INFORMATION TO USERS

The most advanced technology has been used to photograph and reproduce this manuscript from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

The quality of this reproduction is dependent upon the quality of the copy submitted. Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps. Each original is also photographed in one exposure and is included in reduced form at the back of the book. These are also available as one exposure on a standard 35mm slide or as a 17" x 23" black and white photographic print for an additional charge.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

U·M·I

University Microfilms International
A Bell & Howell Information Company
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA
313/761-4700 800/521-0600

Order Number 9000671

Petri net applications in the design and analysis of expert systems

Bassen, Gordon Stuart, Ph.D.

City University of New York, 1989

U·M·I
300 N. Zeeb Rd.
Ann Arbor, MI 48106

A

PETRI NET APPLICATIONS IN THE DESIGN AND ANALYSIS OF EXPERT SYSTEMS

by

GORDON STUART BASSEN

A dissertation submitted to the Graduate Faculty in Computer Science in partial fulfillment of the requirements for the degree of Doctor of Philosophy, The City University of New York.

1989

This manuscript has been read and accepted for the Graduate Faculty in Computer Science in satisfaction of the dissertation requirement for the degree of Doctor of Philosophy.

4/29/89
Date

Ken McAloon
Chair of Examining Committee

4/29/89
Date

TC Wendkagn
Executive Officer

Professor Ken McAloon
Professor Michael Anshel
Professor Mel Fitting
Professor Keith Harrow
Professor Stanley Rabinowitz

Supervisory Committee

The City University of New York

Abstract

Petri Net Applications in the Design and

Analysis of Expert Systems

by

Gordon Bassen

Adviser: Professor Ken McAloon

This work presents the concept and implementation of a Rule-Based Petri Net (RBPN) class, as an addition to the family of Petri Net (PN) model extensions. We demonstrate that these RBPN's are capable of modeling a generic type of expert system shell. We further show how they may function as a tool in the design of expert systems; we also discuss their usefulness in the analysis of a developing expert system. As a design tool, we present RBPN concepts in the development of Production Rule systems. We also indicate the added RBPN contributions toward Conflict Resolution techniques for all types of Rule-Based systems. Before describing our RBPN, we discuss several of the extensions to the basic PN model that were precursors to the formal development of the RBPN.

Our RBPN has been developed with the following characteristics: two transition types, the first representing a rule statement, and

the second a query transition for determining qualifier existence; two place types, one containing standard PN-type (simple) tokens, and the second place type allowing for data to be represented as bags of information. These data places are further refined to allow for choice, qualifier (both fixed and variable), and conditional place types.

· In addition, data places do not have their values exhausted by transition firings, as is the usual case with simple place types in basic Petri Net models. This is done to maintain the attribute value(s) in a qualifier place unless the values are otherwise modified as a result of a transition firing.

We then illustrate the use of our RBPN for modeling a small expert system for mathematics remediation placement. This system allows us to show the capabilities of the RBPN as an aid in the design and development of expert systems. The small remediation system model is followed by a larger, more extensive expert system for student advisement of Data Processing majors at Kingsborough Community College. This system includes advisement for both the major area courses in Data Processing and the general academic requirements. In the future we hope to extend the system for advisement in all areas of study.

Preface

In this thesis, we apply Petri Net technology to the design, development, testing and implementation stages of an expert systems project. To this end we have developed a Rule-Based Petri Net (RBPN), a new Petri Net extension.

In Chapter One we present the Basic Petri Net definitions along with descriptions of several extensions to the basic model. Additionally, we give an overview of the use of expert systems, as well as a brief narrative on the organization of knowledge within these systems.

In Chapter Two, we informally present a small expert system for student advisement in mathematics remediation at Kingsborough Community College CUNY. We use this small system as an example to illustrate how the rules, relations, variables, and goals of an expert system can be modelled by the objects of our RBPN. This small system is given in both a generic expert system shell format, and in our RBPN form. We also present the RBPN Graphs of the small system and the corresponding Reachability Graph.

In Chapter Three the formal definitions of the RBPN are presented along with the definitions of markings and transition firings for these RBPN's. The discussion begins with examples of different RBPN term usages in the transition rules. We give a

description of the new place and transition types found in RBPN's along with corresponding examples. Both the Preconditions and Postconditions related to transition firings are illustrated with examples.

In Chapter Four we present the data necessary for the knowledge engineer to construct an expert system for student advisement. The manual process of advisement is described from its beginning with the pre-registration procedures through the actual registration process. The data that will be used to develop a knowledge base is also presented here, as well as in Appendix C. The chapter concludes with a discussion of the creation of the Computerized Expert System from initial analysis of the application through testing of the prototype system.

In Chapter Five we discuss the extensions made to our basic Rule-Based Petri Net model. These extensions allow for the capability of modeling Production Systems. Also discussed are the contributions of RBPN's to the problems encountered in Conflict Resolution. We present several new techniques which we used in our system to aid in evaluating transition firing precedence.

Chapters Six and Seven address the results of our prototype system. The Prototype Rule Set is given in Appendix B. In Chapter Six we discuss the techniques used in the validation of an expert system. In Chapter Seven a comparison of the results generated by

both the human expert and the automated expert model are examined. Details of both sets of output are presented in Appendix A. Appendix B contains the Rule Set we used in both Explicit form and in RBPN format.

Acknowledgements

Anyone who writes a dissertation is indebted to many persons. The completion of my dissertation is no exception. It could not have been written without the faith and support of my family, teachers, colleagues, and friends.

My thanks go to Professor Frank Beckman who granted me a fellowship that allowed me to begin my graduate studies. Professor Jacob Rootenberg of Queens College was a motivating force both in and out of the classroom.

I wish to thank my colleagues at Kingsborough, especially Provost Michael Zibrin and Professor Samuel Gale for making accomodations in my teaching schedule, and to Professor Philip Greenberg for introducing me to the EXSYS expert system shell. I would also like to thank those students who participated in the test cases of our prototype system.

I am indebted to the Golda Meir Library at the University of Wisconsin-Milwaukee and the Science Library at Marquette University for allowing me access to their resources.

I am very grateful to the members of my dissertation committee for their critically constructive comments on the manuscript as it developed. Individually I want to thank Professor Mike Anshel who introduced me to the topic of Petri Nets and their

applications, Professor Keith Harrow for his unbounded efforts in both reviewing the manuscript and seeing it through to fruition, and to Professors Mel Fitting and Stanley Rabinowitz for agreeing to serve on the committee and bringing their expertise to bear on several fine points which appear in the final manuscript.

