{O}_c(t_{r0}),t_{r0}) - E(\Phi) + E(\phi)$, therefore t_{r1} is enabled.

As from Definition 5.10, $\exists M'$, $\exists M''$ s.t. $M'=\delta(M_0,\sigma_i)$, $M''=\delta(M_0,\sigma_j)$ and $\sigma_i=(t_{r0})$, $\sigma_j=(t_c,t_{r1})$.

Therefore

$$M'_{sk} = \begin{cases} 1 & if\ p_{sk} \in \{\tilde{O}_s(t_{r0}), \ddot{O}_s(t_{r0})\} \\ 0 & otherwise \end{cases}$$

$$M'_{ck} = \begin{cases} 1 & if\ p_{ck} \in \ddot{O}_c(t_{r0}) \\ 0 & otherwise \end{cases}$$

And the colour of the class token at $\ddot{O}_c(t_{r0})=(p_{rk},c_{ck})'$

$$M''_{sk} = \begin{cases} 1 & if\ p_{sk} \in \{\tilde{O}_s(t_{r1}), \ddot{O}_s(t_{r1})\} \\ 0 & otherwise \end{cases}$$

$$M''_{ck} = \begin{cases} 1 & if\ p_{ck} \in \ddot{O}_c(t_{r1}) \\ 0 & otherwise \end{cases}$$

And the colour of the class token at $\ddot{O}_c(t_{r1})=(p_{rk},c_{ck})''$

Since $E(t_{r0},\ddot{O}_c(t_{r0}))=E(t_{r1},\ddot{O}_c(t_{r1}))$, therefore $(p_{rk},c_{ck})'=(p_{rk},c_{ck})''$

Thus, for $p_{sk} \in \ddot{O}_s(t_{r0})$, $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, and for $p_{ck} \in \ddot{O}_c(t_{r0})$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, and $(p_{rk},c_{ck})'=(p_{rk},c_{ck})''$, implying incorrectness with $\sigma_i=(t_{r0})$, $\sigma_j=(t_c,t_{r1})$.

Case (II): Chained inference

Let $E(\Phi)$ be the arc expression function of the predicate IS-A member of Parent Class and

Let $E(\phi)$ be the arc expression function of the predicate IS-A member of Child Class.

In SCCPN representation, there should exists t_{r0}, and $\sigma_i=(t_c,t_{r1},t_{r2},\ldots.t_{rj})$ such that

$E(\tilde{O}_c(t_{r0}),t_{r0}) - E(\Phi)=E(\tilde{O}_c(t_{r1}),t_{r1}) - E(\phi)$
$E(t_{r0},\ddot{O}_c(t_{r0}))=E(t_{rj},\ddot{O}_c(t_{rj}))$

$\ddot{O}_s(t_{r(m)})=\tilde{O}_s(t_{r(m+1)})$ for m=1,2,…..j-1.
$\ddot{O}_c(t_{r(m)})=\tilde{O}_c(t_{r(m+1)})$ for m=1,2,…..j-1.
$\tilde{O}_c(t_{r0})=\tilde{A}_c(t_c)$, $\ddot{A}_c(t_c)=\tilde{O}_c(t_{r1})$,
$\tilde{O}_s(t_{r0})=\tilde{A}_s(t_c)$, $\ddot{A}_s(t_c)=\tilde{O}_s(t_{r1})$,

Choose M_0 with a class token element (p_{r0},c_{c0}) and a state token (p_{r0},c_{s0}) s.t. $\sigma_i=(t_c,t_{r1},t_{r2},\ldots.t_{rj})$ is minimally enabled, i.e., $\forall$m=1,2,3,….j-1,

$$\text{then } M_{sk}=\begin{cases}1 & \text{if } p_{sk}\in \tilde{A}_s(t_c)\\ 0 & \text{otherwise}\end{cases}$$

$$\text{And } M_{ck}=\begin{cases}1 & \text{if } p_{ck}\in \tilde{A}_c(t_c)\\ 0 & \text{otherwise}\end{cases}$$

The execution of transition sequence, σ_i, gives M' s.t. $\forall$m=1,2,3,….j, $\ddot{O}_s(t_m)\in\ddot{O}_s(\sigma_i)$

$$M'_{sk}=\begin{cases}1 & \text{if } p_{sk}\in\{\tilde{O}_s(t_1),\ddot{O}_s(s_i)\}\\ 0 & \text{otherwise}\end{cases}$$

And the colour of the class token at $\ddot{O}_c(t_j)=(p_{rk},c_{ck})$'

Since $E(\tilde{O}_c(t_{r0}),t_{r0})=E(\tilde{O}_c(t_{r1}),t_{r1}) - E(\phi) + E(\Phi)$, therefore t_{r0} is enabled.

Let $M''_{ck}=\delta(M_0,t_{r0})$,

$$M''_{ck} = \begin{cases} 1 & if\ p_{ck} \in Ö_c(t_{r0}) \\ 0 & otherwise \end{cases}$$

And the colour of the class token at $Ö_c(t_{r0})=(p_{rk},c_{ck})''$

Since $E(t_{r0},Ö_c(t_{r0}))=E(t_{rj},Ö_c(t_{rj}))$, therefore $(p_{rk},c_{ck})'=(p_{rk},c_{ck})''$

Since $Ö_c(t_{r0})=Ö_c(t_{rj})\subset Ö_c(\sigma_i)$, thus, for $p_{sk}\in Ö_s(t_{rj})$, $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, and for $p_{ck}\in Ö_c(t_{rj})$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, and $(p_{rk},c_{ck})'=(p_{rk},c_{ck})''$, implying incorrectness with $\sigma_i=(t_c,t_{r1},t_{r2},\ldots.t_{rj})$, $\sigma_j=(t_{r0})$.

7.1.2. Subsumption

Proposition 7.2. 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) $(p_{rk},c_{ck})''\subseteq(p_{rk},c_{ck})'$

Case (I): Rule X is subsumed by Rule Y (condition part) between Parent Class and Child Class

Let E(Φ) be the arc expression function of the predicate IS-A member of Parent Class and

Let E(ϕ) be the arc expression function of the predicate IS-A member of Child Class.

In SCCPN representation, there should exist t_{r0}, t_{r1} and t_c such that

$E(\tilde{O}_c(t_{r1}),t_{r1}) - E(\phi) \subset E(\tilde{O}_c(t_{r0}),t_{r0}) - E(\Phi)$

$E(t_{r0},\ddot{O}_c(t_{r0})) = E(t_{r1},\ddot{O}_c(t_{r1}))$

$\tilde{O}_c(t_{r0}) = \tilde{A}_c(t_c)$, $\ddot{A}_c(t_c) = \tilde{O}_c(t_{r1})$, $\ddot{O}_c(t_{r0}) = \ddot{O}_c(t_{r1})$,

$\tilde{O}_s(t_{r0}) = \tilde{A}_s(t_c)$, $\ddot{A}_s(t_c) = \tilde{O}_s(t_{r1})$, $\ddot{O}_s(t_{r0}) = \ddot{O}_s(t_{r1})$

Choose M_0 with a class token element (p_{r0},c_{c0}) and a state token (p_{r0},c_{s0}) s.t. t_{r0} is minimally enabled,

then $M_{sk}=1$ if $p_{sk} \in \tilde{O}_s(t_{r0})$, 0 otherwise.

And $M_{ck}=1$ if $p_{ck} \in \tilde{O}_c(t_{r0})$, 0 otherwise.

Since $\tilde{O}_s(t_{r0}) = \tilde{A}_s(t_c)$ and $\tilde{O}_c(t_{r0}) = \tilde{A}_c(t_c)$, t_c is enabled (Definition 5.7.).

Since t_c is enabled, the new marking in $\ddot{A}_c(t_c)=1$ and has a colour of (p_{r1},c_{c1}) which is inherited from (p_{r0},c_{c0}). Where $E(p_{r0},t_c) - E(\Phi) = E(t_c,p_{r1}) - E(\phi)$ (Definition 5.8.).

Since $E(\tilde{O}_c(t_{r1}),t_{r1}) = E(\tilde{O}_c(t_{r0}),t_{r0}) - E(\Phi) + E(\phi)$, therefore t_{r1} is enabled.

As from Definition 5.10, $\exists M'$, $\exists M''$ s.t. $M'=\delta(M_0,\sigma_i)$, $M''=\delta(M_0,\sigma_j)$ and $\sigma_i=(t_{r0})$, $\sigma_j=(t_c,t_{r1})$.

Therefore

$$M'_{sk} = \begin{cases} 1 & \text{if } p_{sk} \in \{\tilde{O}_s(t_{r0}),\ddot{O}_s(t_{r0})\} \\ 0 & \text{otherwise} \end{cases}$$

$$M'_{ck} = \begin{cases} 1 & if\ p_{ck} \in Ö_c(t_{r0}) \\ 0 & otherwise \end{cases}$$

And the colour of the class token at $Ö_c(t_{r0})=(p_{rk},c_{ck})'$

$$M''_{sk} = \begin{cases} 1 & if\ p_{sk} \in \{Õ_s(t_{r1}), Ö_s(t_{r1})\} \\ 0 & otherwise \end{cases}$$

$$M''_{ck} = \begin{cases} 1 & if\ p_{ck} \in Ö_c(t_{r1}) \\ 0 & otherwise \end{cases}$$

And the colour of the class token at $Ö_c(t_{r1})=(p_{rk},c_{ck})''$

Since $E(t_{r0},Ö_c(t_{r0}))=E(t_{r1},Ö_c(t_{r1}))$, therefore $(p_{rk},c_{ck})'=(p_{rk},c_{ck})''$

Thus, for $p_{sk} \in Ö_s(t_{r0})$, $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, and for $p_{ck} \in Ö_c(t_{r0})$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, and $(p_{rk},c_{ck})'=(p_{rk},c_{ck})''$, implying incorrectness with $\sigma_i=(t_{r0})$, $\sigma_j=(t_c,t_{r1})$.

Case (II): Rule X is subsumed by Rule Y (action part) between Parent Class and Child Class.

Let $E(\Phi)$ be the arc expression function of the predicate IS-A member of Parent Class and

Let $E(\phi)$ be the arc expression function of the predicate IS-A member of Child Class.

In SCCPN representation, there should exists t_{r0}, t_{r1} and t_c such that

$E(Õ_c(t_{r0}),t_{r0}) - E(\Phi)=E(Õ_c(t_{r1}),t_{r1}) - E(\phi)$

$E(t_{r1},Ö_c(t_{r1})) \subset E(t_{r0},Ö_c(t_{r0}))$

$\tilde{O}_c(t_{r0})=\tilde{A}_c(t_c)$, $\ddot{A}_c(t_c)=\tilde{O}_c(t_{r1})$, $\ddot{O}_c(t_{r0})=\ddot{O}_c(t_{r1})$,
$\tilde{O}_s(t_{r0})=\tilde{A}_s(t_c)$, $\ddot{A}_s(t_c)=\tilde{O}_s(t_{r1})$, $\ddot{O}_s(t_{r0})=\ddot{O}_s(t_{r1})$

Choose M_0 with a class token element (p_{r0},c_{c0}) and a state token (p_{r0},c_{s0}) s.t. t_{r0} is minimally enabled,
then $M_{sk}=1$ if $p_{sk}\in\tilde{O}_s(t_{r0})$, 0 otherwise.
And $M_{ck}=1$ if $p_{ck}\in\tilde{O}_c(t_{r0})$, 0 otherwise.

Since $\tilde{O}_s(t_{r0})=\tilde{A}_s(t_c)$ and $\tilde{O}_c(t_{r0})=\tilde{A}_c(t_c)$, t_c is enabled (Definition 5.7.).
Since t_c is enabled, the new marking in $\ddot{A}_c(t_c)=1$ and has a colour of (p_{r1},c_{c1}) which is inherited from (p_{r0},c_{c0}). Where $E(p_{r0},t_c) - E(\Phi)=E(t_c,p_{r1}) - E(\phi)$ (Definition 5.8.).
Since $E(\tilde{O}_c(t_{r1}),t_{r1})=E(\tilde{O}_c(t_{r0}),t_{r0}) - E(\Phi) + E(\phi)$, therefore t_{r1} is enabled.

As from Definition 5.10, $\exists M'$, $\exists M''$ s.t. $M'=\delta(M_0,\sigma_i)$, $M''=\delta(M_0,\sigma_j)$ and $\sigma_i=(t_{r0})$, $\sigma_j=(t_c,t_{r1})$.

Therefore

$$M'_{sk}=\begin{cases}1 & if\ p_{sk}\in\{\tilde{O}_s(t_{r0}),\ddot{O}_s(t_{r0})\}\\ 0 & otherwise\end{cases}$$

$$M'_{ck}=\begin{cases}1 & if\ p_{ck}\in\ddot{O}_c(t_{r0})\\ 0 & otherwise\end{cases}$$

And the colour of the class token at $\ddot{O}_c(t_{r0})=(p_{rk},c_{ck})'$

$$M''_{sk}=\begin{cases}1 & if\ p_{sk}\in\{\tilde{O}_s(t_{r1}),\ddot{O}_s(t_{r1})\}\\ 0 & otherwise\end{cases}$$

$$M''_{ck} = \begin{cases} 1 & \text{if } p_{ck} \in \ddot{O}_c(t_{r1}) \\ 0 & \text{otherwise} \end{cases}$$

And the colour of the class token at $\ddot{O}_c(t_{r1})=(p_{rk},c_{ck})''$

Since $E(t_{r1},\ddot{O}_c(t_{r1}))\subset E(t_{r0},\ddot{O}_c(t_{r0}))$
therefore $(p_{rk},c_{ck})''\subset(p_{rk},c_{ck})'$

Thus, for $p_{sk}\in\ddot{O}_s(t_{r0})$, $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, and for $p_{ck}\in\ddot{O}_c(t_{r0})$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, and $(p_{rk},c_{ck})''\subset(p_{rk},c_{ck})'$, implying incorrectness with $\sigma_i=(t_{r0})$, $\sigma_j=(t_c,t_{r1})$.

Case (III): Rule X is subsumed by Rule Y (condition and action) between Parent Class and Child Class.

Let $E(\Phi)$ be the arc expression function of the predicate IS-A member of Parent Class and
Let $E(\phi)$ be the arc expression function of the predicate IS-A member of Child Class.

In SCCPN representation, there should exists t_{r0}, t_{r1} and t_c such that

$E(\tilde{O}_c(t_{r1}),t_{r1}) - E(\phi)\subset E(\tilde{O}_c(t_{r0}),t_{r0}) - E(\Phi)$
$E(t_{r1},\ddot{O}_c(t_{r1}))\subset E(t_{r0},\ddot{O}_c(t_{r0}))$

$\tilde{O}_c(t_{r0})=\tilde{A}_c(t_c)$, $\ddot{A}_c(t_c)=\tilde{O}_c(t_{r1})$, $\ddot{O}_c(t_{r0})=\ddot{O}_c(t_{r1})$,
$\tilde{O}_s(t_{r0})=\tilde{A}_s(t_c)$, $\ddot{A}_s(t_c)=\tilde{O}_s(t_{r1})$, $\ddot{O}_s(t_{r0})=\ddot{O}_s(t_{r1})$

Choose M_0 with a class token element (p_{r0},c_{c0}) and a state token (p_{r0},c_{s0}) s.t. t_{r0} is minimally enabled,

then $M_{sk}=1$ if $p_{sk}\in\tilde{O}_s(t_{r0})$, 0 otherwise.

And $M_{ck}=1$ if $p_{ck}\in \tilde{O}_c(t_{r0})$, 0 otherwise.

Since $\tilde{O}_s(t_{r0})=\tilde{A}_s(t_c)$ and $\tilde{O}_c(t_{r0})=\tilde{A}_c(t_c)$, t_c is enabled (Definition 5.7.).
Since t_c is enabled, the new marking in $\ddot{A}_c(t_c)=1$ and has a colour of (p_{r1},c_{c1}) which is inherited from (p_{r0},c_{c0}). Where $E(p_{r0},t_c) - E(\Phi)=E(t_c,p_{r1}) - E(\phi)$ (Definition 5.8.).
Since $E(\tilde{O}_c(t_{r1}),t_{r1})=E(\tilde{O}_c(t_{r0}),t_{r0}) - E(\Phi) + E(\phi)$, therefore t_{r1} is enabled.

As from Definition 5.10, $\exists M'$, $\exists M''$ s.t. $M'=\delta(M_0,\sigma_i)$, $M''=\delta(M_0,\sigma_j)$ and $\sigma_i=(t_{r0})$, $\sigma_j=(t_c,t_{r1})$.

Therefore

$$M'_{sk}=\begin{cases}1 & if\ p_{sk}\in\{\tilde{O}_s(t_{r0}),\ddot{O}_s(t_{r0})\}\\ 0 & otherwise\end{cases}$$

$$M'_{ck}=\begin{cases}1 & if\ p_{ck}\in\ddot{O}_c(t_{r0})\\ 0 & otherwise\end{cases}$$

And the colour of the class token at $\ddot{O}_c(t_{r0})=(p_{rk},c_{ck})'$

$$M''_{sk}=\begin{cases}1 & if\ p_{sk}\in\{\tilde{O}_s(t_{r1}),\ddot{O}_s(t_{r1})\}\\ 0 & otherwise\end{cases}$$

$$M''_{ck}=\begin{cases}1 & if\ p_{ck}\in\ddot{O}_c(t_{r1})\\ 0 & otherwise\end{cases}$$

And the colour of the class token at $\ddot{O}_c(t_{r1})=(p_{rk},c_{ck})''$

Since $E(t_{r1},\ddot{O}_c(t_{r1}))\subset E(t_{r0},\ddot{O}_c(t_{r0}))$
therefore $(p_{rk},c_{ck})''\subset(p_{rk},c_{ck})'$

Thus, for $p_{sk} \in Ö_s(t_{r0})$, $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, and for $p_{ck} \in Ö_c(t_{r0})$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, and $(p_{rk},c_{ck})'' \subset (p_{rk},c_{ck})'$, implying incorrectness with $\sigma_i=(t_{r0})$, $\sigma_j=(t_c,t_{r1})$.

7.1.3. Ambiguity

For the Hybrid Expert System (HES) to be unambiguous, there can be no unambiguous input to the rule set which results in having more than one object classes are being instantiated with subsumed conclusion. Less importantly, there can be no unambiguous input which will trigger a number of object classes that lead to the same conclusion. The problem presented by ambiguity increases as the redundancy and subsumption increase. Therefore, it is useful to consider the sources of the redundancy and subsumption in the HES. The anomalies could be specifically identified which might assemble the effect of having an indeterminate rule in the rule base, that is composed of a set of quasi-separated rules in the form of a number of immediate possible transitions being modeled by SCCPN. For instance, if there exists an ambiguous rule

Rule 1: $P \rightarrow Q$

that involves disjunction of predicates, (i.e. one of the predicates is the IS-A predicate), according to the model transformation, the rule could have been represented by a number of possible immediate transitions in SCCPN, i.e. $\exists \Gamma=\{t_k\}$ for Rule 1 in the form of:

$p_1 \vee p_2 \vee \ldots\ldots p_m \rightarrow q_1 \vee q_2 \vee \ldots\ldots q_n$

where $p_i \in P$ for i=1,2,……m,

$q_j \in Q$ for j=1,2,……n,

s.t. $\tilde{O}_s(\Gamma)=\tilde{O}_s(t_{r0})\cup\tilde{O}_s(t_{r1})\cup\ldots..\cup\tilde{O}_s(t_{rk})$,

$\ddot{O}_s(\Gamma)=\ddot{O}_s(t_{r0})\cup\ddot{O}_s(t_{r1})\cup\ldots\ldots\ddot{O}_s(t_{rk})$.

Consequently, we emphasize on the verification of incorrectness due to a number of redundancy and subsumption, that might also demonstrate the existence of ambiguity in a HES. It may not be our good practice to introduce such anomalies by introducing any of incorrect rules into the knowledge base. However, the importance of the verification, in this context, is to allow for a means of demonstrating the possible inference of an indeterminate rule to the rest of the rules in the HES. In considering with the problems of ambiguity in SCCPN representation, we will put our attention on the reachable effect of transition firings in the set Γ upon a given marking.

Proposition 7.3. 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) $(p_{rk},c_{ck})'=(p_{rk},c_{ck})''$

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

Let $E(\Phi_a)$ be the arc expression function of the predicate IS-A member of Object Class A and

Let $E(\Phi_b)$ be the arc expression function of the predicate IS-A member of Object Class B.

In SCCPN representation, there should exists $\Gamma=\{t_{r0},t_{r1}\}$ such that

$\tilde{O}_s(t_{r0})\cap\tilde{O}_s(t_{r1})=\varnothing$, $\ddot{O}_s(t_{r0})=\ddot{O}_s(t_{r1})=\ddot{O}_s(\Gamma)$,
$\tilde{O}_c(t_{r0})\cap\tilde{O}_c(t_{r1})=\varnothing$, $\ddot{O}_c(t_{r0})=\ddot{O}_c(t_{r1})=\ddot{O}_c(\Gamma)$,

and

$E(\Phi_a)\in E(\tilde{O}_c(t_{r0}),t_{r0})$, $E(\Phi_b)\in E(\tilde{O}_c(t_{r1}),t_{r1})$.

Choose M_0 s.t. t_{r0}, t_{r1} are minimally enabled, therefore, t_{r0}, t_{r1} are active (i.e., $M_{ck}=1$ if $p_{ck}\in\tilde{O}_c(\Gamma)$, 0 otherwise and $M_{sk}=1$ if $p_{sk}\in\tilde{O}_s(\Gamma)$, 0 otherwise).

Without loss of generality, let $M'=\delta(M_0,t_{r0})$. Since $\tilde{O}_c(t_{r0})\cap\tilde{O}_c(t_{r1})=\varnothing$, there is no conflict for t_{r0}, t_{r1} and both transitions will be executed immediately one after the other. Thus

$$M'_{ck}=\begin{cases}1 & if \quad p_{ck}\in\{\tilde{O}_c(t_{r1}),\ddot{O}_c(t_{r0})\}\\ 0 & otherwise\end{cases}$$

and

$$M'_{sk}=\begin{cases}1 & if\ p_{sk}\in\{\tilde{O}_s(t_{r1}),\tilde{O}_s(t_{r0}),\ddot{O}_s(t_{r0})\}\\ 0 & otherwise\end{cases}$$

And the colour of the class token at $\ddot{O}_c(t_{r0})=(p_{rk},c_{ck})'$

As $\ddot{O}_c(t_{r0})=\ddot{O}_c(t_{r1})=\ddot{O}_c(\Gamma)$, Let $M''=\delta(M',t_{r1})$,

$$M''_{ck}=\begin{cases}2 & \text{if } p_{ck}\in \ddot{O}_c(\Gamma)\\ 0 & \text{otherwise}\end{cases}$$

$$M''_{sk}=\begin{cases}2 & \text{if } p_{sk}\in \ddot{O}_s(\Gamma)\\ 1 & \text{if } p_{sk}\in \tilde{O}_s(\Gamma)\\ 0 & \text{otherwise}\end{cases}$$

And the colour of the class token at $\ddot{O}_c(t_{r1})=(p_{rk},c_{ck})''$

Since $\ddot{O}_c(t_{r0})=\ddot{O}_c(t_{r1})=\ddot{O}_c(\Gamma)$, and
$E(t_{r0},\ddot{O}_c(t_{r0}))=E(t_{r1},\ddot{O}_c(t_{r1}))$
therefore $(p_{rk},c_{ck})'=(p_{rk},c_{ck})''$

Thus for $\forall p_{rk}\in \ddot{O}_s(\Gamma)$, $\forall p_{rk}\in \ddot{O}_c(\Gamma)$, $M_{sk}=0$, $M'_{sk}\geq 1$, $M''_{sk}>1$, $M_{ck}=0$, $M'_{ck}\geq 1$, $M''_{ck}>1$ and $(p_{rk},c_{ck})'=(p_{rk},c_{ck})''$ implying incorrectness with $\Gamma=\{t_{r0},t_{r1}\}$

Case (II): Rule with inclusive disjunction of IS-A actions from different Object Classes.