A special thanks goes to my mentor, Professor Ken McAloon, for his guidance, patience, help, and encouragement through the entire project. His direction during the development and writing of this dissertation was invaluable.

Doctoral studies can be an extremely trying experience. During the most difficult of these times one feels a kinship only with those undergoing similar feelings of angst. Dipak Basu and Stephen Lucci are two comrades from that region. They were always there to assist, argue, challenge, or provide feedback on new ideas. Their friendship is greatly valued.

Thanks go to my family for stimulating me to ask questions and try to learn the answers. I thank my parents, who have been a constant source of encouragement. My brother could be counted on to remind me that he saw the light ahead even when I didn't.

Finally, none of this would have been possible without the love, companionship, and support of my wife, Mary. My gratitude to her is immeasureable.

Contents

Figures and Tables

> An object **A** is a model of an object **B** for an observer **C** if the observer can use **A** to answer questions that interest him about **B**.
>
> M. Minsky (1965)

Chapter 1 Petri Net Models and Expert Systems

1.1 Introduction

In recent years the design, development, and implementation of expert systems have developed as rapidly as the increase in the amount of information that can be stored on a chip. Expert systems are now available for microcomputers as well as minicomputer and mainframe architectures. There are currently expert systems which can be used to help create expert systems (e.g., EXSYS, an expert system shell developed by the EXSYS Corporation).

These expert systems make use of a data base of facts about some problem domain (e.g., medicine, geology, chemistry, etc.), along with assertions or inference rules defining relationships between these facts and specifying how new facts may be deduced by means of the assertions. Together the facts and rules form a knowledge base, and the expert systems

use this knowledge base in a search for solutions to given problems. New knowledge is obtained via some particular control strategy that is defined separately and independently from the base level (categorical) knowledge of the problem domain. These Production Systems (or Rule-Based Systems) are highly computation intensive. The slow execution speed of these systems has prevented their use in areas requiring high performance or real-time responses. This need for faster production system execution has led to developments in special purpose hardware architectures [12]. Production system languages have also evolved from the algorithms used for matching production (if...then...) rules with the knowledge base assertions [8, 19, 35]. The activity of most complex systems is generally described by a set of heuristic algorithms, and the solving of a problem or the reaching of a decision is typically determined by a search over all possible options.

A particularly useful tool in modeling the behavior of concurrent processes is the Petri Net. These nets and their extensions have been used in modeling Data Base Concurrency, Robots, Operating Systems, Digital Parallel Processing Systems, Computer Systems, and other highly complex activities [2, 4, 6, 10]. They have been used successfully in situations where system limitations, error detections, and degrees of parallelism have played a role in their choice as a model. Over the last 25

years, many different models of parallel computation have been developed. The differences between these systems may have been only superficial, since many papers have appeared comparing the various models and showing relationships between them [16, 17]. Results obtained from one type of system are often shown to be comparable to a result in another system. For example, Vector Addition Systems (VAS) are in a sense equivalent to Petri Nets, and Vector Replacement Systems (VRS) can be simulated by VAS [13].

In this thesis we will show that Petri Nets can be used to model Rule-Based Systems, and that extensions of these nets can be used to model a variety of complex processes related to the design and analysis of expert systems.

Structure of a Rule-Based System

In general, rule-based systems can be viewed as having the following components:

1) a set of rules of the form :

   IF < condition > THEN < actions > ;

2) a data base against which the rule conditions are tested, and which is altered by the execution of the rules' actions;

3) a control or monitoring program (inference engine) that

contains the logic regarding the order in which rules are to be applied, what action should be taken if more than one rule is viable (i.e., has valid conditions) at one time, and so on.

The rule-based system may be situated between the user and other external systems as shown in the following interpretation.

USER <========> RULE-BASED <=====> EXTERNAL INFORMATION
SYSTEM SYSTEM

Questions and Answers | Rules + Data + Monitor | Knowledge Base

A basic characteristic of rule-based systems is that the set of rules indicates how the system should react to a change in the environment without requiring any advance knowledge about the flow of control. In a procedural programming language, the flow of control and data are predetermined by the algorithm written as the code.

The concept of "Programs = Algorithms + Data" for procedural languages may have its counterpart in expert systems as :

" Rules + Knowledge ==> Solution "

In most instances these expert systems use one of two

general control strategies. These are forward chaining and backward chaining. In forward chaining control strategies, the system starts with a set of facts relating to and describing the characteristics of the problem; it then applies productions or logical inferences whenever their conditional parts are evaluated as true by the presently known facts. This strategy forces the program to search forward from the starting point of its knowledge base making inferences as it goes along.

A backward-chaining or goal-driven expert system (e.g., Mycin or Prolog), begins with a specified goal and works backward from the starting position, trying to find a sequence of rules that can be used to infer the truth of the goal from the initial data or working memory elements. Given a proper set of rules, forward chaining systems can perform backward chaining and vice versa.

Several expert systems and languages (including Prospector and EXSYS) allow for facts and rules with corresponding probabilities or certainty factors. These systems use a weighted or probabilistic matching and search strategy. There are expert systems (such as KEE and LOOPS) where the objects used as facts can have more structure. The OPS5 and Prolog systems together represent most of the potential sources and limitations of parallel execution [7].

1.2 Petri Net Models

Since 1962 when C.A. Petri wrote his seminal paper "Communication with Automata" [29], the definition of a Petri Net has been subject to a variety of constraints and modifications. We will define a basic Petri Net in a style common to automata theory. This has also been the format most commonly found in the literature. The basic Petri Net structure is defined by its places, transitions, input function, and output function.

Definition 1.1 A basic Petri Net, N, is a four-tuple consisting of:

$\mathbf{P} = \{p_1, p_2, \ldots, p_n\}$ a finite set of places, $n \geq 0$

$\mathbf{T} = \{t_1, t_2, \ldots, t_m\}$ a finite set of transitions,

$m \geq 0$ such that $\mathbf{P} \cap \mathbf{T} = \emptyset$

$\mathbf{I}: \mathbf{P} \times \mathbf{T} \Longrightarrow \mathcal{N}$ is the input function

$\mathbf{O}: \mathbf{T} \times \mathbf{P} \Longrightarrow \mathcal{N}$ is the output function

The input and output functions map ordered pairs of places and transitions into the natural numbers $\mathcal{N}$, where $\mathcal{N} = \{0, 1, 2, \ldots\}$.

A marking $\mathbf{M}$ of a Petri Net is a mapping $\mathbf{M} : \mathbf{P} \Longrightarrow \mathcal{N}$. Graphically we may represent a marking $\mathbf{M}$ by putting in each place $p_i \in \mathbf{P}$, $\mathbf{M}(p_i)$ dots called tokens. $\mathbf{M}$ maps tokens to each place in

the net. **M** can also be regarded as an *n*-dimensional vector where the i*th* position, $\mathbf{M}_i$, represents the number of tokens that have been assigned by **M** to place p_i. The starting placement of tokens in the net is referred to as the *initial marking* for the net.

A Petri Net with an initial marking $\mathbf{M}_0$ is the Marked Petri Net $\mathbf{C} = (\mathbf{P}, \mathbf{T}, \mathbf{I}, \mathbf{O}, \mathbf{M}_0)$.

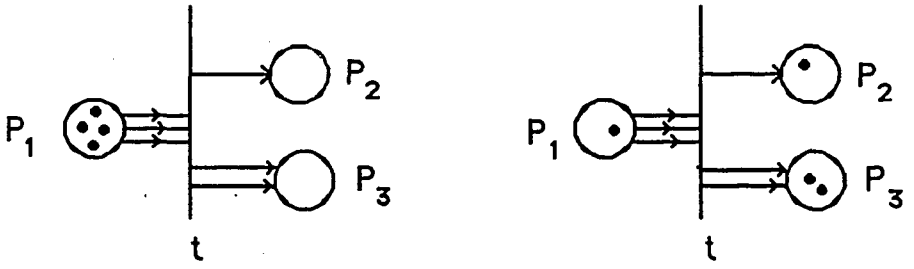
Traditionally, circles have been used to represent places, bars for transitions, and black dots for markings (tokens). When this formalism is used, the structure is usually referred to as a Petri Net Graph (see Figure 1.1a).

In a Petri Net Graph, the input and output functions are commonly represented by directed arcs from the places to the transitions and from the transitions to the places. An arc is directed from a place p_i to a transition t_j if the place is an input of the transition. Similarly, an arc is directed from a transition t_j to a place p_j if the place is an output of the transition.

A transition, t, in a Marked Petri Net $\mathbf{C} = (\mathbf{P}, \mathbf{T}, \mathbf{I}, \mathbf{O})$ with marking **M**, is *enabled* when for all $p_i \in \mathbf{P}$, $\mathbf{M}_i \geq \mathbf{I}(p_i, t)$. Firing an enabled transition t results in the new marking **M'**, where

$$\mathbf{M}_i' = \mathbf{M}_i - \mathbf{I}(p_i, t) + \mathbf{O}(t, p_i)$$

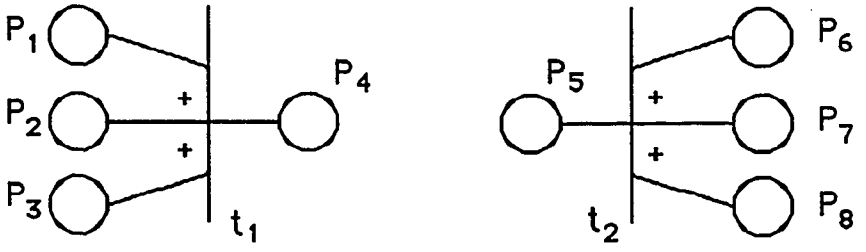
A Petri Net Graph is a directed bipartite multigraph, and its correspondence to the basic Petri Net structure is so natural that,



Initial Marking: P_1 contains 4 tokens
P_2 and P_3 are empty
t is enabled since P_1 contains more than 3 tokens

After t fires
P_2 receives 1 token
P_3 receives 2 tokens

(a)

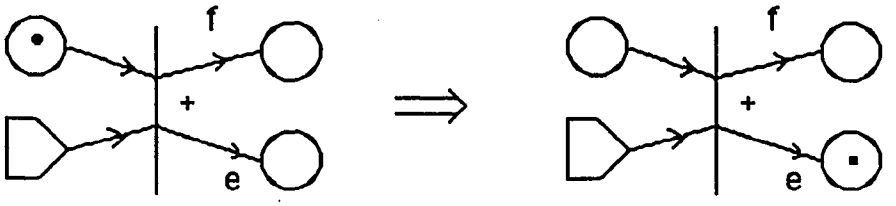


t_1 represents XOR input logic. t_1 can fire if exactly one of P_1, P_2, or P_3 contains a token.
t_2 represents XOR output logic. When t_2 fires exactly one of P_6, P_7, or P_8 will receive a token.

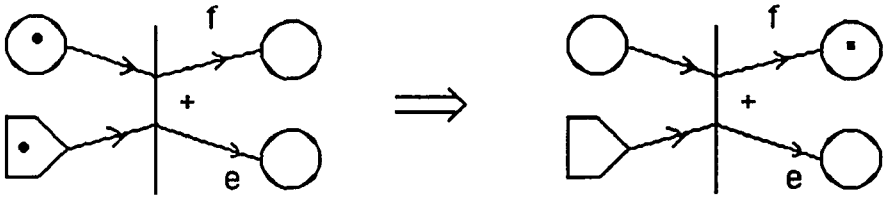
(b)

Petri Net Graphs and Extensions

Figure 1.1 (part 1 of 2)

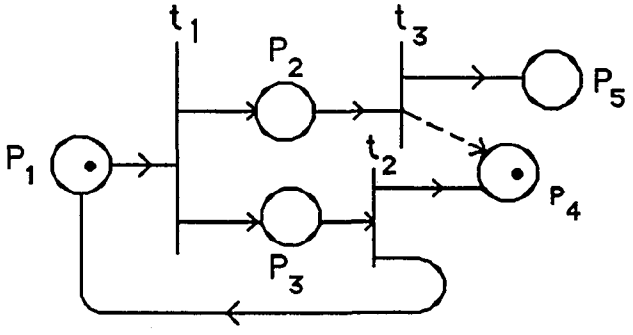


Empty Switch



Full Switch

(c)



When t_3 fires, the token absorber clears P_4 of tokens as P_5 receives a token.

(d)

Petri Net Graphs and Extensions

Figure 1.1 (part 2 of 2)

in most cases, they are considered to be a single concept. The reader who is not familiar with basic Petri Nets or Petri Net Graphs is referred to Peterson's [28] extensive survey of the subject, as well as Reisig's [33] more recent work in the area.

1.2.1 Extended Petri Nets

The nature of any informal graph model (i.e., one with arrows, dots, circles, and lines) is to be modified via another simple graphical structure whenever a user might like to model a slightly different object. Since these designs were formulated with differing intents, changes were made to allow the basic Petri Net to encompass descriptions of additional types of control and data flows.

For example, in one particular case it was appropriate to use extensions of a graph model to allow for the interpretation of data flows [1]. We will discuss the three extensions represented in that case, which were given by Baer and Ellis, that supplement the descriptive power of the basic Petri Net model.