Let $E(\Phi_a)$ be the arc expression function of the predicate IS-A member of Object Class A and
Let $E(\Phi_b)$ be the arc expression function of the predicate IS-A member of Object Class B.

In SCCPN representation, there should exist $\Gamma=\{\sigma_0,\sigma_1,\sigma_2\}$ such that

$\sigma_0=(t_{r1},t_{r2},\ldots\ldots t_{rm})$,

$\sigma_1=(t_{r(m+1)},t_{r(m+2)},\ldots\ldots t_{r(m+n)})$,

$\sigma_2=(t_{r(m+n+1)},t_{r(m+n+2)},\ldots\ldots t_{r(m+n+l)})$,

and there exist $t_{ru}\in\sigma_0$, $t_{rv}\in\sigma_1$, $t_{rw}\in\sigma_2$, such that

$\sigma_0\cap\sigma_1\cap\sigma_2=\varnothing$,
$\tilde{O}_s(t_{ru})=\tilde{O}_s(t_{rv})=\tilde{O}_s(t_{rw})$,
$\tilde{O}_c(t_{ru})=\tilde{O}_c(t_{rv})=\tilde{O}_c(t_{rw})$,
$\ddot{O}_s(t_{ru})\subset\ddot{O}_s(t_{rv})$, $\ddot{O}_s(t_{rw})\subset\ddot{O}_s(t_{rv})$,
$\ddot{O}_c(t_{ru})\subset\ddot{O}_c(t_{rv})$, $\ddot{O}_c(t_{rw})\subset\ddot{O}_c(t_{rv})$.

and

$E(\Phi_a)\in E(t_{ru},\ddot{O}_c(t_{ru}))$, $E(\Phi_b)\in E(t_{rw},\ddot{O}_c(t_{rw}))$,
$E(\Phi_a)\in E(t_{rv},\ddot{O}_c(t_{rv}))$, $E(\Phi_b)\in E(t_{rv},\ddot{O}_c(t_{rv}))$,

Choose M_0 s.t. t_{r1} is minimally enabled, then $M_{sk}=1$ if $p_{sk}\in\tilde{O}_s(t_{r1})$, 0 otherwise and $M_{ck}=1$ if $p_{ck}\in\tilde{O}_c(t_{r1})$, 0 otherwise.

Since $\tilde{O}_s(t_{r1})=\tilde{O}_s(t_{r(m+1)})=\tilde{O}_s(t_{r(m+n+1)})$ and $\tilde{O}_c(t_{r1})=\tilde{O}_c(t_{r(m+1)})=\tilde{O}_c(t_{r(m+n+1)})$, $t_{r(m+1)}$ and $t_{r(m+n+1)}$ are also enabled. The effect of having $\ddot{O}_s(t_{ru})\subset\ddot{O}_s(t_{rv})$, $\ddot{O}_s(t_{rw})\subset\ddot{O}_s(t_{rv})$, $\ddot{O}_c(t_{ru})\subset\ddot{O}_c(t_{rv})$ and $\ddot{O}_c(t_{rw})\subset\ddot{O}_c(t_{rv})$ allows for the identification of subsumption in the representation. We use the ideas of Subsumption (Forward Case Proof *c.f.* 7.1.2.) Case (II), and Subsumption (Converse Case Proof *c.f.* 7.2.2.), and prove its inference between σ_0, σ_1 and σ_2 respectively, thus demonstrating the existence of incorrectness, having the properties of the proposition or vice versa.

7.1.4. Circular Rule Sets

Circular Rule Sets take place as a result of the incorrect rules causing cyclicity between the parent and child object classes. The rules are represented by a series of transitions in SCCPNs, being enabled and fired in sequences. To be able to highlight the interinference of the transitions, the reachability set produced by SCCPN analysis is based on a breadth-first ordering. Given a marking M that enables t_{r0}, if there exists any occurrence of cyclicity, then we can express the reachability set sufficiently deep enough to cover the cyclicity, in terms of a transition sequence Γ, as follows:

$\Gamma=(\sigma_1,\sigma_2,\ldots\ldots,\sigma_i,\sigma_{i+1},\ldots..,\sigma_n,\ldots..,\sigma_m)$

where σ_j is a sequence of alternative transitions spanned immediately by σ_{j-1} in Γ for j=2,3,….m. Note that σ_1 is spanned by t_{r0} initially.

And that there exists a cyclic sequence, $\alpha=[t_{ri},t_{r(i+1)},\ldots.t_{r(n)}]$, in Γ where $t_{ri}\in\sigma_i$ for j=i, i+1,....n<m, forming a path which begins and ends with the same transition, and $T_c\cap\alpha\neq\varnothing$.

Therefore an execution of any transition t_{rk} in Γ where $t_{rk}\in\sigma_k$ for k=1,2,....n, will sufficiently trigger the event of cyclicity in the HES.

We define $M^0_{sk}[\tilde{O}_s(t_{rj})]$ as the marking for any $p_{sk}\in\tilde{O}_s(t_{rj})$, similarly, for $M^0_{ck}[\tilde{O}_c(t_{rj})]$, as the marking for any $p_{ck}\in\tilde{O}_c(t_{rj})$. Also, we define $M^i_{sk}[\tilde{O}_s(t_{rj})]$ as the marking for any $p_{sk}\in\tilde{O}_s(t_{rj})$, and $M^i_{ck}[\tilde{O}_c(t_{rj})]$, as the marking for any $p_{ck}\in\tilde{O}_c(t_{rj})$, after an execution of any transition t_{ri} for i>0 in the sequence.

Proposition 7.4. 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) $M^i \in [M^0> = \{M^0, M^1, M^2, \ldots M^i, .. M^j\}$,

(ii) $M^j = \delta(M^0, \alpha)$ for $j>0$,

(iii) $T_c \cap \alpha \neq \varnothing$;

(iv) $M^i_{sk}[\tilde{O}]>0$, $M^j_{sk}[\tilde{O}]>1$.

Case (I): Self-Reference Rule

In SCCPN representation, there should exist t_{r1} and t_c, forming a connected path such that $\tilde{A}_c(t_c) \subseteq \ddot{O}_c(t_{r1})$, $\ddot{A}_c(t_c) \subseteq \tilde{O}_c(t_{r1})$, $\tilde{A}_s(t_c) \subseteq \ddot{O}_s(t_{r1})$ and $\ddot{A}_s(t_c) \subseteq \tilde{O}_s(t_{r1})$. Choose M^0 s.t. t_{r1} is minimally enabled, therefore

$$M^0_{sk} = \begin{cases} 1 & \text{if } p_{sk} \in \tilde{O}_s(t_{r1}) \\ 0 & \text{otherwise} \end{cases}$$

$$M^0_{ck} = \begin{cases} 1 & \text{if } p_{ck} \in \tilde{O}_c(t_{r1}) \\ 0 & \text{otherwise} \end{cases}$$

i.e. $M^0_{sk}[\tilde{O}_s(t_{r1})]=1$ if $p_{sk} \in \tilde{O}_s(t_{r1})$

Since $\tilde{A}_c(t_c) \subseteq \ddot{O}_c(t_{r1})$ and $\tilde{A}_s(t_c) \subseteq \ddot{O}_s(t_{r1})$, t_c is enabled, therefore the marking in $\ddot{A}_c(t_c)=1$, 0 otherwise and the marking in $\ddot{A}_s(t_c)=1$, 0 otherwise.

Since $\ddot{A}_c(t_c) \subseteq \tilde{O}_c(t_{r1})$, $\ddot{A}_s(t_c) \subseteq \tilde{O}_s(t_{r1})$, t_{r1} is enabled, and $M^1_{sk} = \delta(M^0, t_{r1})$,

$$M^1_{sk} = \begin{cases} 2 & if\ p_{sk} \in \tilde{O}_s(t_{r1}) \\ 1 & if\ p_{sk} \in \ddot{O}_s(t_{r1}) \\ 0 & otherwise \end{cases}$$

$$M^1_{ck} = \begin{cases} 1 & if\ p_{ck} \in \tilde{O}_c(t_{r1}) \\ 0 & otherwise \end{cases}$$

Thus for $p_{sk} \in \tilde{O}_s(t_{r1})$, $M^i_{sk}[\tilde{O}_s] > 0$, $M^j_{sk}[\tilde{O}_s] > 1$, implying incorrectness, with i=1, j=1, and $\alpha=(t_{r1})$

Case (II): Self Reference Chain of Inference

In SCCPN representation, there should exist $\alpha=(t_{r1},t_{r2},....t_{r(l-1)},t_c,t_{rl},...t_{rm})$ forming a connected path such that

$\tilde{O}_s(t_{r(l+1)}) \subseteq \ddot{O}_s(t_{rl})$ for l=1,2,.....m-1,
$\tilde{O}_s(t_{r1}) \subseteq \ddot{O}_s(t_{rm})$.

Choose M^0 s.t. $\alpha=(t_{r1},t_{r2},....t_{r(l-1)},t_c,t_{rl},...t_m)$ is minimally enabled, i.e., $\forall l$=1,2,....m-1,

$$M^0_{sk} = \begin{cases} 1 & if\ p_{sk} \in \tilde{O}_s(t_{r1}), where\ M^0_{ck}(p_{ck}) = 1 \\ 0 & otherwise \end{cases}$$

i.e. $M^0_{sk}[\tilde{O}_s(t_{r1})]=1$ if $p_{sk} \in \tilde{O}_s(t_{r1})$
Since $\tilde{O}_s(t_{r1}) \subseteq \ddot{O}_s(t_{rm})$, and $M^m = \delta(M^0,\alpha_i)$. Therefore the execution of transition sequence, α, gives M^m s.t. $\forall l$=1,2,....m-1.

$$M_{sk}^{m} = \begin{cases} 2 & \text{if } p_{sk} \in \tilde{O}_s(t_{r1}) \\ 1 & \text{if } p_{sk} \in \{\ddot{O}_s(t_{r1}),\ \ddot{O}_s(t_m)\} \\ 0 & \text{otherwise} \end{cases}$$

Thus for $p_{sk} \in \tilde{O}_s(t_{r1})$, $M_{sk}^{0}[\tilde{O}]=1$, $M_{sk}^{j}[\tilde{O}]>1$, implying incorrectness, with i=1, j=m, and $\alpha=(t_{r1},t_{r2},....t_{r(l-1)},t_c,t_{rl},...t_m)$.

7.2. Correctness: Converse Case Proof

7.2.1. Redundancy

Given M_0 which minimally enables a transition sequence σ_i, and $\exists k$, $\exists\sigma_j$ s.t. $\sigma_i \cap \sigma_j = \varnothing$, $T_c \cap \sigma_i = \varnothing$, $T_c \cap \sigma_j \neq \varnothing$, $M' = \delta(M_0,\sigma_i)$, $M'' = \delta(M_0,\sigma_j)$, with $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, and $\exists(p_{rk},c_{ck})' \in M'_{ck}$, $\exists(p_{rk},c_{ck})'' \in M''_{ck}$ s.t. $(p_{rk},c_{ck})' = (p_{rk},c_{ck})''$, if σ_i and σ_j have the following cases:

Considering that σ_i and σ_j are nontrivial transition sequences, i.e. $\sigma_i \neq \varnothing$, and $\sigma_j \neq \varnothing$.

Let σ_i composed of a single transition t_{r0}

Since t_{r0} is minimally enabled in M_0, $\Rightarrow$ $\exists(p_{r0},c_{c0}) \in M_0$, $\exists(p_{r0},c_{s0}) \in M_0$ s.t. $\sum E(p_{r0},t_{r0}) < b >\leq M_0$ (Definition 5.7.) where <b> is $<((p_{r0},c_{c0}),(p_{r0},c_{s0}))>$.

and

$$M_{sk} = \begin{cases} 1 & \text{if } p_{sk} \in \tilde{O}_s(t_{r0}) \\ 0 & \text{otherwise} \end{cases}$$

$$M_{ck} = \begin{cases} 1 & \text{if } p_{ck} \in \tilde{O}_c(t_{r0}) \\ 0 & \text{otherwise} \end{cases}$$

and $M'=\delta(M_0,t_{r0})$,

$$M'_{sk}=\begin{cases}1 & if\ p_{sk}\in\{\tilde{O}_s(t_{r0}),\ddot{O}_s(t_{r0})\}\\ 0 & otherwise\end{cases}$$

$$M'_{ck}=\begin{cases}1 & if\ p_{ck}\in\ddot{O}_c(t_{r0})\\ 0 & otherwise\end{cases}$$

therefore $\Rightarrow \exists(p_{rk},c_{ck})'\in M'_{ck}$

Since there exists another sequence, σ_j, the following cases can happen

Case (I): σ_j is composed of transitions (t_c,t_{r1}).

As σ_j is enabled by M_0, therefore, $\tilde{A}_s(t_c)\subseteq\tilde{O}_s(t_{r0})$, $\tilde{A}_c(t_c)\subseteq\tilde{O}_c(t_{r0})$, $\ddot{A}_s(t_c)\subseteq\tilde{O}_s(t_{r1})$ and $\ddot{A}_c(t_c)\subseteq\tilde{O}_c(t_{r1})$.

Let $M''=\delta(M_0,\sigma_j)$, and since $\exists k$, s.t. $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, therefore, $\ddot{O}_s(t_{r0})\cap\ddot{O}_s(t_{r1})\neq\varnothing$ and $\ddot{O}_c(t_{r0})\cap\ddot{O}_c(t_{r1})\neq\varnothing$.

And $\exists p_{sk}$, s.t. $p_{sk}\in\ddot{O}_s(t_{r0})$ and $p_{sk}\in\ddot{O}_s(t_{r1})$, and $\exists p_{ck}$, s.t. $p_{ck}\in\ddot{O}_c(t_{r0})$ and $p_{ck}\in\ddot{O}_s(t_{r1})$.

$$M''_{sk}=\begin{cases}1 & if\ p_{sk}\in\{\tilde{O}_s(t_{r1}),\ddot{O}_s(t_{r1})\}\\ 0 & otherwise\end{cases}$$

$$M''_{ck}=\begin{cases}1 & if\ p_{ck}\in\ddot{O}_c(t_{r1})\\ 0 & otherwise\end{cases}$$

therefore $\Rightarrow \exists(p_{rk},c_{ck})'' \in M''_{ck}$

Since $(p_{rk},c_{ck})'=(p_{rk},c_{ck})''$, therefore
$E(t_{r0},\ddot{O}_c(t_{r0}))=E(t_{r1},\ddot{O}_c(t_{r1}))$.

Since $M'=\delta(M_0,\sigma_i)$, $M''=\delta(M_0,\sigma_j)$, and $E(t_{r0},\ddot{O}_c(t_{r0}))=E(t_{r1},\ddot{O}_c(t_{r1}))$. This indicates a pair of incorrect sequences $\sigma_i=(t_{r0})$ and $\sigma_j=(t_c,t_{r1})$, possibly having incorrect rules causing redundancy between the parent and child object classes.

Case (II): σ_j is composed of transitions $(t_c,t_{r1},t_{r2}\ldots.t_{rj})$.

Since σ_j is enabled by M_0, therefore

$\ddot{O}_s(t_{r(m)})=\tilde{O}_s(t_{r(m+1)})$ for m=1,2,…..j-1,
$\ddot{O}_c(t_{r(m)})=\tilde{O}_c(t_{r(m+1)})$ for m=1,2,…..j-1,
$\ddot{A}_c(t_c)=\tilde{O}_c(t_{r1})$, $\ddot{A}_s(t_c)=\tilde{O}_s(t_{r1})$.

Let $M^1=\delta(M_0,t_{r(m+1)})$,

$$M^1_{sk}=\begin{cases}1 & if\ p_{ck}\in\{\ddot{O}_s(t_{r(m)}),\tilde{O}_s(t_{r(m+1)})\}\\ 0 & otherwise\end{cases}$$

$$M^1_{ck}=\begin{cases}1 & if\ p_{ck}\in\ddot{O}_c(t_{r(m+1)})\\ 0 & otherwise\end{cases}$$

therefore $\Rightarrow \exists(p_{rk},c_{ck})^1 \in M^1_{ck}$

Similarly for any t_{m+u} in σ_j, where u=1,2,…..n,

$$M^{u}_{sk} = \begin{cases} 1 & if\ p_{sk} \in \{\ddot{O}_s(t_{r(m)}), \tilde{O}_s(t_{r(m+1)}) ..... \tilde{O}_s(t_{r(m+u)})\} \\ 0 & otherwise \end{cases}$$

$$M^{u}_{ck} = \begin{cases} 1 & if\ p_{ck} \in \ddot{O}_c(t_{r(m+u)}) \\ 0 & otherwise \end{cases}$$

therefore $\Rightarrow \exists (p_{rk}, c_{ck})^u \in M^{u}_{ck}$

Let $M''=M^u$ and $(p_{rk},c_{ck})^u=(p_{rk},c_{ck})''$,
Since $(p_{rk},c_{ck})'=(p_{rk},c_{ck})''$, therefore
$E(t_{r0},\ddot{O}_c(t_{r0}))=E(t_{r1},\ddot{O}_c(t_{rj}))$.

Since $M'=\delta(M_0,\sigma_i)$, $M''=\delta(M_0,\sigma_j)$, and $E(t_{r0},\ddot{O}_c(t_{r0}))=E(t_{r1},\ddot{O}_c(t_{rj}))$. This indicates a pair of incorrect sequences $\sigma_i=(t_{r0})$ and $\sigma_j=(t_c,t_{r1},t_{r2....}t_{rj})$, possibly having incorrect rules causing redundancy between the parent and child object classes.

7.2.2. Subsumption

Given M_0 which minimally enables a transition sequence σ_i, and $\exists k$, $\exists \sigma_j$ s.t. $\sigma_i \cap \sigma_j = \varnothing$, $T_c \cap \sigma_i = \varnothing$, $T_c \cap \sigma_j \neq \varnothing$, $M'=\delta(M_0,\sigma_i)$, $M''=\delta(M_0,\sigma_j)$, with $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, and $\exists (p_{rk},c_{ck})' \in M'_{ck}$, $\exists (p_{rk},c_{ck})'' \in M''_{ck}$ s.t. $(p_{rk},c_{ck})'' \subseteq (p_{rk},c_{ck})'$ if σ_i and σ_j have the following cases:

Considering that σ_i and σ_j are nontrivial transition sequences, i.e., . $\sigma_i \neq \varnothing$, and $\sigma_j \neq \varnothing$.

Let σ_i composed of a single transition t_{r0}

Since t_{r0} is minimally enabled in M_0, $\Rightarrow$ $\exists (p_{r0},c_{c0}) \in M_0$, $\exists (p_{r0},c_{s0}) \in M_0$ s.t. $\sum E(p_{r0},t_{r0}) < b >\leq M_0$ (Definition 4.2.) where <b> is $<((p_{r0},c_{c0}),(p_{r0},c_{s0}))>$.

and

$$M_{sk}=\begin{cases}1 & if\ p_{sk}\in \tilde{O}_s(t_{r0})\\ 0 & otherwise\end{cases}$$

$$M_{ck}=\begin{cases}1 & if\ p_{ck}\in \tilde{O}_c(t_{r0})\\ 0 & otherwise\end{cases}$$

and $M'=\delta(M_0,t_{r0})$,

$$M'_{sk}=\begin{cases}1 & if\ p_{sk}\in \{\tilde{O}_s(t_{r0}),\ddot{O}_s(t_{r0})\}\\ 0 & otherwise\end{cases}$$

$$M'_{ck}=\begin{cases}1 & if\ p_{ck}\in \ddot{O}_c(t_{r0})\\ 0 & otherwise\end{cases}$$

therefore $\Rightarrow \exists(p_{rk},c_{ck})'\in M'_{ck}$

Since there exists another sequence, σ_j, the following cases can happen

Let σ_j composed of transitions (t_c,t_{r1}).

As σ_j is enabled by M_0, therefore, $\tilde{A}_s(t_c)\subseteq\tilde{O}_s(t_{r0})$, $\tilde{A}_c(t_c)\subseteq\tilde{O}_c(t_{r0})$, $\ddot{A}_s(t_c)\subseteq\tilde{O}_s(t_{r1})$ and $\ddot{A}_c(t_c)\subseteq\tilde{O}_c(t_{r1})$.

Let M_c be the marking after firing transition t_c, since t_{r1} is minimally enabled in M_c, $\Rightarrow \exists(p_{r1},c_{c1})\in M_c$, $\exists(p_{r1},c_{s1})\in M_c$ s.t. $\sum E(p_c,t_{r1}) < b >\leq M_c$ (Definition 5.7.) where <b> is <((p_c,c_{c1}),(p_c,c_{s1}))>.

Let $M''=\delta(M_0,\sigma_j)$, and since $\exists k$, s.t. $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, therefore, $\ddot{O}_s(t_{r0})\cap\ddot{O}_s(t_{r1})\neq\varnothing$ and $\ddot{O}_c(t_{r0})\cap\ddot{O}_c(t_{r1})\neq\varnothing$.

And $\exists p_{sk}$, s.t. $p_{sk}\in\ddot{O}_s(t_{r0})$ and $p_{sk}\in\ddot{O}_s(t_{r1})$, and $\exists p_{ck}$, s.t. $p_{ck}\in\ddot{O}_c(t_{r0})$ and $p_{ck}\in\ddot{O}_s(t_{r1})$.

$$M''_{sk}=\begin{cases}1 & \text{if } p_{sk}\in\{\tilde{O}_s(t_{r1}),\ddot{O}_s(t_{r1})\}\\ 0 & \text{otherwise}\end{cases}$$

$$M''_{ck}=\begin{cases}1 & \text{if } p_{ck}\in\ddot{O}_c(t_{r1})\\ 0 & \text{otherwise}\end{cases}$$

therefore $\Rightarrow \exists(p_{rk},c_{ck})''\in M''_{ck}$

Since $(p_{rk},c_{ck})''\subseteq(p_{rk},c_{ck})'$, therefore
$E(t_{r1},\ddot{O}_c(t_{r1}))\subseteq E(t_{r0},\ddot{O}_c(t_{r0}))$

Since $M'=\delta(M_0,\sigma_i)$, $M''=\delta(M_0,\sigma_j)$, and $E(t_{r1},\ddot{O}_c(t_{r1}))\subseteq E(t_{r0},\ddot{O}_c(t_{r0}))$. This indicates a pair of incorrect sequences $\sigma_i=(t_{r0})$ and $\sigma_j=(t_c,t_{r1})$, possibly having incorrect rules causing subsumption between the parent and child object classes.

If $\sum E(p_c,t_{r1})<b><M_c$ and $E(t_{r1},\ddot{O}_c(t_{r1}))=E(t_{r0},\ddot{O}_c(t_{r0}))$, we have the case of a pair of incorrect rules having conditions subsumed between Parent Class and Child Class.