Type 1 Disjunctive Logic

The first extension, called disjunctive logic, assists in representing predicates. A decision or branch is modeled in a Petri Net Graph by a single place input to two or more transitions which compete for the tokens in the place. The conflict between the transitions for the tokens is resolved randomly (see Figure 1.1b). The seemingly arbitrary nature of this conflict is not advantageous in a descriptive model, where a choice usually represents a data dependent decision which is explicitly made in the system, but is not visible in the net as an event which may occur. Through the use of disjunctive logic, we can model the conflict-resolving event as a transition. Typically any decision making system that is given two equally evaluated functions as choices will have to select randomly between the events, when operating under time or other constraints.

Type 2 Switch Logic

The second extension, switch logic, allows the use of a specially designated place. This place, or switch, decides which of the two output places will receive the token(s) when the disjunctive transition has fired (see Figure 1.1c). The switch

is convenient when the decision is based entirely on the sequence of transition firings.

Type 3 Token Absorber

The final extension in this class is a token absorber which is helpful in terminating redundant processing, and for cleaning up the net upon termination. The token absorber is denoted graphically by a dashed arc-line. It functions by removing all the tokens from its output places when the input transition fires. The structure is illustrated in Figure 1.1d. This construct is also referred to as a reset to zero transition.

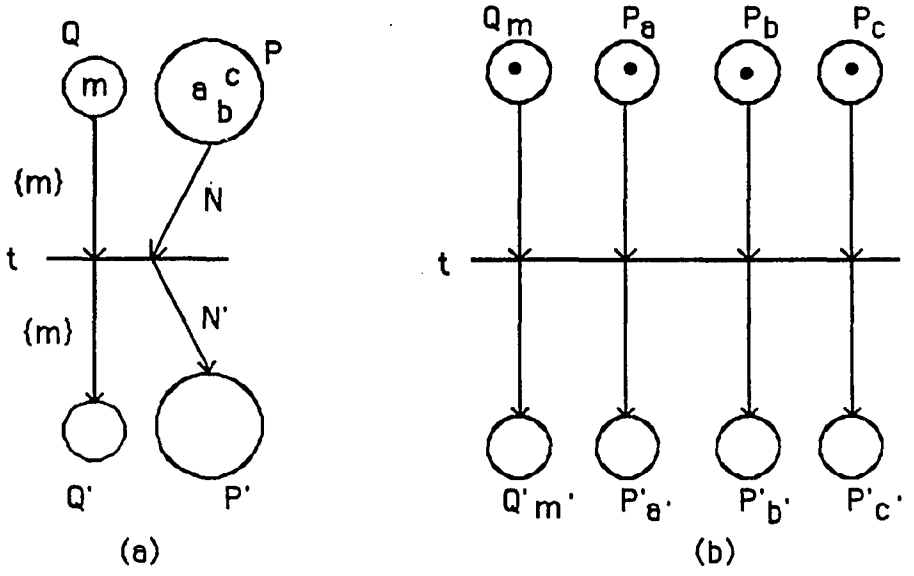
1.2.2 Colored Petri Nets

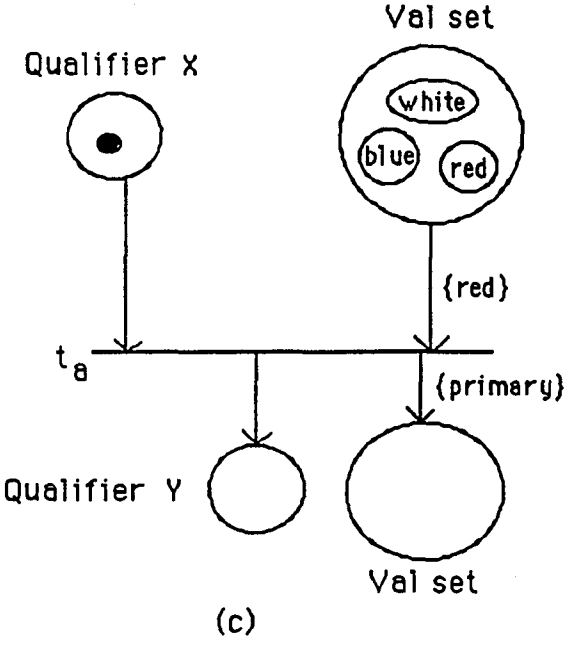
Colored Petri Nets [15] are a slight variation of basic Petri Nets and a particular case of the general model of Predicate/Transition Nets [10]. In this model, the places contain bags of colored tokens. They are connected to the transitions by labeled arcs. The labels are either sets of colors (possibly a singleton) or free variables. The firing rules [2] are extended as follows:

Let (P,T) and (T,P') be input and output arcs, respectively, of transition T. Let N = {a, b, . . . } and N' = {a', b', . . . } be subsets of colors for tokens which can reside in P and P' respectively. If (P,T) is labeled with N, then T can be enabled only if P contains at least one token of each color belonging to N, and the firing will remove one token of each color. If (T,P') is labeled with N', then the firing of T will deposit one token of each color a', b', . . . on P'. A case where N = {a, b, c} can be seen in Figure 1.2a, with Figure 1.2b the equivalent Petri Net construct.

1.2.3 Extended Place / Transition Nets (EPTN's)

This variation of Petri Nets was developed to perform modeling and analysis of concurrency control algorithms, specifically those involved in distributed data base systems. These EPTN's attempt to overcome certain problems associated with basic Petri Nets. These problems include the lack of a time concept (instantaneous transition firings), token interpretation as job time in a system or queue, and the presence of conjunctive (as opposed to disjunctive) logic on the preconditions and postconditions of an event by requiring that all of them are satisfied before and after an event (transition firing) takes place.





Qualifier X : "The color is"

Val settings: (red,blue,grey, white ,black)

t_a : QX(red) => QY(primary)

Upon firing Qualifier Y will receive a token and the QY val set gets the marking primary.

Models of Colored Nets and EPTN's

Figure 1.2

EPTN's modify Petri Nets in token definition, transition definition, and transition firing rules. These extensions draw upon those of Predicate/Transition Nets [10], and Evaluation Nets [24]. The EPTN model allows for two types of tokens; those tokens which can carry data and those which cannot.

Definition 1.2.1 A *Data Token* is a token which has attributes associated with it. The values of these attributes are modified by transitions. A data token **K** with n attributes is indicated by **K**[n], and the ith attribute as **K**(i).

Definition 1.2.2 A *Simple Token* is a token which does not have any associated attributes, and thus cannot carry any data.