If $\sum E(p_c,t_{r1})<b>=M_c$ and $E(t_{r1},\ddot{O}_c(t_{r1}))\subset E(t_{r0},\ddot{O}_c(t_{r0}))$, we have the case of a pair of incorrect rules having subsumed actions between Parent Class and Child Class.

If $\sum E(p_c, t_{r1}) < b >< M_c$ and $E(t_{r1}, \ddot{O}_c(t_{r1})) \subset E(t_{r0}, \ddot{O}_c(t_{r0}))$, we have the case of a pair of incorrect rules having both conditions and actions subsumed between Parent Class and Child Class.

7.2.3. Ambiguity

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

Since Γ is minimally enabled,

$$M_{ck} = \begin{cases} 1 & if \quad p_{ck} \in \tilde{O}_c(\Gamma) \\ 0 & otherwise \end{cases}$$

Since for $\forall p_{sk} \in \ddot{O}_s(\Gamma), \forall p_{ck} \in \ddot{O}_c(\Gamma)$, $M_{sk}=0$, $M_{ck}=0$, $M'_{sk}>1$, $M'_{ck}>1$, at least two transitions must be active, and fired immediately one after the other. These are designated as $t_{r0}, t_{r1} \in \Gamma$ s.t. $\tilde{Q}_c(t_{r0}) \subset \tilde{O}_c(\Gamma)$, $\tilde{Q}_c(t_{r1}) \subset \tilde{O}_c(\Gamma)$, and $\tilde{Q}_c(t_{r0}) \cap \tilde{O}_c(t_{r1}) = \varnothing$ due to nonconflict criterion. Let $M^0 = \delta(M, t_{r0})$, $M' = \delta(M^0, t_{r1})$, s.t.

$$M^0_{ck} = \begin{cases} 1 & if \; p_{ck} \in \{\tilde{O}_c(t_{r1}), \ddot{O}_c(t_{r0})\} \\ 0 & otherwise \end{cases}$$

$$M^0_{sk} = \begin{cases} 1 & if \; p_{sk} \in \{\tilde{O}_s(t_{r1}), \ddot{O}_s(t_{r0})\} \\ 0 & otherwise \end{cases}$$

and

$$M'_{ck} = \begin{cases} 2 & if \; p_{ck} \in \ddot{O}_c(\Gamma) \\ 0 & otherwise \end{cases}$$

$$M'_{sk} = \begin{cases} 2 & if\ p_{sk} \in \ddot{O}_s(\Gamma) \\ 1 & if\ p_{sk} \in \tilde{O}_s(\Gamma) \\ 0 & otherwise \end{cases}$$

Thus for $\forall p_{ck} \in \ddot{O}_c(\Gamma)$, $p_{ck} \in \ddot{O}_c(t_{r1})$ and $p_{ck} \in \ddot{O}_c(t_{r0})$, $\forall p_{sk} \in \ddot{O}_s(\Gamma)$, $p_{sk} \in \ddot{O}_s(t_{r1})$ and $p_{sk} \in \ddot{O}_s(t_{r0})$. This indicates a pair of incorrect sequences $\sigma_i=(t_{r0})$ and $\sigma_j=(t_{r1})$, having problems of ambiguity possibly due to the existence of an indeterminate rule causing inclusive disjunction of conditions from different Object Classes.

Case (II): Rule with inclusive disjunction of IS-A actions from different Object Classes.

Since Γ is minimally enabled,

$$M_{ck} = \begin{cases} 1 & if\ p_{ck} \in \tilde{O}_c(\Gamma) \\ 0 & otherwise \end{cases}$$

Since for $\forall p_{sk} \in \ddot{O}_s(\Gamma), \forall p_{ck} \in \ddot{O}_c(\Gamma)$, $M_{sk}=0$, $M_{ck}=0$, $M'_{sk}>1$, $M'_{ck}>1$, $M''_{sk}>1$, $M''_{ck}>1$, at least three transitions must be active, and fired immediately one after the other. These are designated as t_{r0}, t_{r1} and $t_{r2} \in \Gamma$ s.t. $\tilde{O}_c(t_{r0}) \subset \tilde{O}_c(\Gamma)$, $\tilde{O}_c(t_{r1}) \subset \tilde{O}_c(\Gamma)$, $\tilde{O}_c(t_{r2}) \subset \tilde{O}_c(\Gamma)$, $\tilde{O}_c(t_{r0}) = \tilde{O}_c(t_{r1}) = \tilde{O}_c(t_{r2})$ and $\ddot{O}_c(t_{r0}) \cap \ddot{O}_c(t_{r2}) = \varnothing$, $\ddot{O}_c(t_{r0}) \subset \ddot{O}_c(t_{r1})$, $\ddot{O}_c(t_{r2}) \subset \ddot{O}_c(t_{r1})$.

Let $M^0=\delta(M,t_{r0})$, $M'=\delta(M^0,t_{r2})$, $M''=\delta(M',t_{r1})$, s.t.

$$M^0_{ck} = \begin{cases} 1 & if\ p_{ck} \in \{\tilde{O}_c(t_{r1}), \tilde{O}_c(t_{r2}), \ddot{O}_c(t_{r0})\} \\ 0 & otherwise \end{cases}$$

$$M^{0}_{sk} = \begin{cases} 1 & \text{if } p_{sk} \in \{\tilde{O}_s(t_{r1}), \tilde{O}_s(t_{r2}), \ddot{O}_s(t_{r0})\} \\ 0 & \text{otherwise} \end{cases}$$

and

$$M'_{ck} = \begin{cases} 1 & \text{if } p_{ck} \in \ddot{O}_c(\tilde{A}) \\ 0 & \text{otherwise} \end{cases}$$

$$M'_{sk} = \begin{cases} 1 & \text{if } p_{sk} \in \ddot{O}_s(\tilde{A}) \\ 0 & \text{otherwise} \end{cases}$$

and

$$M''_{ck} = \begin{cases} 2 & \text{if } p_{ck} \in \ddot{O}_c(\tilde{A}) \\ 0 & \text{otherwise} \end{cases}$$

$$M''_{sk} = \begin{cases} 2 & \text{if } p_{sk} \in \ddot{O}_s(\tilde{A}) \\ 0 & \text{otherwise} \end{cases}$$

Thus for $\forall p_{sk} \in \ddot{O}_s(\Gamma)$, $\forall p_{ck} \in \ddot{O}_c(\Gamma)$, $M_{sk}=0$, $M_{ck}=0$, $M'_{sk}>1$, $M'_{ck}>1$, $M''_{sk}>1$, $M''_{ck}>1$. This indicates three incorrect sequences $\sigma_i=(t_{r0})$, $\sigma_j=(t_{r1})$ and $\sigma_k=(t_{r2})$, having problems of ambiguity possibly due to the existence of an indeterminate rule causing inclusive disjunction of conditions from different Object Classes.

7.2.4. Cyclicity

Given M which minimally enables a transition sequence α, and $\exists j \geq i$, $\exists k$, s.t.

(i) $M^i \in [M^0> = \{M^0, M^1, M^2, \ldots M^i, .. M^j\}$,

(ii) $M^j=\delta(M^0,\alpha)$ for $j>0$,

(iii) $T_c \cap \alpha \neq \varnothing$;

(iv) $M^i_{sk}[\tilde{O}]>0$, $M^j_{sk}[\tilde{O}]>1$.

If α has the following cases:

Considering that α is a nontrivial transition sequence, i.e., $\alpha \neq \varnothing$ and α is composed of a series of transitions.

Let the sequence be

$\alpha=(t_{r1},t_{r2},\ldots..t_{ri},t_c,t_{r(i+1)},\ldots.t_{r(n)},\ldots.t_{r(m)})$

Case (I): The subsequence β consists of a single transition that begins at t_{ri} and ends at t_c for i=1.

Since t_{r1} is minimally enabled, let M^0 be the initial marking, such that

$$M^0_{sk}=\begin{cases}1 & if\ p_{sk} \in \tilde{O}_s(t_{r1})\\ 0 & otherwise\end{cases}$$

$$M^0_{ck}=\begin{cases}1 & if\ p_{ck} \in \tilde{O}_c(t_{r1})\\ 0 & otherwise\end{cases}$$

Thus, $M^0_{sk}[\tilde{O}_s(t_{r1})]=1$.

After firing of t_{r1}, the marking M^1 of $\ddot{O}_s(t_{r1})=1$, 0 otherwise and the marking M^1 of $\ddot{O}_c(t_{r1})=1$, 0 otherwise.

Since $\beta=(t_{r1},t_c)$, thus t_c is enabled after firing t_{r1}.

therefore, $\tilde{A}_c(t_c) \subseteq \ddot{O}_c(t_{r1})$ and $\tilde{A}_s(t_c) \subseteq \ddot{O}_s(t_{r1})$.

Let $M^2 = \delta(M^1, t_c)$

$$M^2_{sk} = \begin{cases} 2 & \text{if } p_{sk} \in \tilde{O}_s(t_{r1}) \\ 1 & \text{if } p_{sk} \in \ddot{O}_s(t_c) \\ 0 & \text{otherwise} \end{cases}$$

$$M^2_{ck} = \begin{cases} 1 & \text{if } p_{ck} \in \ddot{O}_c(t_c) \\ 0 & \text{otherwise} \end{cases}$$

Since t_{r1} is immediate enabled in α, and $\exists j \geq i$, for $j=2$, $i=0$, $\exists k$ s.t. $M^0_{sk}[\tilde{O}_s(t_{r1})]=1$, $M^2_{sk}[\tilde{O}_s(t_{r1})]>1$,

$\therefore \exists p_{sk}$ s.t. $p_{sk} \in \ddot{O}_s(t_c)$ and $\ddot{O}_s(t_{r1}) \cap \tilde{O}_s(t_c) \neq \varnothing$.

This indicates an incorrect sequence α containing $\beta=(t_{ri}, t_c)$, for $i=1$, possibly occurs within the object classes.

Case (II): The subsequence β consists of a series of transitions that begins at t_{ri} and ends at t_n for $i=1$ and $n>i$.

Since α is enabled by M^0, therefore

$$M^0_{sk} = \begin{cases} 1 & \text{if } p_{sk} \in \tilde{O}_s(t_{r1}) \\ 0 & \text{otherwise} \end{cases}$$

$$M_{ck}^{0} = \begin{cases} 1 & if\ p_{ck} \in \tilde{O}_c(t_{r1}) \\ 0 & otherwise \end{cases}$$

Thus, $M_{sk}^{0}\ [\tilde{O}_s(t_{r1})]=1$.

Let $M^1=\delta(M^0,t_{r1})$,

$$M_{sk}^{1} = \begin{cases} 1 & if\ p_{sk} \in \{\tilde{O}_s(t_{r1}),\ddot{O}_s(t_{r1})\} \\ 0 & otherwise \end{cases}$$

$$M_{ck}^{1} = \begin{cases} 1 & if\ p_{ck} \in \ddot{O}_c(t_{r1}) \\ 0 & otherwise \end{cases}$$

Since $\beta=(t_{r1},t_c,t_{r(i+1)},\ldots.t_{rn})$, thus t_c is enabled after firing t_{r1}.

therefore, $\tilde{A}_c(t_c)\subseteq\ddot{O}_c(t_{r1})$ and $\tilde{A}_s(t_c)\subseteq\ddot{O}_s(t_{r1})$.

Let $M^2=\delta(M^1,t_c)$

$$M_{sk}^{2} = \begin{cases} 1 & if\ p_{sk} \in \{\tilde{O}_s(t_{r1}),\ddot{O}_s(t_{r1}),\ddot{O}_s(t_c)\} \\ 0 & otherwise \end{cases}$$

$$M_{ck}^{2} = \begin{cases} 1 & if\ p_{ck} \in \ddot{O}_c(t_c) \\ 0 & otherwise \end{cases}$$

Similarly for any t_l in α, where l =2,3,4…..i-1,

$$M^{l}_{sk} = \begin{cases} 1 & if\ p_{sk} \in \{\tilde{O}_s(t_{r1}), \ddot{O}_s(t_{r1}), \ddot{O}_s(t_c), \ddot{O}_s(t_{r2}) \ldots . \ddot{O}_s(t_{rl})\} \\ 0 & otherwise \end{cases}$$

$$M^{l}_{ck} = \begin{cases} 1 & if\ p_{ck} \in \ddot{O}_c(t_l) \\ 0 & otherwise \end{cases}$$

Since t_{rn} is immediate enabled in α, and $\exists j=n \geq i=1$, $\exists k$ s.t. $M^{0}_{sk}[\tilde{O}_s(t_{r1})]=1$, $M^{n}_{sk}[\tilde{O}_s(t_{r1})]>1$, i.e.,

$$M^{n}_{sk} = \begin{cases} 2 & if\ p_{sk} \in \tilde{O}_s(t_{r1}) \\ 1 & if\ p_{sk} \in \{\ddot{O}_s(t_{r1}), \ddot{O}_s(t_c), \ddot{O}_s(t_{r2}) \ldots . \ddot{O}_s(t_{rl})\} \\ 0 & otherwise \end{cases}$$

This indicates an incorrect sequence α containing $\beta=(t_i,t_c,t_{i+1},\ldots t_n)$, possibly having a problem of cyclicity within the object classes. Note that α could be longer than sufficient to demonstrate the effect of such cyclicity if m>n>i.

7.3. Consistency: Forward Case Proof

7.3.1. Contradiction

Proposition 7.5a. For a given marking M_0, that minimally enables a nontrivial transition sequence σ_i, iff the HES has inconsistent 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) $(p_{rk},c_{ck})'=\neg(p_{rk},c_{ck})''$

If there exists incorrect rules applied to the object hierarchy of the following cases:

Case (I): Identical Conditions but Contradict Actions between Parent Class and Child Class.

Let $E(\Phi)$ be the arc expression function of the predicate IS-A member of Parent Class and
Let $E(\phi)$ be the arc expression function of the predicate IS-A member of Child Class.

In SCCPN representation, there should exists t_{r0}, t_{r1} and t_c such that

$E(\tilde{O}_c(t_{r0}),t_{r0}) - E(\Phi)=E(\tilde{O}_c(t_{r1}),t_{r1}) - E(\phi)$
$E(t_{r0},\ddot{O}_c(t_{r0}))=\neg E(t_{r1},\ddot{O}_c(t_{r1}))$

$\tilde{O}_c(t_{r0})=\tilde{A}_c(t_c)$, $\ddot{A}_c(t_c)=\tilde{O}_c(t_{r1})$, $\ddot{O}_c(t_{r0})=\ddot{O}_c(t_{r1})$,
$\tilde{O}_s(t_{r0})=\tilde{A}_s(t_c)$, $\ddot{A}_s(t_c)=\tilde{O}_s(t_{r1})$, $\ddot{O}_s(t_{r0})=\ddot{O}_s(t_{r1})$

Choose M_0 with a class token element (p_{r0},c_{c0}) and a state token (p_{r0},c_{s0}) s.t. t_{r0} is minimally enabled,

then $M_{sk}=1$ if $p_{sk}\in\tilde{O}_s(t_{r0})$, 0 otherwise.
And $M_{ck}=1$ if $p_{ck}\in\tilde{O}_c(t_{r0})$, 0 otherwise.

Since $\tilde{O}_s(t_{r0})=\tilde{A}_s(t_c)$ and $\tilde{O}_c(t_{r0})=\tilde{A}_c(t_c)$, t_c is enabled (Definition 5.7.).

Since t_c is enabled, the new marking in $Ä_c(t_c)=1$ and
has a colour of (p_{r1},c_{c1}) which is inherited from (p_{r0},c_{c0}). Where $E(p_{r0},t_c) - E(\Phi)=E(t_c,p_{r1}) - E(\phi)$ (Definition 5.8.).
Since $E(\tilde{O}_c(t_{r1}),t_{r1})=E(\tilde{O}_c(t_{r0}),t_{r0}) - E(\Phi) + E(\phi)$, therefore t_{r1} is enabled.

As from Definition 5.10, $\exists M'$, $\exists M''$ s.t. $M'=\delta(M_0,\sigma_i)$, $M''=\delta(M_0,\sigma_j)$ and $\sigma_i=(t_{r0})$, $\sigma_j=(t_c,t_{r1})$.

Therefore

$$M'_{sk}=\begin{cases}1 & if\ p_{sk}\in\{\tilde{O}_s(t_{r0}),Ö_s(t_{r0})\}\\ 0 & otherwise\end{cases}$$

$$M'_{ck}=\begin{cases}1 & if\ p_{ck}\in Ö_c(t_{r0})\\ 0 & otherwise\end{cases}$$

And the colour of the class token at $Ö_c(t_{r0})=(p_{rk},c_{ck})'$

$$M''_{sk}=\begin{cases}1 & if\ p_{sk}\in\{\tilde{O}_s(t_{r1}),Ö_s(t_{r1})\}\\ 0 & otherwise\end{cases}$$

$$M''_{ck}=\begin{cases}1 & if\ p_{ck}\in Ö_c(t_{r1})\\ 0 & otherwise\end{cases}$$

And the colour of the class token at $Ö_c(t_{r1})=(p_{rk},c_{ck})''$

Since $E(t_{r0},Ö_c(t_{r0}))=\neg E(t_{r1},Ö_c(t_{r1}))$,
therefore $(p_{rk},c_{ck})'=\neg(p_{rk},c_{ck})''$

Thus, for $p_{sk} \in \ddot{O}_s(t_{r0})$, $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, and for $p_{ck} \in \ddot{O}_c(t_{r0})$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, and $(p_{rk},c_{ck})'=\neg(p_{rk},c_{ck})''$, implying inconsistency with $\sigma_i=(t_{r0})$, $\sigma_j=(t_c,t_{r1})$ in the object classes.

Case (II): Contradictory pair of rules between Parent Class and Child Class.

Let E(Φ) be the arc expression function of the predicate IS-A member of Parent Class and
Let E(ϕ) be the arc expression function of the predicate IS-A member of Child Class.

In SCCPN representation, there should exist t_{r0}, t_{r1} and t_c such that

$\Sigma\{E(\tilde{O}_c(t_{r0}),t_{r0})\} - E(\Phi)=\Sigma\{E(\tilde{O}_c(t_{r1}),t_{r1})\} - E(\phi)$
$E(t_{r0},\ddot{O}_c(t_{r0}))=\neg E(t_{r1},\ddot{O}_c(t_{r1}))$

$\tilde{A}_c(t_c) \subset \tilde{O}_c(t_{r0})$, $\ddot{A}_c(t_c) \subset \tilde{O}_c(t_{r1})$, $\ddot{O}_c(t_{r0})=\ddot{O}_c(t_{r1})$,
$\tilde{A}_s(t_c) \subset \tilde{O}_s(t_{r0})$, $\ddot{A}_s(t_c) \subset \tilde{O}_s(t_{r1})$, $\ddot{O}_s(t_{r0})=\ddot{O}_s(t_{r1})$

Choose M_0 with a class token element (p_{r0},c_{c0}) and a state token (p_{r0},c_{s0}) s.t. t_{r0} is minimally enabled,

then $M_{sk}=1$ if $p_{sk} \in \tilde{O}_s(t_{r0})$, 0 otherwise.
And $M_{ck}=1$ if $p_{ck} \in \{\tilde{O}_c(t_{r0}) \cap \tilde{A}_c(t_c)\}$, 0 otherwise.

Since $\tilde{A}_s(t_c) \subset \tilde{O}_s(t_{r0})$ and $\tilde{A}_c(t_c) \subset \tilde{O}_c(t_{r0})$, t_c is enabled (Definition 5.7.).
Since t_c is enabled, the new marking in $\ddot{A}_c(t_c)=1$ and has a colour of (p_{r1},c_{c1}) which is inherited from (p_{r0},c_{c0}). Where $E(p_{r0},t_c) - E(\Phi)=E(t_c,p_{r1}) - E(\phi)$ (Definition 5.8.).
Since $\Sigma\{E(\tilde{O}_c(t_{r0}),t_{r0})\} - E(\Phi)=\Sigma\{E(\tilde{O}_c(t_{r1}),t_{r1})\} - E(\phi)$, therefore t_{r1} is enabled.

As from Definition 5.10, $\exists M'$, $\exists M''$ s.t. $M'=\delta(M_0,\sigma_i)$, $M''=\delta(M_0,\sigma_j)$ and $\sigma_i=(t_{r0})$, $\sigma_j=(t_c,t_{r1})$.

Therefore

$$M'_{sk}=\begin{cases}1 & if\ p_{sk}\in\{\tilde{O}_s(t_{r0}),\ddot{O}_s(t_{r0})\}\\ 0 & otherwise\end{cases}$$

$$M'_{ck}=\begin{cases}1 & if\ p_{ck}\in\ddot{O}_c(t_{r0})\\ 0 & otherwise\end{cases}$$

And the colour of the class token at $\ddot{O}_c(t_{r0})=(p_{rk},c_{ck})'$

$$M''_{sk}=\begin{cases}1 & if\ p_{sk}\in\{\tilde{O}_s(t_{r1}),\ddot{O}_s(t_{r1})\}\\ 0 & otherwise\end{cases}$$

$$M''_{ck}=\begin{cases}1 & if\ p_{ck}\in\ddot{O}_c(t_{r1})\\ 0 & otherwise\end{cases}$$

And the colour of the class token at $\ddot{O}_c(t_{r1})=(p_{rk},c_{ck})''$

Since $E(t_{r0},\ddot{O}_c(t_{r0}))=\neg E(t_{r1},\ddot{O}_c(t_{r1}))$,
therefore $(p_{rk},c_{ck})'=\neg(p_{rk},c_{ck})''$

Thus, for $p_{sk}\in\ddot{O}_s(t_{r0})$, $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, and for $p_{ck}\in\ddot{O}_c(t_{r0})$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, and $(p_{rk},c_{ck})'=\neg(p_{rk},c_{ck})''$, implying inconsistency with $\sigma_i=(t_{r0})$, $\sigma_j=(t_c,t_{r1})$.

Case(III): Contradictory chains of rules between the parent and child object classes.

Let E(Φ) be the arc expression function of the predicate IS-A member of Parent Class and

Let E(ϕ) be the arc expression function of the predicate IS-A member of Child Class.

In SCCPN representation, there should exists $\sigma_i=(t_{r1},t_{r2},\ldots t_{rj})$ and $\sigma_j=(t_c,t_{r0})$ such that

$E(\tilde{O}_c(t_{r0}),t_{r0}) - E(\Phi)=E(\tilde{O}_c(t_{r1}),t_{r1}) - E(\phi)$

$E(t_{r0},\ddot{O}_c(t_{r0}))=\neg E(t_{rj},\ddot{O}_c(t_{rj}))$

$\ddot{O}_s(t_{r(m)})=\tilde{O}_s(t_{r(m+1)})$ for m=1,2,…..j-1.

$\ddot{O}_c(t_{r(m)})=\tilde{O}_c(t_{r(m+1)})$ for m=1,2,…..j-1.