The existence of a simple token at a place has the same meaning as a token in a Basic Petri Net, whereas data tokens will represent system inputs and correspond to the attribute tokens of Evaluation Nets. These token definitions are the first extension to the Petri net formalism. In the EPTN model, tokens are identified individually and need to have attributes associated with them. Examples of such attributes in a net where tokens may represent inputs to the system (e.g., jobs, transactions, etc.) are the base-set

size of transaction, the entry time to the system, and transaction origination site. These examples illustrated the necessity for a type of data token to be implemented for EPTN's.

We illustrate this type of structure in Figure 1.2c where the data attributes red, white, and blue are present for val set X. Transition t_a is fired when Qualifier X has a simple token and val set X contains the attribute red. Upon firing Qualifier receives a simple token and val set Y gets the data attribute primary.

Definition 1.2.3 Each transition $t_i \in \mathbf{T}$, where **T** is the set of all transitions, is defined as a 5-tuple,

$t_i = [z(t_i), pr(t_i), q(t_i), r(t_i), s(t_i)]$ where,

z is the transition time for t_i

pr is the precondition that has to be satisfied for t_i to be activated

q is the transition procedure specifying the effects, on the token attributes, of activating transition t_i

r is the output resolution procedure indicating the procedure to be followed in routing the tokens to alternative places in the set of output places of t_i

s is the transaction schema.

We will discuss in detail these extensions.

1) Transition time - associates the firing time of an event with each transition. It may be a constant or it may be a function of the specific token that enabled the transition.

2) Precondition - by allowing for preconditions, cases can be modeled where a place p_i participates in only a subset of the transitions, each of whose set of input places includes p_i. Assume t_i such that $\mathbf{I}(t_i) = \{p_j, p_k\}$. Then the following logic can be specified on the inputs:

1) $pr(t_i) = (\mathbf{M}(p_j) > 0) \wedge (\mathbf{M}(p_k) > 0) = \text{AND(Q)}$

2) $pr(t_i) = (\mathbf{M}(p_j) > 0) \vee (\mathbf{M}(p_k) > 0) = \text{OR(Q)}$

3) $pr(t_i) = [(\mathbf{M}(p_j) > 0) \vee (\mathbf{M}(p_k) > 0)] \wedge$
$\neg [(\mathbf{M}(p_j) > 0) \wedge (\mathbf{M}(p_k) > 0)] = \text{EXOR(Q)}$

where $Q \subseteq \mathbf{I}(t_i)$. The above conditions allow for combinations of AND and OR inputs. $\mathbf{I}^{pr}(t_i) \subseteq \mathbf{I}(t_i)$ is the set of all input places which are selected for the firing of transition t_i.

For example, if $pr(t_i) = \text{AND}$ then $\mathbf{I}^{pr}(t_i) = \mathbf{I}(t_i)$. If however,

$pr(t_i)$ = OR [AND(p_i, p_j), AND(p_m, p_n)] and only $\mathbf{M}(p_i) > 0$ and $\mathbf{M}(p_j) > 0$, then $\mathbf{I}^{pr}(t_i) = \{p_i, p_j\}$. If all four places have a marking greater than zero, then $\mathbf{I}^{pr}(t_i) = \{p_i, p_j\}$ or $\mathbf{I}^{pr}(t_i) = \{p_m, p_n\}$ and the selection is random.

3) Transition Procedure- an important feature of the EPTN model is the capability of tokens to carry data. When used for performance evaluation studies, these tokens have been used to carry performance related data (i.e., a data token represents a transaction that the system receives, and the data relates to an arrival time or the number of messages transmitted, etc.). In collecting data, transitions manipulate the attributes of these tokens, and the transition procedure specifies how each transition manipulates these token attributes [25].

4) Output Resolution Procedure- this specifies the assignment of tokens to the output places, or some subset of places in $\mathbf{O}(t_i)$. $\mathbf{O}(t_i)$ consists of a set of data token places, $\mathbf{O}_d(t_i)$, and a set of simple token places, $\mathbf{O}_s(t_i)$.

Thus we have, $\mathbf{O}(t_i) = \mathbf{O}_d(t_i) \cup \mathbf{O}_s(t_i)$. An output resolution procedure provides a selection logic on the elements of these sets. These procedures are functions of the data carried in data tokens.

The evaluation of $r(t_j)$ will result in the selection of some of the output places of t_j (i.e., $\mathbf{O}(t_j)$) to accept tokens. We will denote this set as $\mathbf{O}^r(t_j)$.

Definition 1.2.4 A transition $t_j = (z, pr, q, r, s)$ of an EPTN is enabled for a marking **M**, if and only if $pr(t_j)$ is true for marking **M**.

Definition 1.2.5 The firing of a transition $t_j \in T$ of an EPTN, TN = (P, T, A), under a marking **M**, in which it is enabled, results in a new marking **M`** defined as :

$$\mathbf{M}`(p_j) = \begin{cases} \mathbf{M}(p_j)+1 \text{ if } p_j \in \mathbf{O}^r(t_j) \wedge p_j \notin \mathbf{I}^{pr}(t_j) \\ \mathbf{M}(p_j)-1 \text{ if } p_j \in \mathbf{I}^{pr}(t_j) \wedge p_j \notin \mathbf{O}^r(t_j) \\ \mathbf{M}(p_j) \text{ otherwise} \end{cases}$$

The firing phase begins at the start of transition t_j firing, denoted $f(t_j)$, and ends at time $f(t_j) + z(t_j)$.

5) Transition Schema - for distributed systems it is necessary to describe two types of transitions. One represents a local action (local-type) at a site that is being modeled, and another

represents communication or transmission of messages (transmission-type) between different sites.

The differentiation of two types of transitions in EPTN's and the corresponding changes to the firing rules are detailed in the following descriptions.

Definition 1.2.6 A token may be in one of three states while it travels in the net:

a) *reserved* - during the firing phase of a transition t_j if it resides at a place $p_i \in I^{pr}(t_j)$.

b) *traveling* - during the firing phase of a transmission-type transition t_j if it resides at a place $p_i \in I^{pr}(t_j)$ and if, at the end of the firing phase, it will reside in a foreign place of t_j.

c) *available* - otherwise.

The use of a time concept in net transition firings is presented in the following definition. In this firing time definition we allow for both internally and externally enabled transitions.

Definition 1.2.7 Let $f(t_j)$ indicate the beginning of the firing phase of transition t_j.

a) If t_j is a local transition, then the change from marking **M** to marking **M`** takes place at time $f(t_j) + z(t_j)$.

b) If t_j is a transmission-type transition, then the change from marking **M** to marking **M`** takes place as one of two cases.

Case 1: For $p_j \in \mathbf{O}^r(t_j)$ and p_j foreign, the change is at time $f(t_j) + z(t_j)$.

Case 2: For other places, it is instantaneous (at time $f(t_j)$).

1.2.4 Prompt Nets

Another property of Petri Nets, that has been studied by P. Thiagarajan and S. Patil [22] along with others, is the concept of promptness. We will designate certain transitions of a Petri net as external transitions, and the remaining transitions as internal. The net is then said to be *prompt* if there can be at most a bounded number of firings of internal transitions between successive

firings of external transitions. This allows us to study a Petri net as some type of machine or organism, and the firing of an external transition can be looked upon as an interaction with the outside world. Promptness requires that there be no more than a bounded number of occurrences of events within the machine or organism between successive interactions with the outside world.

This concept of promptness allows us to state that non-promptness of a net implies uncertainty about the response time and an inability to predict the extent of resource usage by the system that is being represented by the net. Promptness, however, can be interpreted as ensuring that the external world has control over the activities initiated by the system. In the following we present two types of promptness.

Let C be a Petri net with initial marking **M**, whose set of transitions **T**, has been partitioned into two sets, T_e and T_i, external and internal respectively. If there exists some integer $\mathbf{K_S}$, such that starting from any reachable marking **M`**, we can fire internal transitions alone at most $\mathbf{K_S}$ times, then C is said to be *strongly-prompt*. Given a net C, such an *a priori* bound $\mathbf{K_S}$ might not exist. However, corresponding to every reachable marking **M`**, there could be an integer bound $\mathbf{K_W}$, such that, starting from **M`** we can fire internal transitions alone at most $\mathbf{K_W}$ times. In this case, we

will say that the net is *weakly-prompt*. It is evident that if a net is strongly-prompt then it is weakly-prompt as well.

It has been shown that strong-promptness of a Petri net is decidable. In addition, decidability of weak-promptness has been reduced to the open problem of deciding whether a given transition in a net is *hot* in the sense of Keller [17] (i.e., if there exists a firing sequence in which the transition appears an infinite number of times).

1.3 Expert Systems and Rule-Based Languages

Interest about expert systems has grown rapidly in the past few years. The topic has received large amounts of attention in both the press and the technical literature. The Fifth Generation Computer Project currently under way in Japan has created a new wave of popularity and concern in the U.S. and Europe for the development of similar programs. There is a degree of difficulty in defining precisely what makes an expert system. One definition of an expert system given by an authority in the field is "a knowledge-intensive program that solves problems normally requiring human expertise" [7]. In another paper, the same researcher said that these programs "attempt to mimic the manner in which an expert practitioner solves problems in his field" [3].

1.3.1 Preliminaries

Expert systems are usually identified by having the following components:

- A **data base** upon which the system operates.
- A set of **rules** that are used to reach conclusions from the data and perform some action as a result of the conclusion.
- An **inference engine**, that is, a program which selects the rules to be applied.

The rules are usually given in the form "IF xxx THEN yyy", where xxx is some statement about the data which evaluates to true or false, and yyy is a set of actions to be performed if xxx is true. These actions may include, but are not limited to, modifying or adding data to the data base, requesting more data, making recommendations to the user, or running another program.

The inference engine usually works in cycles. In each cycle, it selects a rule whose xxx statement is true, and executes the action(s) yyy. This is commonly referred to as 'firing' the rule. There may be more than one rule whose IF part is valid and therefore eligible to be fired. Many strategies for selecting the rule to be fired have been developed, and they fall into two main categories:

- Forward chaining: Rules continue to be selected until no further change occurs in the data base. A rule may be selected more than once if the data values it tests continue to change. Forward chaining is used to draw all possible conclusions from a given set of facts.

- Backward chaining: a list of goal variables is prescribed. The inference engine attempts to select rules that assign values to these variables. If that does not succeed, rules that determine inputs for the former rules are selected, and so on. Backward chaining is used to achieve specific goals.

One of the important questions in the design of an inference engine is what rules are eligible for testing at any time. The simplest technique of having all rules eligible all the time makes it difficult to control the course of the inference, and causes wasteful testing of irrelevant rules. It is more efficient to limit the inference to a subset of rules that are meaningful relative to what is current knowledge. Any set of rules might be imbedded in a procedural language and run more efficiently than a general purpose inference engine. However, there would not be a separation between the expert knowledge, contained in the rules, and the procedure program, as represented by the inference engine. By creating these two distinct partitions it is easier to modify, append, or

restructure the knowledge base without regard to the usual considerations when making changes to a working program.

We have available "expert system shells" that are used to promote the development of practical expert systems. These packages contain an inference engine, as well as facilities for entering data and rule definitions. Many of these shells also contain utilities for analyzing user sessions, offering trace explanations as to why or how a conclusion was reached by the system. As spreadsheets have become the dominant tool in business applications, so too have shells proliferated for the creation of expert systems. Matching the shell to the problem is now a major consideration of the knowledge engineer in expert system development.

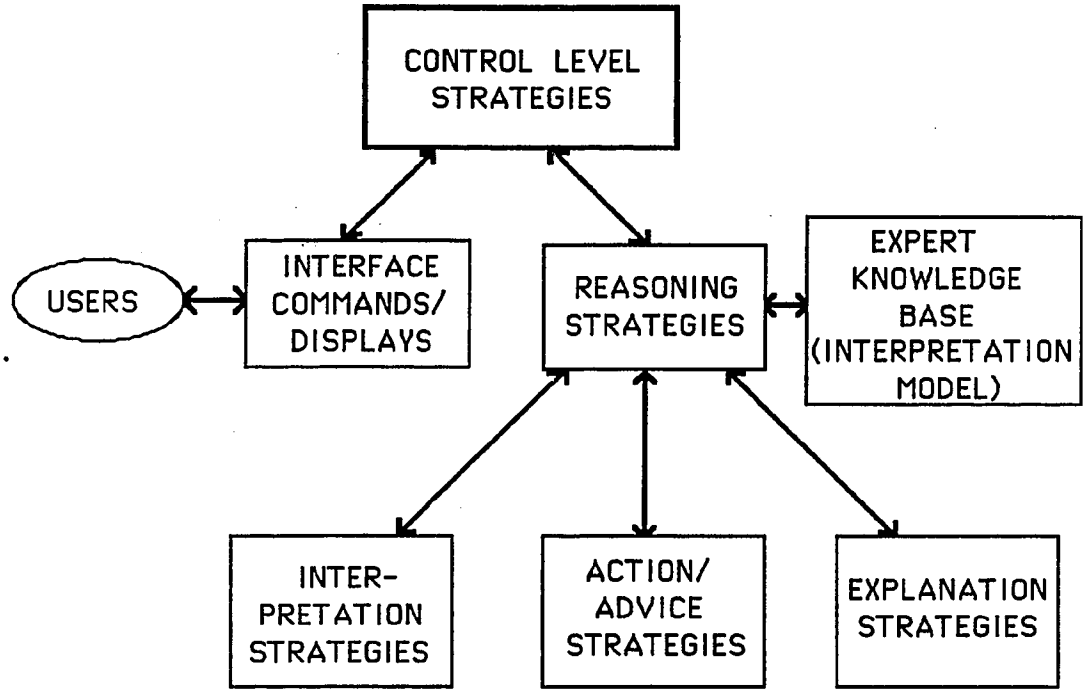
1.3.2 Organization of Knowledge and Reasoning in Expert Systems