$\tilde{O}_c(t_{r1})=\tilde{A}_c(t_c)$, $\ddot{A}_c(t_c)=\tilde{O}_c(t_{r0})$,

$\tilde{O}_s(t_{r1})=\tilde{A}_s(t_c)$, $\ddot{A}_s(t_c)=\tilde{O}_s(t_{r0})$,

Choose M_0 with a class token element (p_{r0},c_{c0}) and a state token (p_{r0},c_{s0}) s.t. $\sigma_i=(t_{r1},t_{r2},\ldots t_{rj})$ is minimally enabled, i.e., $\forall m=1,2,3,\ldots j-1$,

$$\text{then } M_{sk}=\begin{cases}1 & if\ p_{sk}\in \tilde{A}_s(t_c)\\ 0 & otherwise\end{cases}$$

$$\text{And } M_{ck}=\begin{cases}1 & if\ p_{ck}\in \tilde{A}_c(t_c)\\ 0 & otherwise\end{cases}$$

The execution of transition sequence, σ_i, gives M' s.t. $\forall m=1,2,3,\ldots j$, $\ddot{O}_s(t_m)\in\ddot{O}_s(\sigma_i)$

$$M'_{sk}=\begin{cases}1 & if\ p_{sk}\in\{\tilde{O}_s(t_{r1}),\ddot{O}_s(s_i)\}\\ 0 & otherwise\end{cases}$$

And the colour of the class token at $\ddot{O}_c(t_j)=(p_{rk},c_{ck})'$

Since $\tilde{A}_s(t_c)=\tilde{O}_s(t_{r1})$ and $\tilde{A}_c(t_c)=\tilde{O}_c(t_{r1})$, t_c is enabled (Definition 5.7.).
Since t_c is enabled, the new marking in $\ddot{A}_c(t_c)=1$ and has a colour of (p_{r1},c_{c1}) which is inherited from (p_{r0},c_{c0}). Where $E(p_{r1},t_c) - E(\Phi)=E(t_c,p_{r0}) - E(\phi)$ (Definition 5.8.).
Since $E(\tilde{O}_c(t_{r1}),t_{r1}) - E(\Phi)=E(\tilde{O}_c(t_{r0}),t_{r0}) - E(\phi)$, therefore t_{r0} is enabled.

Let $M''_{ck}=\delta(M'_{ck},t_{r0})$,

$$M''_{ck}=\begin{cases}1 & \text{if } p_{ck}\in \ddot{O}_c(t_{r0})\\ 0 & \text{otherwise}\end{cases}$$

And the colour of the class token at $\ddot{O}_c(t_{r0})=(p_{rk},c_{ck})''$

$E(t_{r0},\ddot{O}_c(t_{r0}))=\neg E(t_{rj},\ddot{O}_c(t_{rj}))$, therefore $(p_{rk},c_{ck})'=\neg(p_{rk},c_{ck})''$

Case (IV): Self Contradictory chain of inference between the parent and child object classes.

Proposition 7.5b. For a given marking M^0, that minimally enables transition sequence α, iff the HES has inconsistent rules causing self-contradictory chain of inference between the parent and child object classes, then $\exists j$, $\exists k$ such that the sequence has the following properties:

(i) $M^i\in[M^0>=\{M^0,M^1,M^2,\ldots M^i,..M^j\}$,
(ii) $M^j=\delta(M^0,\alpha)$ for $j>0$,
(iii) $T_c\cap\alpha\neq\varnothing$;
(iv) $M_{sk}[\tilde{O}]=0$, $M^0_{sk}[\tilde{O}]>0$, $M^j_{sk}[\tilde{O}]>1$.
(v) $\exists(p_{rk},c_{ck})'\in M^0_{ck}$, $\exists(p_{rk},c_{ck})''\in M^j_{ck}$
(vi) $(p_{rk},c_{ck})'=\neg(p_{rk},c_{ck})''$

In SCCPN representation, there should exist $\alpha=(t_{r1},t_{r2},....t_{r(l-1)},t_c,t_{rl},...t_{rm})$ forming a connected path such that

$\tilde{O}_s(t_{r(l+1)})\subseteq\ddot{O}_s(t_{rl})$ for l=1,2,.....m-1,
$\tilde{O}_s(t_{r1})\subseteq\ddot{O}_s(t_{rm})$.
$E(\tilde{O}_c(t_{r1}),t_{r1})=\neg E(t_{rm},\ddot{O}_c(t_{rm}))$

Choose M^0 with a class token element $(p_{r0},c_{c0})'$ and a state token $(p_{r0},c_{s0})'$ s.t. $\alpha=(t_{r1},t_{r2},....t_{r(l-1)},t_c,t_{rl},...t_m)$ is minimally enabled,i.e., $\forall l$=1,2,....m-1,

then M_{sk}=1 if $p_{sk}\in\tilde{O}_s(t_{r1})$, 0 otherwise.
And M_{ck}=1 if $p_{ck}\in\tilde{O}_c(t_{r1})$, 0 otherwise.

$$M_{sk}^0=\begin{cases}1 & \textit{if } p_{sk}\in\tilde{O}_s(t_{r1}),\textit{where } M_{ck}^0(p_{ck})=1\\ 0 & \textit{otherwise}\end{cases}$$

i.e. $M_{sk}^0[\tilde{O}_s(t_{r1})]$=1 if $p_{sk}\in\tilde{O}_s(t_{r1})$
Since $\tilde{O}_s(t_{r1})\subseteq\ddot{O}_s(t_{rm})$, and $M^m=\delta(M^0,\alpha_i)$. Therefore the execution of transition sequence, α, gives M^m s.t. $\forall l$=1,2,....m-1.

$$M_{sk}^m=\begin{cases}2 & \textit{if } p_{sk}\in\tilde{O}_s(t_{r1})\\ 1 & \textit{if } p_{sk}\in\{\ddot{O}_s(t_{r1}),\ddot{O}_s(t_{rm})\}\\ 0 & \textit{otherwise}\end{cases}$$

And the colour of the class token at $\ddot{O}_c(t_{rm})=(p_{rk},c_{ck})''$

Since $E(\tilde{O}_c(t_{r1}),t_{r1})=\neg E(t_{rm},\ddot{O}_c(t_{rm}))$
therefore $(p_{rk},c_{ck})'=\neg(p_{rk},c_{ck})''$

Thus for $p_{sk} \in \tilde{O}_s(t_{r1})$, $M_{sk}[\tilde{O}]=0$, $M^0_{sk}[\tilde{O}]>0$, $M^j_{sk}[\tilde{O}]>1$, $\exists(p_{rk},c_{ck})' \in M^0_{ck}$, $\exists(p_{rk},c_{ck})'' \in M^j_{ck}$ and $(p_{rk},c_{ck})' = \neg(p_{rk},c_{ck})''$, implying inconsistent rules causing self-contradictory chain of inference between the parent and child object classes.

7.3.2. Deadend

The problems 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 concerns 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 7.6. 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 $M'=\delta(M,\sigma_j)$, $M'_{sk}=0$.

We consider a nontrivial case where $M \neq [0]$, i.e. there exists some $p_{sj} \notin \Omega$ that $M_{sj} \neq 0$.

In SCCPN representation, there should exist t_{r0} with some p_{sj} such that

$p_{sj} \in \ddot{O}_s(t_{r0})$ and $p_{sj} \notin \Omega$.

Choose M s.t.

X

$$M_{sj}=\begin{cases}1 & \text{if } p_{sj}\in \ddot{O}_s(t_{r0}) \text{ and } p_{sj}\notin \Omega \\ 0 & \text{otherwise}\end{cases}$$

Therefore, $\forall p_{sk}\in\Omega$, $M_{sk}=0$.

Since $\neg\exists t_i$ such that $p_{sj}\in\tilde{O}_s(t_i)$, $\neg\exists\sigma_j$ s.t. $M'=\delta(M,\sigma_j)$, thus $\sigma_j=\varnothing$ or $M'=M$, i.e. $M'_{sk}=0$ for $\forall p_{sk}\in\Omega$, implying inconsistency with t_{r0} being the deadend rule in the object hierarchy.

7.3.3. Unnecessary IF condition

Proposition 7.7. 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 inconsistent rules causing unnecessary IF conditions between the parent and child object classes, then $\exists k$, $\exists Y$ (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) $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$;

(v) $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$;

(vi) $\exists(p_{rk},c_{ck})'\in M'$, $\exists(p_{rk},c_{ck})''\in M''$;

(vii) $\exists Y$, $\sum_{(t,b)\in Y} E(p,t) < b > \leq (M'\cup M'')$

Let E(Φ) be the arc expression function of the predicate IS-A member of Parent Class and

Let E(ϕ) be the arc expression function of the predicate IS-A member of Child Class.

In SCCPN representation, there should exists t_{r0}, t_{ry} and t_c such that

$\tilde{A}_c(t_c) \subseteq \tilde{O}_c(t_{r0})$, $(\ddot{A}_c(t_c) \cup \ddot{O}_c(t_{r0})) = \tilde{O}_c(t_{ry})$,
$\tilde{A}_s(t_c) \subseteq \tilde{O}_s(t_{r0})$, $(\ddot{A}_s(t_c) \cup \ddot{O}_s(t_{r0})) = \tilde{O}_s(t_{ry})$,

Choose M_0 with a class token element (p_{r0},c_{c0}) and a state token (p_{r0},c_{s0}) s.t. t_{r0} is minimally enabled,

then $M_{sk}=1$ if $p_{sk} \in \tilde{O}_s(t_{r0})$, 0 otherwise.
And $M_{ck}=1$ if $p_{ck} \in \tilde{O}_c(t_{r0})$, 0 otherwise.

Since $\tilde{A}_c(t_c) \subseteq \tilde{O}_c(t_{r0})$ and $\tilde{A}_s(t_c) \subseteq \tilde{O}_s(t_{r0})$, t_c is enabled (Definition 5.7.).
Since t_c is enabled, the new marking in $\ddot{A}_c(t_c)=1$ and has a colour of (p_{r1},c_{c1}) which is inherited from (p_{r0},c_{c0}).

As from Definition 5.10, $\exists M'$, $\exists M''$ s.t. $M'=\delta(M_0,\sigma_i)$, $M''=\delta(M_0,\sigma_j)$ and $\sigma_i=(t_{r0})$, $\sigma_j=(t_c)$.

Therefore

$$M'_{sk} = \begin{cases} 1 & if\ p_{sk} \in \{\tilde{O}_s(t_{r0}), \ddot{O}_s(t_{r0})\} \\ 0 & otherwise \end{cases}$$

$$M'_{ck} = \begin{cases} 1 & if\ p_{ck} \in \ddot{O}_c(t_{r0}) \\ 0 & otherwise \end{cases}$$

And the colour of the class token at $\ddot{O}_c(t_{r0})=(p_{rk},c_{ck})'$

$$M''_{sk} = \begin{cases} 1 & if\ p_{sk} \in \{\tilde{A}_s(t_c), \ddot{A}_s(t_c)\} \\ 0 & otherwise \end{cases}$$

$$M''_{ck} = \begin{cases} 1 & \text{if } p_{ck} \in Ä_c(t_c) \\ 0 & \text{otherwise} \end{cases}$$

And the colour of the class token at $Ä_c(t_c)=(p_{rk},c_{ck})''$

Since $(Ä_c(t_c) \cup Ö_c(t_{r0})) = Õ_c(t_{ry})$ and $(Ä_s(t_c) \cup Ö_s(t_{r0})) = Õ_s(t_{ry})$ therefore t_{ry} is enabled, thus $\exists Y$ s.t. $\sum_{(t,b)\in Y} E(p,t) < b > \leq (M' \cup M'')$.

Thus, for $p_{sk} \in (Ä_s(t_c) \cup Ö_s(t_{r0}))$, $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, and for $p_{ck} \in (Ä_c(t_c) \cup Ö_c(t_{r0}))$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, and $\exists Y$ s.t. $\sum_{(t,b)\in Y} E(p,t) < b > \leq (M' \cup M'')$, implying inconsistent rules causing unnecessary IF conditions between the parent and child object classes with $\sigma_i=(t_{r0})$ and $\sigma_j=(t_c)$.

7.4. Consistency: Converse Case Proof

7.4.1. Contradiction

Given M_0 which minimally enables a transition sequence σ_i, and $\exists k$, $\exists \sigma_j$ s.t. $\sigma_i \cap \sigma_j = \varnothing$, $T_c \cap \sigma_i = \varnothing$, $T_c \cap \sigma_j \neq \varnothing$, $M'=\delta(M_0,\sigma_i)$, $M''=\delta(M_0,\sigma_j)$, with $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, and $\exists (p_{rk},c_{ck})' \in M'_{ck}$, $\exists (p_{rk},c_{ck})'' \in M''_{ck}$ s.t. $(p_{rk},c_{ck})' = \neg (p_{rk},c_{ck})''$, if σ_i and σ_j have the following cases:

Considering that σ_i and σ_j are nontrivial transition sequences, i.e. $\sigma_i \neq \varnothing$, and $\sigma_j \neq \varnothing$.

(A). Let σ_i composed of a single transition t_{r0}

Since t_{r0} is minimally enabled in M_0, $\Rightarrow \exists(p_{r0},c_{c0}) \in M_0$, $\exists(p_{r0},c_{s0}) \in M_0$ s.t. $\sum E(p_{r0},t_{r0}) < b > \leq M_0$ (Definition 5.7.) where <b> is <((p$_{r0}$,c$_{c0}$),(p$_{r0}$,c$_{s0}$))>.

and

$$M_{sk}=\begin{cases}1 & if\ p_{sk} \in \tilde{O}_s(t_{r0}) \\ 0 & otherwise\end{cases}$$

$$M_{ck}=\begin{cases}1 & if\ p_{ck} \in \tilde{O}_c(t_{r0}) \\ 0 & otherwise\end{cases}$$

and $M'=\delta(M_0,t_{r0})$,

$$M'_{sk}=\begin{cases}1 & if\ p_{sk} \in \{\tilde{O}_s(t_{r0}),\ddot{O}_s(t_{r0})\} \\ 0 & otherwise\end{cases}$$

$$M'_{ck}=\begin{cases}1 & if\ p_{ck} \in \ddot{O}_c(t_{r0}) \\ 0 & otherwise\end{cases}$$

therefore $\Rightarrow \exists(p_{rk},c_{ck})' \in M'_{ck}$

Since there exists another sequence, σ_j, the following cases can happen

Case (A.I): σ_j is composed of transitions (t_c,t_{r1}).

As σ_j is enabled by M_0, therefore, $\tilde{A}_s(t_c) = \tilde{O}_s(t_{r0})$, $\tilde{A}_c(t_c) = \tilde{O}_c(t_{r0})$, $\ddot{A}_s(t_c) = \tilde{O}_s(t_{r1})$ and $\ddot{A}_c(t_c) = \tilde{O}_c(t_{r1})$.

Let $M''=\delta(M_0,\sigma_j)$, and since $\exists k$, s.t. $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, therefore, $\ddot{O}_s(t_{r0}) \cap \ddot{O}_s(t_{r1}) \neq \varnothing$ and $\ddot{O}_c(t_{r0}) \cap \ddot{O}_c(t_{r1}) \neq \varnothing$.

And $\exists p_{sk}$, s.t. $p_{sk} \in \ddot{O}_s(t_{r0})$ and $p_{sk} \in \ddot{O}_s(t_{r1})$, and $\exists p_{ck}$, s.t. $p_{ck} \in \ddot{O}_c(t_{r0})$ and $p_{ck} \in \ddot{O}_s(t_{r1})$.

$$M''_{sk} = \begin{cases} 1 & if\ p_{sk} \in \{\tilde{O}_s(t_{r1}), \ddot{O}_s(t_{r1})\} \\ 0 & otherwise \end{cases}$$

$$M''_{ck} = \begin{cases} 1 & if\ p_{ck} \in \ddot{O}_c(t_{r1}) \\ 0 & otherwise \end{cases}$$

therefore $\Rightarrow \exists (p_{rk}, c_{ck})'' \in M''_{ck}$

Since $(p_{rk}, c_{ck})' = \neg(p_{rk}, c_{ck})''$, therefore
$E(t_{r0}, \ddot{O}_c(t_{r0})) = \neg E(t_{r1}, \ddot{O}_c(t_{r1}))$.

Since $M' = \delta(M_0, \sigma_i)$, $M'' = \delta(M_0, \sigma_j)$, and $E(t_{r0}, \ddot{O}_c(t_{r0})) = \neg E(t_{r1}, \ddot{O}_c(t_{r1}))$. This indicates a pair of inconsistent sequences $\sigma_i = (t_{r0})$ and $\sigma_j = (t_c, t_{r1})$, possibly having inconsistent rules causing contradiction between the parent and child object classes.

Case (A.II): σ_j is composed of transitions (t_c, t_{r1}).

As σ_j is enabled by M_0, therefore, $\tilde{A}_s(t_c) \subset \tilde{O}_s(t_{r0})$, $\tilde{A}_c(t_c) \subset \tilde{O}_c(t_{r0})$, $\ddot{A}_s(t_c) \subset \tilde{O}_s(t_{r1})$ and $\ddot{A}_c(t_c) \subset \tilde{O}_c(t_{r1})$.

Let $M'' = \delta(M_0, \sigma_j)$, and since $\exists k$, s.t. $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, therefore, $\ddot{O}_s(t_{r0}) \cap \ddot{O}_s(t_{r1}) \neq \varnothing$ and $\ddot{O}_c(t_{r0}) \cap \ddot{O}_c(t_{r1}) \neq \varnothing$.

And $\exists p_{sk}$, s.t. $p_{sk} \in \ddot{O}_s(t_{r0})$ and $p_{sk} \in \ddot{O}_s(t_{r1})$, and $\exists p_{ck}$, s.t. $p_{ck} \in \ddot{O}_c(t_{r0})$ and $p_{ck} \in \ddot{O}_s(t_{r1})$.

$$M''_{sk}=\begin{cases}1 & if\ p_{sk}\in\{\tilde{O}_s(t_{r1}),\ddot{O}_s(t_{r1})\}\\ 0 & otherwise\end{cases}$$

$$M''_{ck}=\begin{cases}1 & if\ p_{ck}\in\ddot{O}_c(t_{r1})\\ 0 & otherwise\end{cases}$$

therefore $\Rightarrow \exists(p_{rk},c_{ck})''\in M''_{ck}$

Since $(p_{rk},c_{ck})'=\neg(p_{rk},c_{ck})''$, therefore
$E(t_{r0},\ddot{O}_c(t_{r0}))=\neg E(t_{r1},\ddot{O}_c(t_{r1}))$.

Since $M'=\delta(M_0,\sigma_i)$, $M''=\delta(M_0,\sigma_j)$, and $E(t_{r0},\ddot{O}_c(t_{r0}))=\neg E(t_{r1},\ddot{O}_c(t_{r1}))$. This indicates a pair of inconsistent sequences $\sigma_i=(t_{r0})$ and $\sigma_j=(t_c,t_{r1})$, possibly having inconsistent rules causing contradiction between the parent and child object classes.

(B). Let σ_i composed of transitions $(t_{r1},t_{r2}\ldots.t_{rj})$

Since σ_i is enabled by M_0, therefore

$\ddot{O}_s(t_{r(m)})=\tilde{O}_s(t_{r(m+1)})$ for m=1,2,…..j-1,
$\ddot{O}_c(t_{r(m)})=\tilde{O}_c(t_{r(m+1)})$ for m=1,2,…..j-1,
$\tilde{A}_c(t_c)=\tilde{O}_c(t_{r1})$, $\tilde{A}_s(t_c)=\tilde{O}_s(t_{r1})$.

Let $M^1=\delta(M_0,t_{r(m+1)})$,

$$M^1_{sk}=\begin{cases}1 & if\ p_{ck}\in\{\ddot{O}_s(t_{r(m)}),\tilde{O}_s(t_{r(m+1)})\}\\ 0 & otherwise\end{cases}$$

$$M^1_{ck}=\begin{cases}1 & if\ p_{ck}\in\ddot{O}_c(t_{r(m+1)})\\ 0 & otherwise\end{cases}$$

therefore $\Rightarrow \exists(p_{rk},c_{ck})^1 \in M^1_{ck}$

Similarly for any t_{m+u} in σ_j, where u=1,2,…..n,

$$M^u_{sk} = \begin{cases} 1 & if\ p_{sk} \in \{\ddot{O}_s(t_{r(m)}), \tilde{O}_s(t_{r(m+1)}) ..... \tilde{O}_s(t_{r(m+u)})\} \\ 0 & otherwise \end{cases}$$

$$M^u_{ck} = \begin{cases} 1 & if\ p_{ck} \in \ddot{O}_c(t_{r(m+u)}) \\ 0 & otherwise \end{cases}$$

therefore $\Rightarrow \exists(p_{rk},c_{ck})^u \in M^u_{ck}$

Let $M'=M^u$ and $(p_{rk},c_{ck})^u=(p_{rk},c_{ck})'$

Case (B.I): σ_j is composed of transitions (t_c,t_{r0}).

As σ_j is enabled by M_0, therefore, $\tilde{A}_s(t_c)= \tilde{O}_s(t_{r1})$, $\tilde{A}_c(t_c)=\tilde{O}_c(t_{r1})$, $\ddot{A}_s(t_c)=\tilde{O}_s(t_{r0})$ and $\ddot{A}_c(t_c)= \tilde{O}_c(t_{r0})$.

Let $M''=\delta(M_0,\sigma_j)$, and since $\exists k$, s.t. $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, therefore, $\ddot{O}_s(t_{r0}) \cap \ddot{O}_s(t_{rj}) \neq \varnothing$ and $\ddot{O}_c(t_{r0}) \cap \ddot{O}_c(t_{rj}) \neq \varnothing$.

And $\exists p_{sk}$, s.t. $p_{sk} \in \ddot{O}_s(t_{r0})$ and $p_{sk} \in \ddot{O}_s(t_{rj})$, and $\exists p_{ck}$, s.t. $p_{ck} \in \ddot{O}_c(t_{r0})$ and $p_{ck} \in \ddot{O}_c(t_{rj})$.

$$M''_{sk} = \begin{cases} 1 & if\ p_{sk} \in \{\tilde{O}_s(t_{r0}), \ddot{O}_s(t_{r0})\} \\ 0 & otherwise \end{cases}$$

$$M''_{ck} = \begin{cases} 1 & if\ p_{ck} \in \ddot{O}_c(t_{r0}) \\ 0 & otherwise \end{cases}$$

therefore $\Rightarrow \exists(p_{rk},c_{ck})'' \in M''_{ck}$

Since $(p_{rk},c_{ck})' = \neg(p_{rk},c_{ck})''$, therefore
$E(t_{rj},\ddot{O}_c(t_{rj})) = \neg E(t_{r0},\ddot{O}_c(t_{r0}))$.

Since $M' = \delta(M_0,\sigma_i)$, $M'' = \delta(M_0,\sigma_j)$, and $E(t_{rj},\ddot{O}_c(t_{rj})) = \neg E(t_{r0},\ddot{O}_c(t_{r0}))$. This indicates a pair of inconsistent sequences $\sigma_i = (t_{r1},t_{r2}\ldots.t_{rj})$ and $\sigma_j = (t_c,t_{r0})$, possibly having inconsistent rules causing contradiction between the parent and child object classes.

Case (B.II): σ_j is composed of transitions (t_c,t_{r0}).

In Case (B.II) σ_j is enabled by M_j in stead of M^0 in Case (B.I).

Therefore, Given M^0 which minimally enables a transition sequence σ_i, and $\exists k$, $\exists\sigma_j$ s.t. $\sigma_i \cap \sigma_j = \varnothing$, $T_c \cap \sigma_i = \varnothing$, $T_c \cap \sigma_j \neq \varnothing$, $M^j = \delta(M^0,\sigma_i)$, $M'' = \delta(M^j,\sigma_j)$, with $M^0_{sk} = 1$, $M^j_{sk} = 0$, $M''_{sk} > 0$, $M^0_{sk} = 1$, $M^j_{ck} = 0$, $M''_{ck} > 0$, and $\exists(p_{rk},c_{ck})^0 \in M^0_{ck}$, $\exists(p_{rk},c_{ck})'' \in M''_{ck}$ s.t. $(p_{rk},c_{ck})^0 = \neg(p_{rk},c_{ck})''$, and $\exists t_{rk}$ s.t. $E(t_{rk},\ddot{O}_c(t_{rk})) = \neg E(t_{r0},\ddot{O}_c(t_{r0}))$, if σ_i and σ_j have the following properties:

Considering that σ_i and σ_j are nontrivial transition sequences, i.e. $\sigma_i \neq \varnothing$, and $\sigma_j \neq \varnothing$.

As σ_j is enabled by M_j, therefore, $\tilde{A}_s(t_c) = \ddot{O}_s(t_{rj})$, $\tilde{A}_c(t_c) = \ddot{O}_c(t_{rj})$, $\ddot{A}_s(t_c) = \tilde{O}_s(t_{r0})$ and $\ddot{A}_c(t_c) = \tilde{O}_c(t_{r0})$.

Let $M'' = \delta(M^j,\sigma_j)$, and since $\exists k$, s.t. $M^0_{sk} = 1$, $M^j_{sk} = 0$, $M''_{sk} > 0$, $M^0_{sk} = 1$, $M^j_{ck} = 0$, $M''_{ck} > 0$, therefore, $\ddot{O}_s(t_{r0}) \cap \tilde{O}_s(t_{r1}) \neq \varnothing$ and $\ddot{O}_c(t_{r0}) \cap \ddot{O}_c(t_{r1}) \neq \varnothing$.

And $\exists p_{sk}$, s.t. $p_{sk} \in \ddot{O}_s(t_{r0})$ and $p_{sk} \in \tilde{O}_s(t_{r1})$, and $\exists p_{ck}$, s.t. $p_{ck} \in \ddot{O}_c(t_{r0})$ and $p_{ck} \in \tilde{O}_c(t_{r1})$.

$$M''_{sk} = \begin{cases} 1 & if\ p_{sk} \in \{\tilde{O}_s(t_{r0}), \ddot{O}_s(t_{r0})\} \\ 0 & otherwise \end{cases}$$

$$M''_{ck} = \begin{cases} 1 & if\ p_{ck} \in \ddot{O}_c(t_{r0}) \\ 0 & otherwise \end{cases}$$

therefore $\Rightarrow \exists(p_{rk}, c_{ck})'' \in M''_{ck}$

Since $(p_{rk}, c_{ck})^0 = \neg(p_{rk}, c_{ck})''$, therefore $\exists t_{rk}$ s.t. $E(t_{rk}, \ddot{O}_c(t_{rk})) = \neg E(t_{r0}, \ddot{O}_c(t_{r0}))$.

Since $M^j = \delta(M^0, \sigma_i)$, $M'' = \delta(M^j, \sigma_j)$, and $E(t_{rk}, \ddot{O}_c(t_{rk})) = \neg E(t_{r0}, \ddot{O}_c(t_{r0}))$. This indicates a pair of inconsistent sequences $\sigma_i = (t_{r1}, t_{r2} \ldots . t_{rj})$ and $\sigma_j = (t_c, t_{r0})$, possibly having inconsistent rules causing contradiction between the parent and child object classes.

7.4.2. Deadend

Given a marking M s.t. $M_{sk}=0$ for $\forall p_{sk} \in \Omega$, and $\forall \sigma_l$ where $M' = (M, \sigma_l)$, $M'_{sk} = 0$, if σ_l has the following cases:

Assuming σ_l is the longest sequence that can be fired,

Case (I): σ_l is an empty sequence

Since $\sigma_l = \varnothing$, $\neg\exists$ any transition t_i for some p_{sj} being marked by M, s.t. $p_{sj} \in \tilde{O}_s(t_i)$. Therefore, p_{sj} belongs to a deadend. This indicates that the rule set is inconsistent having problems of deadend.

Case (II): σ_l is composed of a single transition t_0.

Let $M'=\delta(M,t_0)$, since

$$M'_{sj}=\begin{cases}1 & if\ p_{sj}\in Ö_s(t_0)\ and\ p_{sj}\notin \Omega\\ 0 & otherwise\end{cases}$$

Therefore, $M'_{sk}=0$ for $\forall p_{sk}\in\Omega$.

As σ_l is the longest sequence that can be fired and $\neg\exists$ any transition t_i after t_0 s.t. $p_{sj}\in \tilde{O}_s(t_i)$. Therefore, $t_i=\varnothing$ and p_{sj} belongs to a deadend. This indicates that the rule set is inconsistent having problems of deadend.

Case (III): σ_l is composed of a transitions $(t_1,t_2\ldots.t_m)$.

Let $M'=\delta(M,\sigma_l)$, since

$$M'_{sj}=\begin{cases}1 & if\ p_{sj}\in Ö_s(s_i)\ and\ p_{sj}\notin \Omega\\ 0 & otherwise\end{cases}$$

Therefore, $M'_{sk}=0$ for $\forall p_{sk}\in\Omega$.

As σ_l is the longest sequence that can be fired and $\neg\exists$ any transition t_i after σ_l s.t. $p_{sj}\in \tilde{O}_s(t_i)$. Therefore, $t_i=\varnothing$ and p_{sj} belongs to a deadend. In fact, M is on a path through σ_l to a deadend. This indicates that the rule set is inconsistent having problem of deadend.

7.4.3. Unnecessary IF condition

Given M_0 which minimally enables a transaction sequence σ_i, and $\exists\sigma_j$, $\exists k$, $\exists Y$(a step Y) s.t. $\sigma_i\cap\sigma_j=\varnothing$, $T_c\cap\sigma_i=\varnothing$, $T_c\cap\sigma_j\neq\varnothing$, $M'=\delta(M_0,\sigma_i)$, $M''=\delta(M_0,\sigma_j)$, $M_{sk}=0$, $M'_{sk}>0$,

$M''_{sk}>0$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, $\exists(p_{rk},c_{ck})'\in M'$, $\exists(p_{rk},c_{ck})''\in M''$, $\exists Y$ s.t. $\sum_{(t,b)\in Y} E(p,t) < b >\leq (M' \cup M'')$, if σ_i and σ_j has the following properties:

Considering that σ_i and σ_j are nontrivial transition sequences, i.e. $\sigma_i \neq \varnothing$, and $\sigma_j \neq \varnothing$.

Let σ_i composed of a single transition t_{r0}, since t_{r0} is minimally enabled in M_0, $\Rightarrow$ $\exists(p_{r0},c_{c0})\in M_0$, $\exists(p_{r0},c_{s0})\in M_0$ s.t. $\sum E(p_{r0},t_{r0}) < b >\leq M_0$ (Definition 5.7.) where <b> is $<((p_{r0},c_{c0}),(p_{r0},c_{s0}))>$.

and

$$M_{sk}=\begin{cases}1 & if\ p_{sk}\in \tilde{O}_s(t_{r0})\\ 0 & otherwise\end{cases}$$

$$M_{ck}=\begin{cases}1 & if\ p_{ck}\in \tilde{O}_c(t_{r0})\\ 0 & otherwise\end{cases}$$

and $M'=\delta(M_0,t_{r0})$,

$$M'_{sk}=\begin{cases}1 & if\ p_{sk}\in \{\tilde{O}_s(t_{r0}),\ddot{O}_s(t_{r0})\}\\ 0 & otherwise\end{cases}$$

$$M'_{ck}=\begin{cases}1 & if\ p_{ck}\in \ddot{O}_c(t_{r0})\\ 0 & otherwise\end{cases}$$

therefore $\Rightarrow$ $\exists(p_{rk},c_{ck})'\in M'_{ck}$

Since there exists another sequence, σ_j, the following case can happen:

Let σ_j is composed of transition t_c.

As σ_j is enabled by M_0, therefore, $\tilde{A}_s(t_c) \subseteq \tilde{O}_s(t_{r0})$, $\tilde{A}_c(t_c) \subseteq \tilde{O}_c(t_{r0})$.

Let $M''=\delta(M_0,\sigma_j)$,

$$M''_{sk} = \begin{cases} 1 & if\ p_{sk} \in \{\tilde{O}_s(t_c), \ddot{O}_s(t_c)\} \\ 0 & otherwise \end{cases}$$

$$M''_{ck} = \begin{cases} 1 & if\ p_{ck} \in \ddot{O}_c(t_c) \\ 0 & otherwise \end{cases}$$

therefore $\Rightarrow \exists(p_{rk},c_{ck})'' \in M''_{ck}$

Since $\exists Y$ s.t. $\sum_{(t,b)\in Y} E(p,t) < b > \leq (M' \cup M'')$, therefore $(\ddot{A}_c(t_c) \cup \ddot{O}_c(t_{r0})) = \tilde{O}_c(t_{ry})$ and $(\ddot{A}_s(t_c) \cup \ddot{O}_s(t_{r0})) = \tilde{O}_s(t_{ry})$.

Since $M'=\delta(M_0,\sigma_i)$, $M''=\delta(M_0,\sigma_j)$ and $\exists Y$ s.t. $\sum_{(t,b)\in Y} E(p,t) < b > \leq (M' \cup M'')$. This indicates a pair of inconsistent sequences $\sigma_i=t_{r0}$, and $\sigma_j=t_c$, possibly having inconsistent rules causing unnecessary IF conditions between the parent and child object classes.

7.5. Completeness: Forward Case Proof

7.5.1.Unreachability

The problems of completeness about a rule set containing incomplete rules applied to the object hierarchy might involve unreachability in terms of mutually exclusive

classes instantiation by some object instance. Testing for the completeness generally requires exhaustive search of the possible paths in the SCCPNs.

For the following analysis, we assume the collection of the goal states be Ω, and any goal state will be treated as a deadend in SCCPN.

Let the goal states that are in question be $\Gamma \subseteq \Omega$.

Proposition 7.8. 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 $M'=\delta(M,\sigma j)$, $M'_{sk}=0$.

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

Let $E(\Phi_a)$ be the arc expression function of the predicate IS-A member of Object Class A and
Let $E(\Phi_b)$ be the arc expression function of the predicate IS-A member of Object Class B.

In SCCPN representation, there should exists $\Gamma=\{t_{r0},t_{r1}\} \subseteq \Omega$ such that

$\tilde{O}_s(t_{r0}) \cap \tilde{O}_s(t_{r1})=\varnothing$, $\ddot{O}_s(t_{r0})=\ddot{O}_s(t_{r1})=\ddot{O}_s(\Gamma)$,
$\tilde{O}_c(t_{r0}) \cap \tilde{O}_c(t_{r1})=\varnothing$, $\ddot{O}_c(t_{r0})=\ddot{O}_c(t_{r1})=\ddot{O}_c(\Gamma)$,

and

$(E(\Phi_a),E(\Phi_b)) \in E(\tilde{O}_c(t_{r0}),t_{r0})$, $(E(\Phi_a),E(\Phi_b)) \in E(\tilde{O}_c(t_{r1}),t_{r1})$.
For t_{r0} to be minimally enabled, $\exists(p_{rk},c_{ck})$ s.t. the arc expression

$E(\tilde{O}_c(t_{r0}),t_{r0})<(p_{rk},c_{ck})>\leq M(p_{rk})$ (Definition 5.7)

Since the IS-A predicate in $E(\tilde{O}_c(t_{r0}),t_{r0})$ cannot simultaneously bind with two values (i.e. IS-A member of Object Class A and IS-A member of Object Class B), therefore,

$\forall\sigma_j$ where $M'=\delta(M,\sigma j)$, $M'_{sk}=0$ for $p_{sk}\in \ddot{O}_s(t_{r0})$.

Similarly,

For t_{r1} to be minimally enabled, $\exists(p_{rk},c_{ck})$ s.t. the arc expression

$E(\tilde{O}_c(t_{r1}),t_{r1})<(p_{rk},c_{ck})>\leq M(p_{rk})$ (Definition 5.7)

Since the IS-A predicate in $E(\tilde{O}_c(t_{r1}),t_{r1})$ cannot simultaneously bind with two values (i.e. IS-A member of Object Class A and IS-A member of Object Class B), therefore,

$\forall\sigma_j$ where $M'=\delta(M,\sigma j)$, $M'_{sk}=0$ for $p_{sk}\in \ddot{O}_s(t_{r1})$.

Thus for $p_{sk}\in \ddot{O}_s(t_{r0})$, $M_{sk}=0$ for $p_{rk}\in\Gamma\subseteq\Omega$, and $\forall\sigma_j$ where $M'=\delta(M,\sigma j)$, $M'_{sk}=0$ implying incomplete rules applied to the object hierarchy involving unreachability in terms of mutually exclusive classes instantiation by some object instance.

Case (II): Mutually exclusive classes chains.

Let $E(\Phi_a)$ be the arc expression function of the predicate IS-A member of Object Class A and
Let $E(\Phi_b)$ be the arc expression function of the predicate IS-A member of Object Class B.

In SCCPN representation, there should exists $\sigma_i=(t_{r1},t_{r2},\ldots t_{rj})$ and $\sigma_j=(t_c,t_{r0})$ such that

$\tilde{A}_s(t_s)\cap\tilde{O}_s(t_{r1})\neq\varnothing$, $\ddot{A}_s(t_c)=\tilde{O}_s(t_{r0})$,
$\tilde{A}_c(t_c)\cap\tilde{O}_c(t_{r1})\neq\varnothing$, $\ddot{A}_c(t_c)=\tilde{O}_c(t_{r0})$,
$\ddot{O}_s(t_{r(m)})=\tilde{O}_s(t_{r(m+1)})$ for m=1,2,…..j-1,
$\ddot{O}_c(t_{r(m)})=\tilde{O}_c(t_{r(m+1)})$ for m=1,2,…..j-1,
$\ddot{O}_s(t_{r0})=\ddot{O}_s(t_{rj})$, $\ddot{O}_c(t_{r0})=\ddot{O}_c(t_{rj})$,

and

$(E(\Phi_a),E(\Phi_b))\in E(\tilde{O}_c(t_{r0}),t_{r0})$, $(E(\Phi_a),E(\Phi_b))\in E(\tilde{O}_c(t_{rj}),t_{rj})$.

For t_{r0} to be minimally enabled, $\exists(p_{rk},c_{ck})$ s.t. the arc expression

$E(\tilde{O}_c(t_{r0}),t_{r0})<(p_{rk},c_{ck})>\leq M(p_{rk})$ (Definition 5.7)

Since the IS-A predicate in $E(\tilde{O}_c(t_{r0}),t_{r0})$ cannot simultaneously bind with two values (i.e. IS-A member of Object Class A and IS-A member of Object Class B), therefore,

$\forall\sigma_j$ where $M'=\delta(M,\sigma j)$, $M'_{sk}=0$ for $p_{sk}\in\ddot{O}_s(t_{r0})$.

Similarly,

For t_{rj} to be minimally enabled, $\exists(p_{rk},c_{ck})$ s.t. the arc expression

$E(\tilde{O}_c(t_{rj}),t_{rj})<(p_{rk},c_{ck})>\leq M(p_{rk})$ (Definition 5.7)

Since the IS-A predicate in $E(\tilde{O}_c(t_{rj}),t_{rj})$ cannot simultaneously bind with two values (i.e. IS-A member of Object Class A and IS-A member of Object Class B), therefore,

$\forall\sigma_j$ where $M'=\delta(M,\sigma j)$, $M'_{sk}=0$ for $p_{sk}\in \ddot{O}_s(t_{rj})$.

Thus for $p_{sk}\in \ddot{O}_s(t_{r0})$, $M_{sk}=0$ for $p_{rk}\in\Gamma\subseteq\Omega$, and $\forall\sigma_j$ where $M'=\delta(M,\sigma j)$, $M'_{sk}=0$ implying incomplete chain of rules applied to the object hierarchy involving unreachability in terms of mutually exclusive classes instantiation by some object instance.

7.6. Completeness: Converse Case Proof

7.6.1.Unreachability

If $\forall$ marking M, $M_{sk}=0$ for $p_{sk}\in\Gamma\subseteq\Omega$, and $\forall\sigma_l$ where $M'=\delta(M,\sigma_l)$, $M'_{sk}=0$, then the rule set is incomplete.

Choose M that asserts the input states such that $M_{sk}=0$ for any $p_{sk}\in\Gamma\subseteq\Omega$. Let any sequence, $\sigma_l=(t_1, t_2, t_3 \ldots t_m)$ where

$\tilde{O}_s(t_{ri})\subseteq\ddot{O}_s(t_{r(i-1)})$, for i=2, 3, ….m,

and let $M'=\delta(M,\sigma_l)$.

Since $M'_{sk}=0$ for $\forall\sigma_l$, therefore, $\neg\exists\sigma_l$ s.t. $p_{sk}\in\Gamma\subseteq\Omega$. Thus p_{sk} is not reachable from M.

This is valid for any marking M that asserts any input places. Hence, Γ is not reachable from any input state or any sequence of transactions. This indicates that the rule set is incomplete, possibly having problem of unreachability in the object classes.

7.7. Illustration of the Formal Methodology using the Personnel Selection Expert System

The Personnel Selection Expert System described in Chapter 6 will be used here as an illustration of the formal methodology developed. The Selection System is represented by a State Controlled Coloured Petri Net shown in Figure 6.2. The rules are labeled R1 to R12. The inheritance relations are represented by T1 to T3. S1 to S7 represent the predicates of these rules.

7.7.1. Subsumption

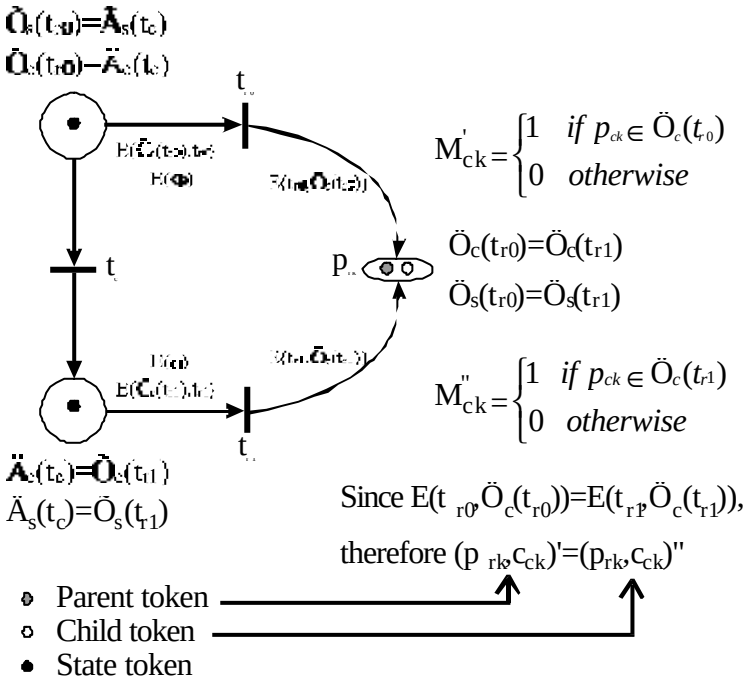


Figure 7.1. Illustration of Subsumption

To illustrate the application of our formal methodology, the net in Figure 7.1. are representing the followings:

$E(\tilde{O}_c(t_{r0}),t_{r0})$ = IF X is a junior office staff AND X's quality of service is Good AND X's seniority is High.

$E(\Phi)$ = X is a junior office staff.

$E(\tilde{O}_c(t_{r1}),t_{r1})$ = IF X is a clerk AND X's quality of service is Good AND X's seniority is High.

$E(\phi)$ = X is a clerk.

t_{r0} is Rule 1 which states that 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. t_{r1} is Rule 2 which states that 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.

(p_{r0},c_{c0}) is a junior office staff token in Place Class A and with colour (data value) "quality of service is Good" is TRUE and "seniority is High" is also TRUE.

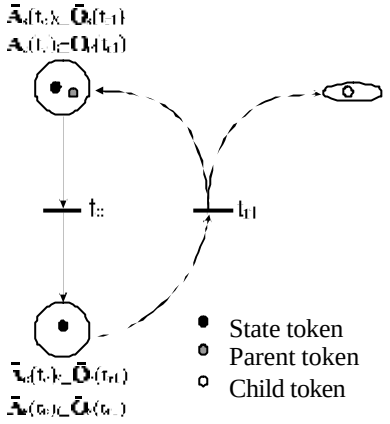
(p_{r1},c_{c1}) is a clerk token in Place Class A1 and with colour (data value) "quality of service is Good" is TRUE and "seniority is High" is also TRUE.

σ_i = firing of t_{r0}, σ_j= firing of t_c,t_{r1}.

$(p_{rk},c_{ck})' = (p_{rk},c_{ck})$" because the slot "promotion" is this two tokens reveals that they have the same value, i.e. "YES".

Thus, for $p_{sk} \in Ö_s(t_{r0})$, $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, and for $p_{ck} \in Ö_c(t_{r0})$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, and $(p_{rk},c_{ck})'=(p_{rk},c_{ck})''$, implying incorrectness with $\sigma_i=(t_{r0})$, $\sigma_j=(t_c,t_{r1})$.

7.7.2. Cyclicity



$$M^0_{sk} = \begin{cases} 1 & if\ p^{sk} \in \tilde{O}_r(t^{r1}) \\ 0 & otherwise \end{cases}$$

$$M^0_{ck} = \begin{cases} 1 & if\ p^{sk} \in \tilde{O}_r(t^{r1}) \\ 0 & otherwise \end{cases}$$

$$M^1_{sk} = \begin{cases} 2 & if\ p^{sk} \in \tilde{O}_r(t^{r1}) \\ 1 & if\ p^{sk} \in Ö_s(t^{r1}) \\ 0 & otherwise \end{cases}$$

$$M^1_{ck} = \begin{cases} 1 & if\ p^{sk} \in \tilde{O}_r(t^{r1}) \\ 0 & otherwise \end{cases}$$

Thus for $p^{sk} \in \tilde{O}_s(t^{r1})$, $M^i_{sk}[\tilde{O}_s]>0$, $M^j_{sk}[\tilde{O}_s]>1$,

implying incorrectness, with i=1, j=1, and $\alpha=(t^{r1})$

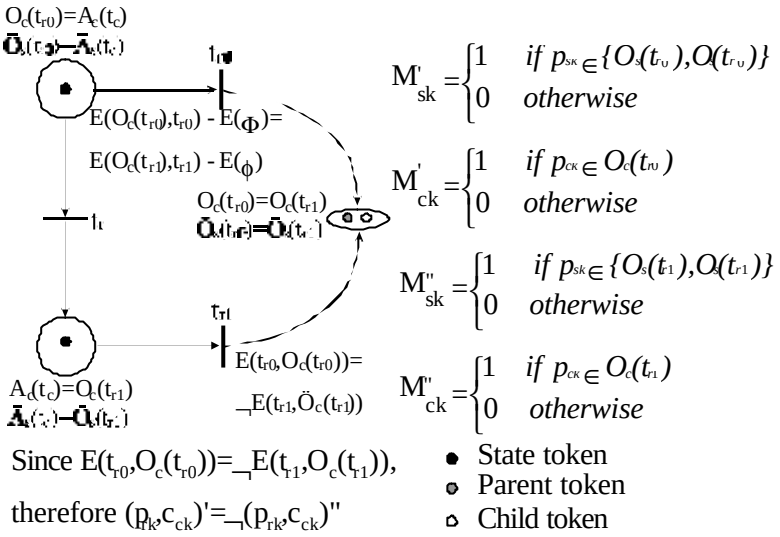
Figure 7.2. Illustration of Cyclicity

In Figure 7.2., t_{r1} is Rule 12 which states that IF X is a clerk THEN X is a junior office staff. This is a self-reference rule. The Marking M^1_{sk} is 2 because there are two state tokens deposited in the input place of t_c after firing of Rule 12. i.e.

$$M^1_{sk} = \begin{cases} 2 & if\ p_{sk} \in \tilde{O}_s(t_{r1}) \\ 1 & if\ p_{sk} \in Ö_s(t_{r1}) \\ 0 & otherwise \end{cases}$$

Thus for $p_{sk} \in \tilde{O}_s(t_{r1})$, $M^i_{sk}[\tilde{O}_s]>0$, $M^j_{sk}[\tilde{O}_s]>1$, implying incorrectness, with i=1, j=1, and $\alpha=(t_{r1})$.