Among the most explored areas for the development and implementation of expert systems have been diagnostic or interpretative applications. Based on those results we are able to reach several conclusions regarding the structure of knowledge and the forms of reasoning that are shared by these types of systems [36].

- The advice or choices and conclusions must be generated by the system from a finite set of discrete and previously known collection of elements. For example, all causes of failure must be known.
- The evidence, whether observed facts or data regarding the problem, must be reliably obtained by the user of the system or the system itself.
- The starting assumptions must limit the problem to a highly specific area, thereby directing the reasoning. The availability of a knowledge base or organization of relations and reasoning rules must be able to link the evidence about the problem to the conclusions. These may be logical or probabilistic rule-based associations, or a combination of the two.
- A reasoning control strategy must be designed to guide the reasoning of the system. The resultant output of the system must correspond to a reasonable set of responses to the human users. As a rule the system must respond much the same as an expert, requesting additional information when appropriate, giving reasons for actions when asked, and examining alternative methods when working with limited available data.

Figure 1.3 illustrates the major components of an expert

system and indicates how problem solving can be viewed as a flow of reasoning that goes from evidence to conclusions. The objective is to reach the most appropriate goal for each particular instance. What characterizes the human as an expert is an ability to channel and reduce the number of possible choices, as more useful data is received. Similarly the expert system must have a reasoning model that takes into account differing patterns of data, and extracts from them a preliminary set of feasible results from which the final conclusions are precipitated. The rate at which this funneling process occurs is much faster than that typically encountered in design or game-playing scenarios, where the choice of hypotheses is quite extensive and often difficult to narrow down.



Major Components of an Expert System Showing the Principal Paths of Reasoning

Figure 1.3

Chapter 2 A Model of a Small Expert System for Remediation Advisement

In this chapter, we present an expert system designed to evaluate whether a student needs remedial mathematics, and if so to determine the student's appropriate placement. The function of presenting this system is twofold. First, to exhibit the range of capabilities of our system in a small system environment; and second, to demonstrate the ability to model this type of expert system using the notion of a Rule-Based Petri Net (RBPN).

We will present the qualifiers and their value sets, choice sets, and the formal rules of the system. This will be followed by a trace through the system using the RBPN model presented here.

2.1 Qualifier Val Sets

Qualifier val sets consist of the qualifier predicates along with a finite set of possible attribute value settings. The attribute values may be examined as a conditional part, or modified in an action clause of a rule. In a RBPN, we write Qn[m] to represent Qualifier n, with m the number of unique value settings, and Qn(i) to represent the i*th* setting.

Here are the qualifiers we will use in our first example system given in both textual, as well as explicit Rule-Based Petri Net notation:

	RBPN notation
Q1: Last CUNY math assessment was 1) not taken yet 2) Failed (below 25) 3) Passed (25 or higher)	Q1[3]
Q2: This student needs 1) initial placement 2) no initial placement 3) placement review 4) preplacement testing	Q2[4]
Q3: Last math course was 1) MAT M1 2) MAT M2 3) MAT R2 4) Independent Lab Study 5) MAT 03 or higher 6) None taken yet	Q3[6]
Q4: Last math grade was 1) A 2) B 3) C 4) D 5) F 6) R 7) INC 8) W 9) no grade (lab only)	Q4[9]

2.2 Choice Sets

Choice sets are the possible resultant choices that are available as outcomes or decisions to the conditions as represented in the rules of the system. Every choice also has a corresponding confidence factor value, from 0 through 100, assigned to it when the choice is employed. In RBPN notation C1,90 would signify Choice 1 with a certainty factor of 90 out of 100. A choice may be employed more than once, in which case its confidence factor can be adjusted accordingly. Confidence factors of 0 and 100 are unchangeable. Once a choice place has been given either of these two values it can not be modified any further by the system. A certainty factor is a probabilistic value, used to indicate the degree of belief for a particular choice place. (The terms confidence factor and certainty factor are understood to have the same meaning.)

The choice places we will use in our example are:

		RBPN notation
Choice 1 -	Student needs CUNY assessment exam	C1
Choice 2 -	MAT M1	C2
Choice 3 -	MAT M2	C3
Choice 4 -	MAT R2	C4

Choice 5 -	R3 self-study in Math Lab	C5
Choice 6 -	Follow curriculum requirements of major	C6
Choice 7 -	Inconsistency in answer; restart session	C7

2.3 Relational Val Sets

Relational val sets are the relational expressions that need to be evaluated within the course of rule examination. They may include mathematical terms, set theoretic expressions, or other Boolean conditionals.

Here are the relational val sets that are used in our first example:

Relational Expression	RBPN notation
ICUNY $\geq$ 25	R1
ICUNY < 15	R2
(ICUNY $\geq$ 15) AND (ICUNY < 20)	R3
(ICUNY $\geq$ 20) AND (ICUNY < 25)	R4

2.4 Rule Sets

The Rule Set is the formal collection of all transition rules used by the system. The IF part of a transition rule is formed by the conjunction of appropriately selected qualifier val setting(s) with relational val set(s). The THEN part of the rule is formed by the conjunction of any resultant qualifier val setting(s), and newly evaluated choice places with their corresponding confidence factors.