7.7.3. Contradiction



$$M'_{sk} = \begin{cases} 1 & \text{if } p_{sk} \in \{O_s(t_{r0}), O_s(t_{r0})\} \\ 0 & \text{otherwise} \end{cases}$$

$$M'_{ck} = \begin{cases} 1 & \text{if } p_{ck} \in O_c(t_{r0}) \\ 0 & \text{otherwise} \end{cases}$$

$$M''_{sk} = \begin{cases} 1 & \text{if } p_{sk} \in \{O_s(t_{r1}), O_s(t_{r1})\} \\ 0 & \text{otherwise} \end{cases}$$

$$M''_{ck} = \begin{cases} 1 & \text{if } p_{ck} \in O_c(t_{r1}) \\ 0 & \text{otherwise} \end{cases}$$

Figure 7.3. Illustration of Contradiction

t_{r0} is Rule 5 which states that IF X is a junior office staff AND X's year of service is greater then Five THEN X's seniority is High. t_{r1} is Rule 4 which states that IF X is a clerk AND X's year of service is greater than Five THEN X's seniority is Not High.

$E(t_{r0},Ö_c(t_{r0}))$ is Rule 5's action part which states that X's seniority is "HIGH" while $E(t_{r1},Ö_c(t_{r1}))$ is Rule 4's action part which states that X's seniority is "NOT HIGH".

Thus, for $p_{sk} \in Ö_s(t_{r0})$, $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, and for $p_{ck} \in Ö_c(t_{r0})$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, and $(p_{rk},c_{ck})'=\neg(p_{rk},c_{ck})"$, implying inconsistency with $\sigma_i=(t_{r0})$, $\sigma_j=(t_c,t_{r1})$ in the object classes.

7.7.4. Unnecessary IF Condition

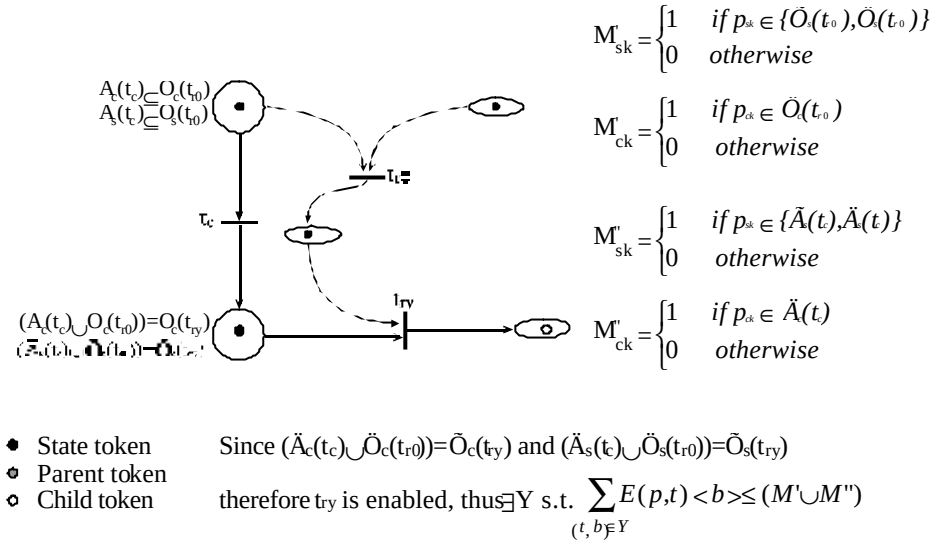


Figure 7.4. Illustration of Unnecessary IF condition

t_{r0} is Rule 6 which states that 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. t_{ry} is Rule 7 which states that IF X is a junior office staff AND X needs to attain training course THEN X's experience is Low.

Since $(\ddot{A}_c(t_c) \cup \ddot{O}_c(t_{r0})) = \tilde{O}_c(t_{ry})$ and $(\ddot{A}_s(t_c) \cup \ddot{O}_s(t_{r0})) = \tilde{O}_s(t_{ry})$ therefore t_{ry} is enabled, thus $\exists Y$ s.t. $\sum_{(t,b) \in Y} E(p,t) < b > \leq (M' \cup M'')$.

Thus, for $p_{sk} \in (\ddot{A}_s(t_c) \cup \ddot{O}_s(t_{r0}))$, $M_{sk}=0$, $M'_{sk}>0$, $M''_{sk}>0$, and for $p_{ck} \in (\ddot{A}_c(t_c) \cup \ddot{O}_c(t_{r0}))$, $M_{ck}=0$, $M'_{ck}>0$, $M''_{ck}>0$, and $\exists Y$ s.t. $\sum_{(t,b) \in Y} E(p,t) < b > \leq (M' \cup M'')$, implying inconsistent rules causing unnecessary IF conditions between the parent and child object classes with $\sigma_i=(t_{r0})$ and $\sigma_j=(t_c)$.

7.7.5. Unreachability

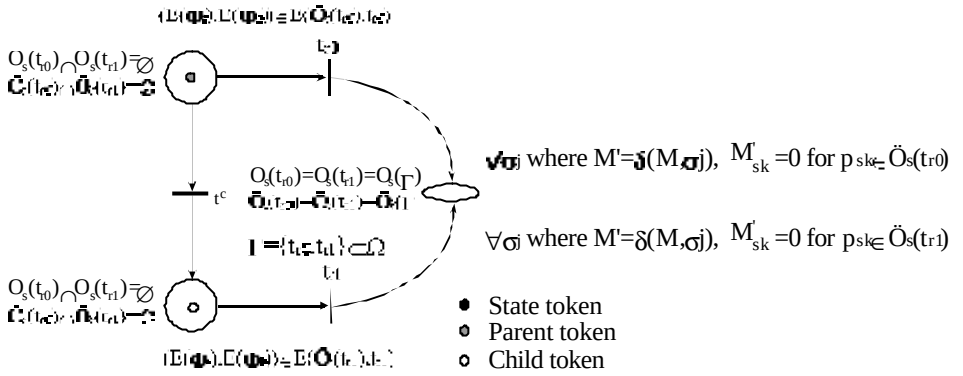


Figure 7.5. Illustration of Unreachability

Rule 8 in the Personnel Selection Expert System could be represented either by t_{r0} or t_{r1}. $E(\Phi_a)$ represents IF X is a clerk and $E(\Phi_b)$ represents IF X is a junior office staff, therefore $(E(\Phi_a),E(\Phi_b)) \in E(\tilde{O}_c(t_{r0}),t_{r0})$, $(E(\Phi_a),E(\Phi_b)) \in E(\tilde{O}_c(t_{r1}),t_{r1})$. In either case, in order to fire Rule 8, both conditions should be satisfied. (i.e. $E(\tilde{O}_c(t_{r0}),t_{r0}) < (p_{rk},c_{ck}) > \leq M(p_{rk})$). Since the IS-A predicate in $E(\tilde{O}_c(t_{r0}),t_{r0})$ cannot simultaneously bind with two values (i.e. IS-A clerk AND IS-A junior office staff), therefore,

$\forall \sigma_j$ where $M'=\delta(M,\sigma j)$, $M'_{sk}=0$ for $p_{sk} \in \ddot{O}_s(t_{r0})$.

Similarly,

For t_{r1} to be minimally enabled, $\exists (p_{rk},c_{ck})$ s.t. the arc expression

$E(\tilde{O}_c(t_{r1}),t_{r1}) < (p_{rk},c_{ck}) > \leq M(p_{rk})$

Since the IS-A predicate in $E(\tilde{O}_c(t_{r1}),t_{r1})$ cannot simultaneously bind with two values (i.e. IS-A clerk AND IS-A junior office staff), therefore,

$\forall\sigma_j$ where $M'=\delta(M,\sigma j)$, $M'_{sk}=0$ for $p_{sk}\in \ddot{O}_s(t_{r1})$.

Thus for $p_{sk}\in \ddot{O}_s(t_{r0})$, $M_{sk}=0$ for $p_{rk}\in\Gamma\subseteq\Omega$, and $\forall\sigma_j$ where $M'=\delta(M,\sigma j)$, $M'_{sk}=0$ implying incomplete rules applied to the object hierarchy involving unreachability in terms of mutually exclusive classes instantiation by some object instance.

7.8. Summary

A formal approach for the verification of Hybrid Expert Systems is given. Propositions are derived for checking the sequence of rule firings and properties inheritance in the object hierarchy. Based on the properties of reachability and colour tokens in the SCCPN, anomalies as defined in Chapter 5 can be formally located and detected in the model of the hybrid knowledge base. This is done exhaustively by minimally initiating any sequence of transitions and closely examining the reachability markings at each transition. The testing of any occurrence of alternative markings, multiple coloured tokens, deadlocks and the like lead to the system being verified in the end. Lastly, The Personnel Selection Expert System described in Chapter 6 is used as an illustration of the formal methodology developed.

CHAPTER 8. COMPLEXITY ANALYSIS OF SCCPN METHODOLOGY

8.1. Introduction

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, A. 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, M., 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.

8.2. Measuring Complexity

There is substantial reason to suggest that the underlying structure of the knowledge base is a major component of complexity (O'Leary, D. E., 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:

8.2.1. 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 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.

8.2.2. The size of the Rule Set and their 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, Z. & Suen, C.Y., 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.

8.2.3. 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.

8.2.4. 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.

8.3. Complexity Analysis

The complexity of verifying the anomalies in knowledge base, in the context of this thesis, 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.

8.3.1. 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 $2p \leq 2k(u+v) + 2m$. 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.

8.3.2. 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, X. et al., 1993; Christensen, S. & Petrucci, L., 1995; Kemper, P. 1996; Kondratyev, A. et at, 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, K., 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.

We propose the following algorithm for generating the (Occurrence Graph) reachability set of a SCCPN as follows:

```
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 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. The basic idea is as follows:

Let Matrix D represent a node with Marking M_0 for the (predicate + inheritance) states and control states, respectively. The matrix is m rows (one for each token place) and by n columns (one for each transition). E.g. Given the following SCCPN (*cf*. Figure 5.1.)

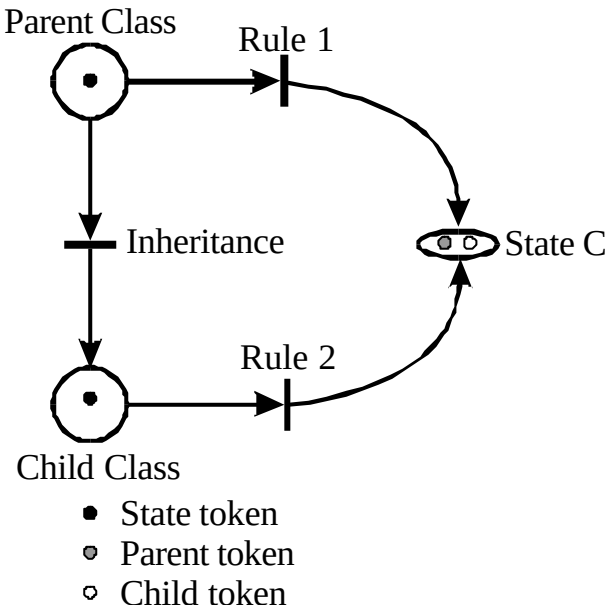
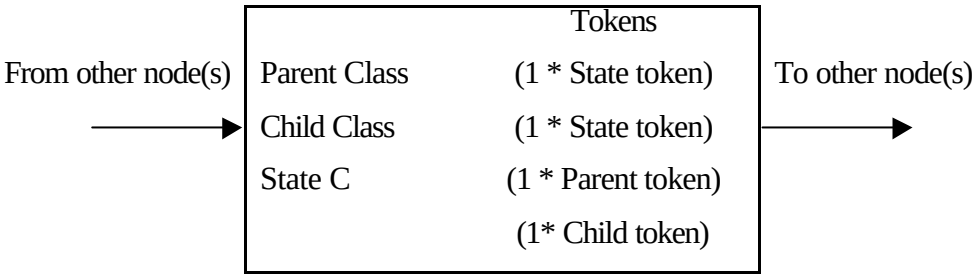


Figure 8.1. SCCPN for the generation of D

The Matrix D of the SCCPN in Figure 8.1 is



D represents a node of marking, and the content of this marking is described by the text inscription of the node. D can be linked to other nodes, and each link will represent the occurrence of a binding element, and the content of this binding element is described by the text attached to the corresponding arc. Detection of any form of error in the SCCPN will require the generation of a reachability tree for close examination. All markings that are reachable from a given marking will need to be stored for examination.

8.3.3. 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, M., 1991).

1. Put the Start Node [M_0] 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:

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

   where $D(r_i,r_j)$ 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 7.1 to 7.8.
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 higher scores are having larger changes 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 shall still have to be done in order to guarantee an error free knowledge base.

8.4. Comparative Performance of the Breadth-first search and Heuristic search algorithms for Occurrence Graphs Analysis

We will use the Personnel Selection Expert System described in Chapter 6 as an illustration of the comparative performance of the breadth-first and heuristic search

algorithms for Occurrence Graphs Analysis. In breadth-first search strategy, the root node is expanded first, then all the nodes generated by the root node are expanded next, and then their successors. Breadth-first search is a systematic strategy, the time and memory it takes to complete a search depends on the branching factor of these states. For example, if the root of the search tree generates n nodes at the first level, each of which generates n more nodes, for a total of n^2 at the second level. Each of these generates n more nodes at the third level, yielding n^3 nodes at the third level, and so on. In Occurrence Graphs, since the branching factor is not constant, and it also allows for many-to-many relationship among the reachable nodes, therefore, we have to rely on computer tool such as DESIGN/CPN to generate the Occurrence Graphs for searching of a particular marking. Using the heuristic search algorithm proposed in section 8.3, we based on the distance metric to guide the generation of next reachable marking. Since we have twelve rules in this Personnel Selection Expert System, the distance metric $D(r_i, r_j)$ is as follows:

i / j	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
R1	-	7/2	8/2	6/0	6/1	7/0	6/1	6/1	6/0	5/1	5/0	5/1
R2	7/2	-	8/3	6/0	6/1	7/1	6/0	6/1	6/0	5/0	5/1	5/1
R3	8/2	8/2	-	7/0	7/1	8/1	7/0	7/1	7/0	6/0	6/1	6/1
R4	6/0	6/0	7/1	-	5/1	6/1	5/0	5/1	5/0	4/0	4/1	4/1
R5	6/0	6/0	7/0	5/1	-	6/0	5/1	5/1	5/0	4/1	4/0	4/1
R6	7/0	7/1	8/1	6/1	6/0	-	6/0	6/1	6/0	5/0	5/1	5/1
R7	6/1	6/0	7/0	5/0	5/1	6/1	-	5/1	5/1	4/1	4/0	4/1
R8	6/1	6/1	7/1	5/1	5/1	6/1	5/1	-	5/0	4/1	4/1	4/2
R9	6/0	6/0	7/0	5/0	5/0	6/1	5/1	5/0	-	4/0	4/0	4/0
R10	5/1	5/0	6/0	4/0	4/1	5/0	4/1	4/1	4/0	-	3/0	3/1
R11	5/0	5/0	6/0	4/0	4/0	5/0	4/0	4/0	4/0	3/1	-	3/0
R12	5/0	5/1	6/1	4/1	4/0	5/1	4/0	4/1	4/0	3/0	3/1	-

Table 8.1. The Distance Matrix of Rule 1 to Rule 12

The above Table 8.1. can be simplified to Table 8.2. by taking out all the value which is divided by zero (i.e. no relationship identified). Note that when calculating the Distance Matrix, we do not include the inheritance transitions, it is because this

inheritance transitions will always have a higher priority (compare with rules) for firing. (i.e. for the identification of possible anomalies among object classes)

i / j	R1	R2	R3	R4	R5	R6	R7	R8	R9	R10	R11	R12
R1	-	3.5	4	-	6	-	6	6	-	5	-	5
R2	3.5	-	2.7	-	6	7	-	6	-	-	5	5
R3	4	2.7	-	-	7	8	-	7	-	-	6	6
R4	-	-	7	-	5	6	-	5	-	-	4	4
R5	-	-	-	5	-	-	5	5	-	4	-	4
R6	-	7	8	6	-	-	-	6	-	-	5	5
R7	6	-	-	-	5	6	-	5	5	4	-	4
R8	6	6	7	5	5	6	5	-	-	4	4	2
R9	-	-	-	-	-	6	5	-	-	-	-	-
R10	5	-	-	-	4	-	4	4	-	-	-	3
R11	-	-	-	-	-	-	-	-	-	3	-	-
R12	-	5	6	4	-	5	-	4	-	-	3	-

Table 8.2. The Final Distance Matrix of Rule 1 to Rule 12

Therefore, when we refer to the Occurrence Graphs described in Chapter 6, the effort for searching anomalies are:

8.4.1. Subsumption Case I (*c.f.* Figure 6.3b)

For breadth-first search, the number of nodes generated are six, and the effort required for searching the problem node is equal to the total number of comparisons and substitutions times the number of steps for each comparison and substitution. (i.e. 15*630 = 4,410 (*c.f.* Chapter 6.3.4.1.1.))

For heuristic search, the search sequence is as follows:

Giving an initial marking of M_0 (i.e. a junior office staff token deposited in input place Class A, and this token's slot "quality of service is Good" is TRUE and this token's slot "seniority is High" is also TRUE.) There are two enabled transactions

which are R1 and T1, given that inheritance transition always has a higher priority than rules, therefore, T1 is fired which generate marking M1. There are two enabled transitions in M1 which are R1 and R2, the distance between this two rules are: D(R1,R2) = 3.5 and D(R2,R1) = 3.5. Since their distances are the same, we can arbitrary take R1 to fire which generates marking M4. In M4, there is only one transition being enabled which is R2, after firing this transition, we have the marking M5. Since M5 is our goal, therefore, the total number of computation required is 3*630 = 1,890 which has a 57.15% reduction of efforts compared with the breadth-first search.

8.4.2. Subsumption Case II (*c.f.* Figure 6.4b)

For breadth-first search, the number of nodes generated are six, and the effort required for searching the problem node is equal to the total number of comparisons and substitutions times the number of steps for each comparison and substitution. (i.e. 9*630 = 5,670 (*c.f.* Chapter 6.3.4.1.2.))

For heuristic search, the search sequence is as follows:

Giving an initial marking of M_0 (i.e. a junior office staff token deposited in input place Class A, and this token's slot "quality of service is Good" is TRUE, slot "seniority is High" is TRUE and slot "local citizen" is also TRUE.) There are two enabled transactions which are R1 and T1, given that inheritance transition always has a higher priority than rules, therefore, T1 is fired which generate marking M1. There are three enabled transitions in M1 which are R1, R2 and R3, the distance between this three rules are: D(R1,R2) = 3.5, D(R1,R3) = 4, D(R2,R1) = 3.5, D(R2,R3) = 2.7, D(R3,R1) = 4 and D(R3,R2) =2.7. Since the minimal distance is between R3 and R2, therefore, we may chose either R3 or R2 to be fired, and this generates the marking M3. In M3, there is only one transition being enabled which is R1, after firing this transition, we have the marking M5. Since M5 is our goal,

therefore, the total number of computation required is 3*630 = 1,890 which has a 67% reduction of efforts compared with the breadth-first search.

8.4.3. Cyclicity (*c.f.* Figure 6.5b)

For breadth-first search, the number of nodes generated are fifteen, and the effort required for searching the problem node is equal to the total number of comparisons and substitutions times the number of steps for each comparison and substitution. (i.e. 29*630 = 18,270 (*c.f.* Chapter 6.3.4.1.2.))

For heuristic search, the search sequence is as follows:

Giving an initial marking of M_0 (i.e. a junior office staff token deposited in input place Class A). There are two enabled transactions which are R10 and T1, given that inheritance transition always has a higher priority than rules, therefore, T1 is fired which generate marking M1. There are two enabled transitions in M1 which are R12 and R10, the distance between this two rules are: D(R10,R12) = Nil and D(R12,R10) = 3. Therefore, we will fire R12 which generates marking M3. Since M3 is our goal, therefore, the total number of computation required is 2*630 = 1,260 which has a 93% reduction of efforts compared with the breadth-first search.

8.4.4. Contradiction (*c.f.* Figure 6.6b)

For breadth-first search, the number of nodes generated are six, and the effort required for searching the problem node is equal to the total number of comparisons and substitutions times the number of steps for each comparison and substitution. (i.e. 7*630 = 4,410 (*c.f.* Chapter 6.3.4.2.1.))

For heuristic search, the search sequence is as follows:

Giving an initial marking of M_0 (i.e. a junior office staff token deposited in input place Class A, this token's slot "Year of service greater than Five years" is TRUE). There are two enabled transactions which are R5 and T1, given that inheritance transition always has a higher priority than rules, therefore, T1 is fired which generate marking M1. There are two enabled transitions in M1 which are R4 and R5, the distance between this two rules are: D(R4,R5) = 5 and D(R5,R4) = 5. Since their distances are the same, we can arbitrary take R4 to fire which generates marking M3. In M3, there is only one transition being enabled which is R5, after firing this transition, we have the marking M5. Since M5 is our goal, therefore, the total number of computation required is 3*630 = 1,890 which has a 57.15% reduction of efforts compared with the breadth-first search.

8.4.5. Unnecessary IF Condition (*c.f.* Figure 6.7b)

For breadth-first search, the number of nodes generated are three, and the effort required for searching the problem node is equal to the total number of comparisons and substitutions times the number of steps for each comparison and substitution. (i.e. 2*630 = 1,260 (*c.f.* Chapter 6.3.4.2.2.))

For heuristic search, the search sequence is as follows:

Giving an initial marking of M_0 (i.e. a junior office staff token deposited in input place Class A, this token's slot "knowledge of work is Not Good" is TRUE and slot "English is Not Good" is also TRUE). There is only one transition being enabled which is T1, after T1 is fired that will generate marking M1. There is only one enabled transition in M1 which is R6, therefore, there is no need to compare the distance with other rules. After firing R6, it will which generate marking M2. Since M2 is our goal, therefore, the total number of computation required is 2*630 = 1,260. In this case, the effort required is the same compared with the breadth-first search.

8.4.6. Unreachability Case I (*c.f.* Figure 6.8b)

For breadth-first search, the number of nodes generated are two, and the effort required for searching the problem node is equal to the total number of comparisons and substitutions times the number of steps for each comparison and substitution. (i.e. 1*630 = 630 (*c.f.* Chapter 6.3.4.3.1.))

For heuristic search, the search sequence is as follows:

Giving an initial marking of M_0 (i.e. a junior office staff token deposited in input place Class A). There is only one transition being enabled which is T1, after T1 is fired that will generate marking M1. Since M1 is our goal, therefore, the total number of computation required is 1*630 = 630. In this case, the effort required is the same compared with the breadth-first search.

8.4.7. Unreachability Case II (*c.f.* Figure 6.9b)

For breadth-first search, the number of nodes generated are five, and the effort required for searching the problem node is equal to the total number of comparisons and substitutions times the number of steps for each comparison and substitution. (i.e. 6*630 = 3,780 (*c.f.* Chapter 6.3.4.3.1.))

For heuristic search, the search sequence is as follows:

Giving an initial marking of M_0 (i.e. a junior office staff token deposited in input place Class A). There is only two transitions being enabled which are T1 and T2, since they are both inheritance transitions, we may arbitrary take T1 for firing and this generates marking M1. There are two transition being enabled in M1, which are R6 and T2, giving inheritance transition has a higher priority than rules, we chose T2 to be fired, and this will generate marking M4. There is only one transition enabled in M4 which is R6, and after firing R6, this will generate marking M3. Since M3 is

our goal, therefore, the total number of computation required is 3*630 = 1,890. In this case, the effort saved is 50% compared with the breadth-first search.

Based on the above calculations, we can conclude that the average number of computation steps saved when using the heuristic search algorithm is (57.15% + 67% + 93% + 57.15% + 0% + 0% + 50%) / 7 = 46.33% as compared with the breadth-first search.

8.5. Summary

The analysis would not be complete without some form of performance analysis of the SCCPN model. It should be highlighted that the complexity issue of the SCCPN depends on a number of issues. These include the number of Object-Classes in the Frame Hierarchy; the size of the Rule Set and their Connectivity; the Depth of Reasoning Structure and the nature of Semantic Information; Transformation of Rules and Object hierarchy to SCCPN and Derivation of Occurrence Graph. Among these issues, the state space complexity of generation of the full Occurrence Graph is the most important part because a small SCCPN may generate a very large Occurrence Graph with exponential growth of nodes and arcs. Fortunately, recent research has been taken to allow for a partial examination of a subportion of the reachability graph, therefore reduce the efforts in deriving possible solutions. In the context of searching a particular marking in the SCCPN, we have developed a heuristic search algorithm which based on the concepts of rule clustering. The algorithm will shorten the time to search through an Occurrence Graph for location of errors and anomalies. Lastly, we used the Personnel Selection Expert System described in Chapter 6 as an illustration of the comparative performance of the breadth-first and heuristic search algorithms for Occurrence Graphs Analysis.

CHAPTER 9. POTENTIAL FOR EXTENSION AND CONCLUSION

9.1. Introduction

This is the final chapter of the thesis. We would like to spend some effort here to provide an assessment of the proposed State Controlled Coloured Petri Nets (SCCPNs) model and the methodology for supporting the description and verification of Hybrid Expert Systems. A discussion on the limitations of the approach and an investigation into potential opportunities for future research are given.

9.2. An Assessment of SCCPN Methodology

This research set out to provide a dynamic and a state by state analysis of a Hybrid Expert System in order to verify its correctness, consistency and completeness in a defined domain space. It recognized the importance and a need to search for a means of representing knowledge and its structure syntactically and semantically that could support and automate the processes involved in verification. With the development of the SCCPN model we have been able to simulate the effects of possible chained inference in an object hierarchy integrated with production rules, and considerably expand the scope of verification.

Several constraints were introduced to simplify the verification process. First, the verification of Rule/Frame-based Expert Systems concentrates on the problems introduced by the inheritance mechanism within the object hierarchy. As a result of this inheritance, various forms of errors and anomalies exist which called for attention. The SCCPN methodology at present is designed for tackling this set of problems. Attention was not put on other hybrid aspects although SCCPN has the potential to describe more complicated structures, such as those mentioned in Chapter 3, e.g. rules with demons and rules with methods.

Secondly, a special feature of introducing state tokens to represent the states of a predicate is not only to increase the expressive power of the model, but also to allow the use of its colours as a necessary basis for verifying the knowledge base. This characteristic together with the colours of the object tokens are used to trap more subtle versions of anomalies, particularly contradiction and deadend. SCCPN that supports the independent use of input and output constraints and operations permit efficient detection of a wider set of problems and it is useful in locating possible errors in the knowledge base.

In order to allow for greater applicability to a variety of hybridizing mechanism, (how different types of rules are attached to the object hierarchy), a set of schemes was provided in Chapter 5, so that all production rules had to be transformed into the appropriate specific SCCPN format. Rules that involved disjunction of conditions or actions needed to be decomposed into a number of alternative rules. This constraint requires that some effort be expanded in converting a rule set to the standard form before any verification should be attempted.

Additional major features specific to this research are the capability of performing constant maintenance of the predicate states as well as the slot values of the object class instances. The former is achieved by the introduction of a self-loop attached to an individual input place. Its significance includes an opportunity to update the state of the predicate. The latter is done by evaluating the corresponding arc expression functions, which provide the basis for dynamic verification of the knowledge base.

It has been shown to be possible that all anomalies extensively outline in Chapter 7 were detectable using the SCCPN methodology. This might require exhaustive testing of the knowledge and involve a certain degree of complexity. As such, we recognize the importance of developing a formal model in that it allows delineation between semantics, the property being proved, and the actual proof itself. A number of propositions were therefore derived from the principle of reachability markings, which provide a formal basis to give some guarantee of the validity of the verification process.

9.3. Limitation of the Research

This research has a number of limitations. The most serious one relates to the completeness of the knowledge representation. Besides simple production rules and frames structures, there are other kinds of knowledge which need explicit scheme for their description and representation. E.g. Common Sense, Knowledge about how to learn from experience, Uncertainty reasoning, Non-monotonic reasoning, etc. These kinds of knowledge are high level knowledge whose representation are not yet precisely defined and formalized. Should they be better represented by some means other than the production rule and simple frame structures? If these were the case, it would be conceptually different and beyond the present format of SCCPN that could handle otherwise.

Furthermore, the taxonomy of anomalies so defined might not be appropriate to cover all forms of verification problem that could arise in the knowledge base should the knowledge be represented using different paradigms. Extra set of principles and criteria would have to be defined to identify any possible anomalies in other schemes such as: Abduction; Case-Based Reasoning; Circumscription; Default Logic; Fuzzy Logic; Non-monotonic Reasoning; Temporal Systems and the like.

Another major limitation of using SCCPN is the state space complexity of the Occurrence Graphs. Obviously, such a graph may become very large, even for small SCCPNs. They may grow exponentially with respect to the number of independent processes. Although recent research (Li, X. et al., 1993; Christensen, S. & Petrucci, L., 1995; Kemper, P. 1996; Kondratyev, A. 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 development of the partition algorithms, theories of sub-net analysis and reduction methodologies for Occurrence Graphs are beyond the scope of this research.

9.4. Future Research

For Expert Systems technology to gain wider acceptance, the ability to integrate it with other forms of information technology and development methods, is necessary. Every limitation of an Expert System presents opportunities for further research. Hence we would like to highlight some of the potential of the present SCCPN model at a conceptual level, that is worthy for future development and research.

9.4.1. Extension of Methodology for Modelling Hybrid Expert Systems with Uncertainty

Imprecision plays an important role in many Expert System applications. It is involved in a variety of applications that are very important and potentially life saving. Most of the more difficult problems for which experts are available have a high amount of imprecision associated with them. In knowledge abstraction, uncertainty might be present because of noise in observation and incompleteness of knowledge. Thus so-called approximate, inexact, plausible reasoning methods are strongly needed in knowledge engineering. The ability to represent and reason about information with uncertainty is dependent upon the form and detail of the constructs of this information. A number of numerical approaches had been proposed in the literature (Baldwin, J. F., 1985; Buckley, J. J. et al., 1986; Grzymala-Busse, 1991; Durkin, J., 1994). These approaches are based on various kinds of theoretical calculi such as Bayesian inference, Dempster-Shafer's Belief theory or Zadeh's Fuzzy Set Theory. The construction of Expert System and other intelligent computer systems require a sophisticated mechanism for representing and reasoning with uncertain information. The verification of these Expert Systems with Uncertainty involves investigation of suitable measures for consistency, correctness and completeness of "uncertain" propositions. Ordinary and high level Petri Nets have been proposed (Chen, S. M., et al. 1990; Looney, C. G., 1994; Scarpelli, H. & Gomide, F., 1994b; Yeung, D. S. & Tsang, E. C. C., 1994; Cao, T., & Sanderson, A. C., 1995) as knowledge representation formalisms where structural and behavioural properties of the net can be used to prove properties of the system being modelled or to verify the knowledge base integrity. These approaches consist of using the structural properties of the high level Petri Nets model representing a Fuzzy knowledge base to verify the

necessary conditions for the existence of potential conflicts. However, all these techniques only work with Rule-based Expert Systems (i.e. Rules with certainty factors; Fuzzy Rules; Rules with probabilities attachments). Research into how a hybrid approach of knowledge representation will affect these "uncertain" representations of information is necessary for future enhancement of Expert System technology. This requires an extension of the current definitions of SCCPNs to cover these "uncertain" cases.

9.4.2. Extension of Methodology for Modelling Hybrid Expert Systems with Temporal Properties

In recent years, the increasing need for reasoning about time in various areas of artificial intelligence applications (Allen, J. F., 1983; Berthomieu, B. & Diaz, M., 1991; Yao, Y., 1994) requires models that can handle both qualitative and quantitative temporal information. These temporal qualitative relations indicated how two propositions related to each other in a specific time interval. (e.g. Tom goes to school either by Train (Proposition 1 (P1)) OR by Bus (P2). Once he arrives at the school, he either has breakfast (P3) AND read newspaper (P4) OR goes to the classroom (P5) OR plays tennis with Peter (P6)..etc). Given such temporal information, we want to verify the system's consistency, (e.g. Does the proposition P holds at certain time t? Is it possible that both temporal proposition P and Q hold at certain time t?)

Representing these concurrent, temporal relationships in Hybrid Expert Systems will definitely be another area of future research.

9.4.3. Extension of Methodology for Modelling Hybrid Expert Systems with Case-Based Systems

Access to a large mental library of past cases is what it distinguishes most experts from non-experts, particularly in subjects where there are no fundamental models. Expertise in those subjects is applied, and evolves, by generalization from cases or by discovery of regularities and links between cases. Any computing scheme that accommodates cases should therefore make it as easy as

possible to create, maintain and apply cases. Obviously, the validation and verification of these Case-based Systems are never less important that the traditional rule based Expert Systems. In Case-Based Expert Systems, Cases typically are represented using Frames, this indicates that verification approaches for Case-based structures could exploit the frame structure. In addition, these systems add solved cases to their case library, and previous solutions become part of their experience. This is a critical difference from Rule-based system where the knowledge is in the forms of production rules and are usually static. Errors and anomalies in a Case-Based Expert System (O'Leary, D. E. 1993) may include: (1) Misspelling or using different names for the same Case object (attributes); (2) Duplicate Cases; (3) Missing Cases attributes; (4) Cyclic inheritance of Cases; (5) Conflicts in cases and (6) Problems in matching Cases.

Application of our SCCPN methodology to cope with Hybrid Expert Systems that involves case-based reasoning seems another interesting research topic for the future work.

9.4.4. Extension of Methodology for Modelling Hybrid Expert Systems with Conventional Software Systems

Many real world applications are neither purely conventional nor purely knowledge based. A beneficial consequence of extending our SCCPN model to cover conventional software systems verification is that it will be able to tackle problems from some large-scale hybrid systems (Preece, A. D. 1995) (Conventional and Expert Systems integration). These may include: (1) the problems of hybridizing the procedural and declarative problem solving paradigm. (2) the integration of object-oriented programming method with rules. (3) other forms of hybridization namely: Blackboard reasoning, default-based approaches, non-monotonic reasoning, and the like.

9.5. Summary

An assessment of the proposed SCCPN methodology is given, it is followed by a discussion on the limitations of the approach. Lastly, the potential for future

research are suggested, these 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.

REFERENCE

Abbott, R. J., 1987 — Abbott, R. J., *Knowledge Abstraction. Protocol Analysis*, published by MIT Press, Cambridge, MA, 1987.

Agarwal, R. & Tanniru, M., 1992 — Agarwal, R. and Tanniru, M., "A Petri Net Based Approach for Verifying the integrity of Production systems," *International Journal of Man-Machine Studies*, Vol. 36, pp. 447-468, 1992.

Aikins, J. S., 1993 — Aikins, J. S., "Prototypical Knowledge for Expert Systems: a retrospective analysis". In Bobrow D.G. (Ed.) *Artificial Intelligence*. Vol. 59, pp. 207-211, Elsevier, Amsterdam, 1993.

Ali, S. S., 1993 — Ali, S. S., "Node Subsumption in a Propositional Semantic Network with Structured Variables," in *Proceedings of the 6th Australia Joint Conference on AI*, pp. 255-260, 1993.

Allen, J. F., 1983 — Allen, J. F., "Maintaining Knowledge about temporal intervals," *Communications of ACM*, Vol. 26, No. 11, pp. 832-843, 1983.

Antoniou, G. & Sperschneider, V., 1995 — Antoniou, G. and Sperschneider, V., "On the Verification of Modular Logical Knowledge Bases," *Expert Systems with Applications*, Vol. 8, No. 3, pp. 351-357, 1995.

Ayel, M. & Vignollet, L., 1994 — Ayel, Marc and Vignollet, Laurence, "SYCOJET and SACCO, Two tools for Verifying Expert Systems," *International Journal of Intelligent Systems*, Vol. 9, pp. 357-382, 1994.

Baase, S., 1988 — Baase, S., *Computer Algorithms*, published by Addison-Wesley Company, 2nd Edition, 1988.

Baldwin, J. F., 1985 — Baldwin, J. F., "Fuzzy Sets and Expert Systems," Information Science, Vol. 36, pp. 123-156.

Barthes, J. P. A., 1994 — Barthes, J. P. A., "Developing integrated object environments for building large knowledge-based

systems," *Int. J. Human Computer Studies*, Vol. 41, pp. 33-58, 1994.

Beauvieux, A., 1990 — 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.

Becker, L. et al., 1994 — Becker, L., Duckworth, J. and Laznovsky, A., "Automated Test Generation and Evaluation for Real-Time Expert Systems," *International Journal of Intelligent Systems*, Vol. 9, pp. 659-682, 1994.

Berthomieu, B. & Diaz, M., 1991 — Berthomieu, B. and Diaz, M., "Modelling and verification of time dependent systems using time Petri Nets," *IEEE Transactions on Software Engineering*, Vol. 17, No. 3, pp. 259-273, 1991.

Billington, J. & Reisig, W., 1996 — Billington, J. and Reisig, W., *Application and Theory of Petri Nets 1996*, published by Springer-Verlag, 1996.

Bogus, P., 1991 — Bogus, P., "Some problems of Frame-based Systems Inconsistency," in *Proceedings of IFIP WG5.4/IFAC Workshop on Dependability of AI Systems (Daisy-91)*, pp. 135-140, 1991.

Bose, R., 1994 — Bose, R., "Strategy for integrating object-oriented and logic programming," *Knowledge-Based Systems*, Vol. 7, No. 2, pp. 66-74, 1994.

Brachman, R. J. & Levesque, H. J. (Eds), 1985 — Brachman, R. J. and Levesque, H. J. (Eds), *Readings in Knowledge Representation*, published by M. Kaufmann Publishers, 1985.

Brown, D. E. & Pomykalski, J., 1991 — Brown, D. E. and Pomykalski, J., "Reliability estimation during prototyping of knowledge-based systems," Institute for Parallel Computation, School of Engineering and Applied Science, University of Virginia Charlottesville, VA, January 11, pp. 1-23, 1991.

Broy, M., 1991	Broy, Manfred, "Methodological Objectives for Formal Description Techniques," In J. Quemada, & Vazquez, M.E., (Eds), *Formal Description Techniques III*, pp. 1-16, 1991.
Buchanan, B. & Feigenbaum, E., 1978	Buchanan, B. and Feigenbaum, E., "DENDRAL and Meta-DENDRAL: Their Applications Dimension," *Artificial Intelligence*, Vol. 11, 1978.
Buckley, J. J. et al., 1986	Buckley, J. J., Siler, W. and Tucker, D., "A Fuzzy Expert System," *Fuzzy Sets and Systems*, Vol. 20, pp. 1-16, 1986.
Bundy, A., 1997	Bundy, Alan, *Artificial Intelligence Techniques*, ed. by Bundy, A., 4th Edition, Springer Verlag, p. 43, 1997.
Canamero, D. et al., 1995	Canamero, D., Geldof, S. and Mcintyre, A., "Coupling Modeling and Validation in COMMET," *Expert Systems with Applications*, Vol. 8, No. 3, pp. 359-369, 1995.
Cao, T., & Sanderson, A. C., 1995	Cao, T. and Sanderson, A. C., "Task Sequence Planning Using Fuzzy Petri Nets," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 25, No. 5, pp. 755-768, 1995.
Cardenosa, J., 1995	Cardenosa, J., "VALID: An Environment for Validation of KBS," *Expert Systems with Applications*, Vol. 8, No. 3, pp. 323-331, 1995.
Catsing, J., 1991	Castaing, J., "A New Formalization of Subsumption in Frame-Based Representation Systems," in *Proceedings of International Conference on Knowledge Representation – (KR'91)*, pp. 78-88, 1991.
Chang, C. L. et al., 1990	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, pp. 219-230, 1990.

Charles, E., 1991	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.
Chen, A. P. et al., 1992	Chen, A. P., Hsu, Sheng-Hsurng and Tan, G. L. H., "Using Fuzzy Petri Net in Rule-Based Knowledge Management," in *Proceeding of 4th International HK Computer Society Database Workshop*, pp.77-93, 1992.
Chen, S. M., et al. 1990	Chen, S. M., Ke, J. S. and Chang, J. F., "Knowledge Representation Using Fuzzy Petri Nets". *IEEE Transaction on Knowledge and Data Engineering*, Vol. 2, No. 3, pp. 311-319, 1990.
Chen, Z. & Suen, C. Y., 1994	Chen, Z. and Suen, C. Y., "Measuring the Complexity of Rule-Based Expert System," *Expert Systems with Applications*, Vol. 7, No. 4, pp. 467-481, 1994.
Chouraqui, E. & Dugerdil, P., 1988	Chouraqui, E. and Dugerdil, P., "Conflict solving in a frame-like multiple inheritance system," in *Proceedings of the 8th European Conference on A.I.*, pp. 226-231, 1988.
Christensen, S. & Petrucci, L., 1995	Christensen, S. and Petrucci, L., "Modular State Space Analysis of Coloured Petri Nets," In Giorgio De Michelis, and Michel Diaz (Eds.), *Application and Theory of Petri Nets 1995*, pp. 201-217, 1995.
Coenen, F. & Bench-Capon, T., 1993.	Coenen, F. and Bench-Capon, T. *Maintenance of Knowledge-based Systems.* Academic Press, 1993.
Coenen, F., 1995	Coenen, F., "Advanced binary encoded matrix representation for rule base verification," *Knowledge-Based Systems*, Vol. 8, No. 4, pp. 201-210, 1995.

Cragen, B. J. & Steudel, H. J., 1987	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, pp. 633-648, 1987.
Craigen, D. et al, 1993	Craigen, D., Gerhart, S. and Ralston, T., "On the use of formal methods in industry - an authoritative assessment of the efficacy, utility and applicability of formal methods to systems design and engineering by the analysis of real industrial cases," *Report to the US National Institute of Standards and Technology*, March, 1993.
Craw, S. & Sleeman, D., 1995	Craw, S. & Sleeman, D., "Refinement in Response to Validation," *Expert Systems with Applications*, Vol. 8, No. 3, pp. 343-349, 1995.
Craw, S., 1996	Craw, Susan, "Refinement Complements Verification and Validation," *Int. J. Human-Computer Studies*, Vol. 44, pp. 245-256, 1996.
Cuda, T. V. & Dolan, C. P., 1991	Cuda, T. V. and Dolan, C. P., "Automating the Refinement of Knowledge-Based Systems". *Proceedings ECAI-90*, Stockholm, August 6-10, pp. 167-172, 1991.
Culbert, C. 1994	Culbert Chris, "NASA MMU-FDIR System," Lyndon B. Johnson Space Centre. USA, 1994.
de Kleer, J., 1986	de Kleer, J. "An Assumption-Based TMS". *Artificial Intelligence* 28, pp. 127-162, 1986.
De Michelis, G. & Diza M., 1995	De Michelis, G. and Diza M., (Eds), *Application and Theory of Petri Nets 1995*, published by Springer-Verlag, 1995.
Dori, D. & Tatcher, E., 1994	Dori, D. and Tatcher, E., Selective multiple inheritance. *IEEE Software*. Vol. 11. No. 3, pp. 77-85, 1994.

Duchessi, P. & O'Keefe, R. M., 1995	Duchessi, P. and O'Keefe, R. M., "Understanding Expert Systems Success and Failure," *Expert Systems with Applications*, Vol. 9, No. 2, pp. 123-133, 1995.
Durkin, J., 1994	Durkin, J., *Expert Systems: Design and Development*. Published by Macmillan Publishing Company, 1994.
Evertsz, R. & Motta, E., 1991	Evertsz, R. and Motta, E., "The Abstract Interpretation of Hybrid Rule/Frame-based Systems," In Ardizzone, E. Gaglio, S. & Sorbello F. (eds.) *Trends in artificial intelligence : 2nd Congress of the Italian Association for Artificial Intelligence*, AI/IA, Palermo, Italy, October 29-31, 1991, published by Springer-Verlag, 1991.
Evertsz, R., 1991	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.
Feigenbaum, E. A., 1982	Feigenbaum, Edward A, *Knowledge Engineering in the 1980's*, Dept of Computer Science, Stanford University, Stanford CA, 1982.
Fikes, R. & Kehler, T., 1985	Fikes, R. and Kehler, T., "The role of Frame-Based Representation in Reasoning," *Communications of the ACM*, Vol. 28, No. 9, pp.904-920, September, 1985.
French, S. W. & Hamilton, D., 1994	French, S. W. and Hamilton, D., "A Comprehensive Framework for Knowledge-Base Verification and Validation". *International Journal of Intelligent Systems*. Vol. 9, pp. 809-837, 1994.
Gamble, R. F. & Baughman D. M., 1996	Gamble, R. F. and Baughman D. M., "A methodology to incorporate formal methods in hybrid KBS verification". *International Journal of Human Computer Studies*. Vol. 44, pp. 213-244, 1996.
Gamble, R. F. et al., 1994	Gamble, R. F., Roman G., Ball W. E. and Cunningham H. C., "Applying Formal Verification Methods to Rule-Based Programs". *International Journal of Expert Systems*. Vol. 7, No. 3, pp. 203-239, 1994.

Geissman, J. R. & Schultz R. D., 1988	Geissman, J. R. and Schultz R. D., "Verification and Validation of Expert Systems," *AI Expert*, Vol. 3, No. 2 Feb., pp. 26-33, 1988.
Ginsberg, A., 1988	Ginsberg, A., "Knowledge-base reduction: A new approach to checking knowledge bases for inconsistency and redundancy," In *Proc. 7th National Conference on Artificial Intelligence* (AAAI-88), Vol. 2. pp. 585-589, 1988.
Gold, D. I. & Plant, R. T., 1994	Gold, D. I. and Plant, R. T., "Towards the Formal Specification of an OPS5 Production System Architecture," *International Journal of Intelligent Systems*, Vol. 9, pp. 739-768, 1994.
Graham, J. A. et al., 1993	Graham, J. A., Drakeford, A. C. T. and Turner, C. D., "The verification, validation and testing of object oriented systems," *British Telecom Technology Journal*, Vol. 11, No. 3, pp. 79-88, 1993.
Grzymala-Busse, 1991	Grzymala-Busse, *Managing Uncertainty in Expert Systems*, Kluwer Academic Publishers, 1991.
Gupta, U. G., 1991	Gupta, U. G., *Validating and Verifying Knowledge-based Systems.* IEEE Computer Society Press. 1991
Gupta, U. G., 1993	Gupta, U.G., "Validation and Verification of Knowledge-based Systems: A Survey". *Journal of Applied Intelligence* Vol. 3, pp. 343-363, 1993.
Hors, P. & Russet, M. C., 1995	Hors, P. and Russet, M. C., "Consistency of Structured Knowledge: A Formal Framework Based on Description Logics," *Expert Systems with Applications*, Vol. 8, No. 3, pp. 371-380, 1995.
Huen, H. S. M., 1993	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.

Jacob, R. J. K. & Forscher, J. N., 1991 — Jacob, R. J. K. and Forscher, J. N. "A Software engineering methodology for rule-base systems". *IEEE Transactions on Knowledge and Data Engineering*, Vol. 2. No. 2, pp. 173-189, 1991.

Jang, H. C., 1995 — Jang, H. C., "A Development Framework and Verification Methodologies for Knowledge-Based Systems," *International Journal on Artificial Intelligence Tools*, Vol. 4, Nos. (1&2), pp. 219-256, 1995.

Jensen, K., 1995 — Jensen, K., *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. Vol. 2. Springer-Verlag, 1995.

Jensen, K., 1996 — Jensen, K., *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. Vol. 1. 2nd Ed. Springer-Verlag, 1996.

Jensen, K., 1997 — Jensen, K., *Coloured Petri Nets: Basic Concepts, Analysis Methods and Practical Use*. Vol. 3. Springer-Verlag, 1997.

Kandelin, N. A. & O'Leary, D. E., 1995 — Kandelin, N. A. and O'Leary, D. E., "Verification of Object-Oriented Systems: Domain-Dependent and Domain Independent Approaches," *J. Systems Software*, Vol. 29, pp. 261-269, 1995.

Kang, B. H. et al., 1996 — Kang, B. H., Gametta, W. and Compton, P., "Verification and Validation with ripple down rules," *Int. J. Human Computer Studies*, Vol. 44, pp. 257- 269, 1996.

Kemper, P., 1996 — Kemper, P., "Reachability Analysis Based on Structured Representations," In Jonathan Billington and Wolfgang Reisig (Eds.), *Application and Theory of Petri Nets 1996*, pp. 269-288, 1996.

Kondratyev, A. et al., 1996 — Kondratyev, A., Kishinevsky, M., Taubin, A. and Ten, S., "A Structural Approach for the Analysis of Petri Nets by Reduced Unfoldings," In Jonathan Billington and

Wolfgang Reisig (Eds.), *Application and Theory of Petri Nets 1996*, pp. 346-365, 1996.

Laita, L. M. et al., 1994 — Laita, L. M., Couto, J., de Ledesma, L. and Margarit A. F., "A Formal Model for Knowledge-Based Systems Verification," *International Journal of Intelligent Systems*, Vol. 9, pp. 769-786, 1994.

Laita, L. M. et al., 1995 — Laita, L. M., Ramirez, B., de Ledesma, L. and Riscos, A., "A Formal Model for Verification of Dynamic Consistency of KBSs," *Computers and Mathematics Applications*, Vol. 29, No. 5, pp. 81-96, 1995.

Landauer, C., 1990 — Landauer, C. "Correctness Principles for Rule-Based Expert Systems". *Expert Systems with Applications*, Vol. 1, No. 3, pp. 291-317, 1990.

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

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

Lee, K. M. & Lee-Kwang, H., 1995 — Lee, K. M. and Lee-Kwang, H., "Fuzzy Information Processing for Expert Systems," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, Vol. 3, No. 1, pp. 93-109, 1995.

Lee, S. & O'Keefe, R. M., 1993 — Lee, S. and O'Keefe, R. M., "Subsumption Anomalies in Hybrid Knowledge Bases". *International Journal of Expert Systems*. Vol. 6, No. 3, 299-320, 1993.

Lee, S. & O'Keefe, R. M., 1994 — Lee, S. and O'Keefe, R. M., "Developing a Strategy for Expert System Verification and Validation," *IEEE Transactions on System, Man and Cybernetics*, Vol. 24, No. 4, pp. 643-655, 1994.

Li, X. et al., 1993	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.
Liebowitz, J., 1991	Liebowitz, J. *Operational Expert System Applications in the United States*, Published by Pergamon Press, 1991.
Lin, C. et al., 1993	Lin, C., Chaudhury, A., Whinston, A. B. and Marinescu, D. C., "Logical Inference of Horn Clauses in Petri Net Models," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 5, No. 3, pp. 416-425, 1993.
Liu, N. K. & Dillon, T. S., 1995	Liu, N. K., and Dillon, T. S., "Formal Description and Verification of Production Systems," *International Journal of Intelligent Systems*, Vol. 10, pp. 399-442, 1995.
Liu, N. K., 1991	Liu, N. K., "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	Liu, N. K., "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	Liu, N. K., "Formal Verification of Some Potential Contradictions in Knowledge Base Using a High Level Net Approach," *Applied Intelligence* 6, pp. 325-343, 1996.

Liu, N.K. & Dillon, T. S., 1991	Liu, N. K. and Dillon, T. S. "An Approach Towards the Verification of Expert Systems Using Numerical Petri Nets". *International Journal of Intelligent Systems*, Vol. 6, pp. 255-276, 1991.
Loiseau, S. & Rousset, M. C., 1993	Loiseau, S. and Rousset, M. C., "Formal Verification of Knowledge Bases Focused on Consistency: Two Experiments Based on *ATMS Techniques," International* Journal of Expert Systems, Vol. 6, No. 3, pp. 273-298, 1993.
Loiseau, S., 1994	Loiseau, S., "A method for checking and restoring the consistency of knowledge bases," *Int. J. Human Computer Studies*, Vol. 40, pp. 425-442, 1994.
Long, J. A. & Neale, I. M., 1993	Long, J. A. and Neale, I. M., "Using Paper Models in Validation, Verification & Testing," *International Journal of Expert Systems*, Vol. 6, No. 3, pp. 383-400, 1993.
Looney, C. G., 1988	Looney, C. G., "Fuzzy Petri Nets for Rule-Based Decisionmaking," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 18, No. 1, pp. 178-183, 1988.
Looney, C. G., 1994	Looney, C. G., "Fuzzy Petri Nets and Applications," *Fuzzy Reasoning in Information, Decision and Control Systems*, Kluwer Academic, pp. 511-527, 1994.
Lounis, H., 1995	Lounis, H., "Knowledge-Based Systems Verification: A Machine Learning-Based Approach," Expert Systems with Applications, Vol. 8, No. 3, pp. 381-389, 1995.
Lucas, P. & Van Der Gaag, L., 1991	Lucas, Peter and Van Der Gaag, Linda, *Principle of Expert systems*, published by Addison Wesley 1991.
Lunardhi, A. D. & Passino, K. M., 1995	Lunardhi, A. D. and Passino, K. M., "Verification of Qualitative Properties of Rule-Based Expert Systems," *Applied Artificial Intelligence*, Vol. 9. Pp. 587-621, 1995.

Marsan, M. A., 1993 — Marsan, M. A., (Eds), *Application and Theory of Petri Nets 1993*, published by Springer-Verlag, 1993.

Matsumoto, K. et al., 1991 — Matsumoto, K., Takano, T. and Sakaguchi, T., "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. & Wild, C., 1995 — Mehrotra, M. and Wild, C., "Analyzing Knowledge-Based Systems with Multiviewpoint Clustering Analysis," *J. Systems Software*, Vol. 29, pp. 235-249, 1995.

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

Meseguer, P. & Verdaguer, A., 1993 — Meseguer, P. and Verdaguer, A., "Verification of Multi-Level Rule-Based Expert Systems: Theory and Practice," *International Journal of Expert Systems*, Vol. 6, No. 2, pp. 163-192, 1993.

Meseguer, P., 1990 — Meseguer, Pedro, "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, 1990.

Meseguer, P., 1994 — Meseguer Pedro, "The VALID Project: Goal, Development, and Results," *International Journal of Intelligent Systems*, Vol. 9, pp. 867-892, 1994.

Morell, L. J., 1988 — Morell, L. J., "Use of Metaknowledge in the Verification of Knowledge-based Systems," in *Proceedings of 1st Int. Conference of Industrial Engineering Application of AI & ES*, pp.847-857, 1988.

Murata, T., 1991 — Murata, T., "A Petri Net Model for Reasoning in the Presence of Inconsistency," *IEEE Transactions on Knowledge and Data Engineering*, Vol. 3, No. 3, pp. 281-292, 1991.

Murrell, S. & Plant, R., 1995	Murrell, S. and Plant, R., "Formal Semantics for Rule-Based Systems," *J. Systems Software*, Vol. 29, pp. 251-259, 1995.
Murrell, S. & Plant, R., 1996	Murrell, S. and Plant, R., "On the Validation and Verification of Production Systems: a graph reduction approach". *International Journal of Human Computer Studies*. Vol. 44, pp. 127-144, 1996.
Nazareth, D. L. & Kennedy, M. H., 1993	Nazareth, D. L. and Kennedy, M. H., "Knowledge-Based System Verification, Validation, and Testing: The Evolution of a Discipline," *International Journal of Expert Systems*, Vol. 6, No. 2, pp. 143-162, 1993.
Nazareth, D. L., 1993	Nazareth, D. L., "*Investigating the Applicability of Petri Nets for Rule-Based System Verification*". IEEE Transactions on Knowledge and Data Engineering. Vol. 4, No. 3, pp. 402-415, 1993.
Newell, A., 1990	Newell, A., *Unified Theories of Cognition*, published by Harvard University Press, 1990.
Nguyen T. A., et al., 1985	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.
Nguyen T. A., et al., 1987	Nguyen T. A., Perkins, W. A., Laffey, T. J. and Pecora, D. 1987, "Knowledge base verification," *AI magazine*, Vol. 8. No. 2, pp. 69-75, 1987.
O'Keefe, R. E. & O'Leary, D. E., 1993	O'Keefe, R. E. and O'Leary, D. E., "Expert System Verification and Validation: A survey and tutorial". *Artificial Intelligence Review*. Vol. 7, pp. 3-42, 1993.
O'Leary, D. E. & Pincus, K. V., 1993	O'Leary, D. E. and Pincus, K. V., "Models of Consensus for Validation of Expert Systems," *International Journal of Expert Systems*, Vol. 6, No. 2, pp. 237-249, 1993.
O'Leary, D. E., 1988	O'Leary, D. E., "Methods of Validating Expert Systems". *Interfaces* Vol. 18. No.6, pp. 72-79, 1988.

O'Leary, D. E., 1991	O'Leary, D. E., 1991, "Measuring and Managing Complexity in Knowledge-Based Systems: A Network and Mathematical Programming Approach." In Brown, D.E. and White, C.C. (Eds), *Operations Research and Artificial Intelligence: The Integration of Problem-Solving Strategies*, 1991.
O'Leary, D. E., 1993	O'Leary, D. E., "Verification and Validation of Case-Based Systems," *Expert Systems with Applications*, Vol. 6, pp. 57-66, 1993.
O'Leary, D. E., 1994	O'Leary, D. E., "Toward a Theory of Verification and Validation: Artifacts," *International Journal of Intelligent Systems*, Vol. 9, pp. 853-866, 1994.
O'Leary, D. E., 1995	O'Leary Daniel E., "The impact of semantic ambiguity on Bayesian weights," *European Journal of Operational Research*, Vol. 84, pp. 163-169, 1995.
O'Leary, D. E., 1996	O'Leary, D. E., "The relationship between errors and size in knowledge-based systems," *Int. J. Human Computer Studies*, Vol. 44, pp. 171-185, 1996.
O'Neal, M. B. & Edwards, Jr. W. R., 1994	O'Neal, M. B. and Edwards, Jr. W. R., "Complexity Measures for Rule-Based Programs," IEEE Transactions on Knowledge and Data Engineering, Vol. 6, No. 5, pp. 669-680, 1994.
Prakash, G. R. & Mahabala, H. N., 1993	Prakash, G. R. and Mahabala, H. N., "SVEPOA: A Tool to Aid Verification and Validation of OPS5-based AI Applications," *International Journal of Expert Systems*, Vol. 6, No. 2, pp. 193-236, 1993.
Preece, A. D. & Shinghal, R., 1991a	Preece, A. D. and Shinghal, R., "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	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, pp. 971-979, 1991.
Preece, A. D., 1989	Preece, A. D., "Verification of rule-based expert systems in wide domains," in *Research and Development in Expert Systems VI* (Proc. Expert Systems 89), N. Shadbolt, Ed., Cambridge University Press, pp. 66-77, 1989.
Preece, A. D., 1991	Preece, A. D. "Practical issues in specifying expert systems," *Intelligent Systems Review*, Vol. 2, No.3/4, pp. 3-26,1991.
Preece, A. D., 1995	Preece, A. D., "Toward a Quality Assessment Framework for Knowledge-Based Systems," *Systems Software*, No. 29, pp. 219-234, 1995.
Preece, A. D., et al., 1996	Preece, A. D., Grossner, C. and Radhakrishnan, T., "Validating dynamic properties of rule-based systems," *Int. Journal Human-Computer Studies*, Vol. 44, pp. 145-169, 1996.
Prerau, D. S., et al., 1993	Prerau, D. S., Papp, W. L., Bhatnagar, R. and Weintraub, M., "Verification and Validation of Expert Systems: Experience with Four Diverse Systems," *International Journal of Expert Systems*, Vol. 6, No. 2, pp. 251-269, 1993.
Reimer, U. & Schek, H. J., 1989	Reimer, U. and Schek, H. J., "A frame-based knowledge representation model and its mapping to nested relations," *Data & Knowledge Engineering*, Vol. 4, pp. 321-352, 1989.
Renard, F. X. et al., 1993	Renard, F. X., Sterling, L. and Brosilow, C., "Knowledge Verification in Expert Systems Combining Declarative and Procedural Representations," *Computers & Chemical Engineering*, Vol. 17, No. 11, pp. 1067-1090, 1993.

Rouge, A. et al., 1995	Rouge, A., Lapicque, J. Y., Brossier, F. and Lozinguez, Y., "Validation and Verification of KADS Data and Domain Knowledge," *Expert Systems with Applications*, Vol. 8, No. 3, pp. 333-341, 1995.
Rousset, M. C., 1988	Rousset, M. C., "On the consistency of knowledge bases: The COVADIS system," *Computation Intelligence*, Vol. 4, pp. 166-170, 1988.
Russell, S. & Norvig, P., 1995	Russell, S. and Norvig, P., *Artificial Intelligence: A Modern Approach*, Prentice Hall, 1995.
Scarpelli, H. & Gomide, F., 1994a	Scarpelli, H. and Gomide, F., "A high level net approach for discovering potential inconsistencies in fuzzy knowledge bases," *Fuzzy Sets and Systems*, Vol. 64 pp. 175-193, 1994.
Scarpelli, H. & Gomide, F., 1994b	Scarpelli, H. and Gomide, F., "Consistency Checking based on High level fuzzy petri nets," in *Proceedings of 3rd IEEE conference on Fuzzy Systems*, Vol. 3. pp. 1957-1962, 1994.
Shiu, S. C. K. et al., (to appear)	Shiu, S. C. K., Liu, J. N. K. and Yeung, D. S., "Formal Description and Verification of Hybrid Rule/Frame-based Expert Systems," to appear in *Expert Systems with Applications*.
Shiu, S. C. K. et al., 1995a	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 8th International Conference on Industrial and Engineering Applications of AI and ES*. Melbourne, Australia, pp. 525-532, 1995.

Shiu, S. C. K. et al., 1995b — 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 System Man and Cybernetics*. Vancouver, pp. 2257-2262, 1995.

Shiu, S. C. K. et al., 1996a — Shiu, S. C. K., Liu, J. N. K. and Yeung, D. S., "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, 1996.

Shiu, S. C. K. et al., 1996b — Shiu, S. C. K., Liu, J. N. K. and Yeung, D. S., "Detection of Anomalies of Hybrid Rule/Frame-based Expert Systems Using Coloured Petri Nets," Australian Journal of Intelligent Information Processing Systems, Vol. 3, No. 3, pp. 59-76, Spring, 1996.

Shiu, S. C. K. et al., 1997 — Shiu, S. C. K., Liu, J. N. K. and Yeung, D. S., "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, 1997.

Shortliffe, E. H., 1976 — Shortliffe, E. H., *Computer-Based Medical Consultations: MYCIN*, New York: Elsevier, 1976.

Someren, M., 1997. — Someren, Maarten van, *Artificial Intelligence Techniques*, ed. by Bundy, A., 4th Edition, Springer Verlag, p. 262, 1997.

Sowa, J. F., 1984 — Sowa, J. F., *Conceptual Structures: Information Processing in Mind and Machine*, published by Addison-Wesley Publishing Company, 1984.

Srinivasan, P. & — Srinivasan, P. and Gracanin, D., "Approximate

Gracanin, D., 1993	Reasoning with Fuzzy Petri Nets," in *Proceedings IEEE International Conference on Fuzzy Systems*, CA, USA, pp. 396-401, 1993.
Stachowitz, R. A. & Chang, C. L., 1988	Stachowitz, R. A. and Chang, C. L. "Verification and Validation of Expert Systems". *Tutorial note at AAAI-88*, 1988.
Stachowitz, R. A. & Combs, J. B., 1987	Stachowitz, R. A. and Combs, J. B., "Validation of Expert Systems". *Proceedings of the 20th Annual Hawaii International conference on Systems Sciences*, pp. 686-695, 1987.
Suen, C. Y. & Chinghal, R., 1991	Suen, C. Y. and Chinghal, R. *Operational Expert System Applications in Canada*, Published by Pergamon Press, 1991.
Suwa, M. et al., 1982	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.
Taylor, R. N., 1984	Taylor, R. N. *Behavioural Decision Making*, Scott Foresman and Company, 1984.
Terano, T., 1994	Terano, Takao, "The JIPDEC Checklist-based Guideline for Expert System Evaluation," *International Journal of Intelligent Systems*, Vol. 9, pp. 893-952, 1994.
Twine, S., 1989	Twine, S., "Mapping between a NIAM conceptual schema and KEE frames," *Data & Knowledge Engineering*, Vol. 4, pp. 125-155, 1989.
Uma, G. & Prasad, B. E., 1993	Uma, G. and Prasad, B. E., "Reachability Trees for Petri Nets: a Heuristic Approach," *Knowledge-Based Systems*, Vol. 6, No. 3, pp. 174-177, 1993.
Valette, R., 1994	Valette, R., (Ed), *Application and Theory of Petri Nets 1994*, published by Springer-Verlag, 1994.
Valiente, G., 1993	Valiente, G., "Verification of Knowledge Base Redundancy and Subsumption using Graph Transformations," *International Journal of Expert*

Systems, Vol. 6, No. 3, pp. 341-355, 1993.

Vanthienen, J. & Dries, E., 1994 — Vanthienen, J. and Dries, E., "Illustration of a Decision Table Tool for Specifying and Implementing Knowledge Based Systems," *International Journal on Artificial Intelligence Tools*, Vol. 3, No. 2, pp. 267-288, 1994.

Vanthienen, J., 1991. — Vanthienen, J. "Knowledge Acquisition and Validation Using a Decision Table Engineering Workbench". In Liebowitz, J. (ed), *Expert Systems World Congress Proceedings*, Pergamon Press, Vol. 3, pp. 1861-1868, 1991.

Vicat, C. et al., 1995 — Vicat, C., Brezillon, P. and Nottola, C., "Knowledge Validation in the Building of a Knowledge-Based Systems," *Expert Systems with Applications*, Vol. 8, No. 3, pp. 391-397, 1995.

Vignollet, L. & Lelouche, R., 1993 — Vignollet, L. and Lelouche, R., "Test Case Generation using KBS Strategy," in *Proceedings of the 13th International Conference on AI (IJCAI-93)*, pp. 483-488, 1993.

Vranes, S. & Stanojevic, M., 1995 — Vranes, S. and Stanojevic, M., "Integrating Multiple Paradigms within the Blackboard Framework". *IEEE Transactions on Software Engineering*. Vol. 21, No. 3, 244-262, 1995.

Willis, C. P., 1996 — Willis, C. P., "Analysis of inheritance and multiple inheritance". *Software Engineering Journal*, pp. 215-224, July, 1996.

Wu, C. H & Lee S. J., 1995 — Wu, Chih-hung and Lee, Shie-Jue, "Knowledge Validation with an Enhanced High level Petri Net Model," in *Proceedings of 11th Conference on Artificial Intelligence for Applications*, pp. 126-132, 1995.

Wu, C. H., et al., — Wu, C. H., Lee, S. J. and Chou, H. S., "Dependency Analysis for Knowledge Validation in Rule-based Expert Systems," in *Proceedings of the 10th conference*

on AI for Applications, pp. 327-333, 1994.

Xu, Y. et al., 1991 — Xu, Y., Paul, R.P., and Shum, H.Y., "Fuzzy Control of Robot and Compliant Wrist System," *Conference Record of the 1991 IEEE Industry Applications Society Annual Meeting*, pp. 1431-1437, 1991.

Yao, Y., 1994 — Yao, Y., "A Petri Net Model for Temporal Knowledge Representation and Reasoning," *IEEE Transactions on SMC*, Vol. 24. No.9. pp. 1374-1382, 1994.

Yeung, D. S. & Tsang, E. C. C., 1994 — Yeung, D. S. and Tsang, E. C. C., "Fuzzy Knowledge Representation and Reasoning Using Petri Nets," *Expert Systems With Applications*, Vol. 7, No. 2, pp. 281-289, 1994.

Yeung, D. S. et al., 1993 — Yeung, D. S., Lee, J. W. T. and Tsang, E. C. C. "A New Fuzzy Reasoning Algorithm for Fuzzy Expert System," In *Proceedings of 1994 Korean/Japan Conference on Expert Systems*, pp. 115-118, 1993.

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

Zhang, D. & Nguyen D., 1994 — Zhang, D. and Nguyen D., "PREPARE: A Tool for Knowledge Base Verification". *IEEE Transactions on Knowledge and Data Engineering*. Vol. 6, No. 6, December, pp. 983-989, 1994.

Zheng, Z. & Li, W., 1992 — Zheng, Z. and Li, W., "A Hybrid Knowledge Engineering Development Environment (KEDE)," *International Journal of Artificial Intelligence Tools*, Vol. 1, No. 4, pp. 463-502, 1992.

Zlatareva, N. & Preece, A., 1994 — Zlatareva, N. and Preece, A., "An Effective Logical Framework for Knowledge-Based Systems Verification," *International Journal of Expert Systems*, Vol. 7, No. 3, pp. 239-260, 1994.

Zlatareva, N. P., 1992 — Zlatareva, N. P., "Truth Maintenance Systems and Their Application for Verifying Expert System Knowledge Bases," *Artificial Intelligence Review*, Vol. 6, No. 1, pp.

67-110, 1992.

Zlatareva, N. P., 1994 Zlatareva, N. P., "A Framework for Verification, Validation, and Refinement of Knowledge Bases: The VVR System," International Journal of Intelligent Systems, Vol. 9, pp. 703-737, 1994.