The RBPN form Qn(-,-,1,-, . . .), when appearing in an IF clause, indicates that Qualifier n must have a valid third value setting to enable the rule. When this qualifier term is found in a resultant clause, it indicates that Qualifier n will have its third value setting established as a consequent of the rule enabling. When used in an IF clause, a qualifier with a '0' in a val setting would indicate that either the val setting must be a '0', or if not set at '0' then some other val setting for the same qualifier must be a '1' for the rule to be enabled. {Cj,n} indicates that choice place j is validated with a confidence factor of n. We present the full set of transition rules as used in our first example:

Transition Rule Set	RBPN notation
TR1: IF Q1: Last CUNY math assessment was not taken yet	IF Q1(1,-,-)
THEN C1: student needs CUNY exam,100	THEN {C1,100}
AND Q2: This student needs preplacement testing	AND Q2(-,-,-,1)

TR2: IF Q1: Last CUNY math assessment was Passed (25 or higher) THEN C6: Follow Curriculum Requirements,80 AND Q2: This student needs no initial placement	IF Q1(-,-,1) THEN {C6,80} AND Q2(-,1,-,-)
TR3: IF Q1: Last CUNY math assessment was Failed (below 25) AND Q3: Last math course was None taken yet THEN Q2: This student needs initial placement	IF Q1(-,1,-) AND Q3(-,-,-,-,-,1) THEN Q2(1,-,-,-)
TR4: IF Q1: Last CUNY math assessment was Failed (below 25) AND Q3: Last math course was NOT None taken yet THEN Q2: This student needs placement review	IF Q1(-,1,-) AND Q3(-,-,-,-,-,0) THEN Q2(-,-,1,-)
TR5: IF Q2: This student needs placement review AND Q3: Last math course was MAT M1 AND Q4: Last math grade was A OR B OR C THEN C3: MAT M2, 100	IF Q2(-,-,1,-) AND Q3(1,-,-,-,-,-) AND Q4(1,1,1,-, . . . ,-) THEN {C3,100}
TR6: IF Q2: This student needs placement review AND Q3: Last math course was MAT M1 AND Q4: Last math grade was NOT(A OR B OR C) THEN C2: MAT M1, 100	IF Q2(-,-,1,-) AND Q3(1,-,-,-,-,-) AND Q4(0,0,0,-, . . . ,-) THEN {C2,100}

TR7: IF Q2: This student needs placement review
AND Q3: Last math course was MAT R2 OR Independent Lab study
THEN C4: MAT R2, 100

IF Q2(-,-,1,-)
AND
Q3(-,-,1,1,-,-)
THEN {C4,100}

TR8: IF Q2: This student needs placement review
AND Q3: Last math course was MAT M2
THEN C3: MAT M2, 100

IF Q2(-,-,1,-)
AND
Q3(-,1,-,-,-,-)
THEN {C3,100}

TR9: IF Q2: This student needs initial placement OR placement review
AND Q3: Last Math course was MAT 03 or above
THEN C7: Inconsistency in answer; restart session, 99

IF Q2(1,-,1,-)
AND
Q3(-,-,-,-,1,-)
THEN {C7,99}

TR10: IF Q2: This student needs initial placement
AND R1: ICUNY ≥ 25
THEN C7: Inconsistency in answer; restart session, 100

IF Q2(1,-,-,-)
AND
[ICUNY $\geq$ 25]
THEN {C7,100}

TR11: IF Q2: This student needs initial placement
AND R2: ICUNY < 15
THEN C2: MAT M1, 100

IF Q2(1,-,-,-)
AND
[ICUNY < 15]
THEN {C2,100}

TR12: IF Q2: This student needs initial placement AND R3: {(ICUNY ≥ 15) AND (ICUNY < 20)} THEN C4: MAT R2, 100	IF Q2(1,-,-,-) AND {[ICUNY ≥15] AND ICUNY < 20]} THEN {C4,100}
TR13: IF Q2: This student needs initial placement AND R4: {(ICUNY ≥ 20) AND (ICUNY < 25)} THEN C5: MAT R3 self-study in math lab, 90	IF Q2(1,-,-,-) AND {[ICUNY ≥ 20] AND [ICUNY < 25]} THEN {C5,90}
TR14: IF Q2: This student needs no initial placement AND Q3: Last math course was MAT 03 or above AND Q4: Last math grade was A OR B OR C THEN C6: Follow Curriculum Requirements, 99	IF Q2(-,1,-,-) AND Q3(-,-,-,-,1,-) AND Q4(1,1,1,-, . . . ,-) THEN {C6,99}

These rules were developed to assist students at Kingsborough Community College in determining whether or not they needed mathematics remediation, and if necessary the appropriate course to take. The certainty factors are based upon responses from both the mathematics faculty and students who assisted in the system testing. Variation in the certainty values is based upon the following observations during the test phase. We found there were some students who did not have complete confidence in their responses to the system questions. This result was a reflection of our sample population which consists of many students who are

returning to college after an extended period away, or are part-time and take only one or two courses per semester. In either case, it may have been several semesters since the CUNY assessment test or the last Mathematics course was taken. As a result we adjusted the confidence factors accordingly, and also included an inconsistency choice selection.

2.5 Table of Rules

In this section we present the rules of the system organized in a tableau. This format is beneficial for assisting in the determination of redundancies and inconsistencies, as well as in achieving rule minimization.

Transition Rule Table for the Small Expert System

	Q1	Q2	Q3	Q4	[ICUNY]	Ci	Q2
TR1	(1,0,0)					C1 (100)	(0,0,0,1)
TR2	(0,0,1)					C6 (80)	(0,1,0,0)
TR3	(0,1,0)		(0,0,0,0,0,1)				(1,0,0,0)
TR4	(0,1,0)		~(0,0,0,0,0,1)				(0,0,1,0)
TR5		(0,0,1,0)	(1,0,0,0,0,0)	(1,1,1,0,0,0,0,0,0)		C3 (100)	
TR6		(0,0,1,0)	(1,0,0,0,0,0)	~(1,1,1,0,0,0,0,0,0)		C2 (100)	
TR7		(0,0,1,0)	(0,0,1,1,0,0)			C4 (100)	
TR8		(0,0,1,0)	(0,1,0,0,0,0)			C3 (100)	
TR9		(1,0,1,0)	(0,0,0,0,1,0)			C7 (99)	
TR10		(1,0,0,0)			$I \geq 25$	C7 (100)	
TR11		(1,0,0,0)			$I < 15$	C2 (100)	
TR12		(1,0,0,0)			$15 \leq I < 20$	C4 (100)	
TR13		(1,0,0,0)			$20 \leq I < 25$	C5 (90)	
TR14		(0,1,0,0)	(0,0,0,0,1,0)	(1,1,1,0,0,0,0,0,0)		C6 (99)	

Table 1

2.6 A Small System Trace

In the following we describe a trace through a small expert system for student placement in remedial mathematics. We will present the manner in which the expert system functions with the qualifiers and rules as we have written them, and we will show how the trace can be modeled by the use of a RBPN. To facilitate the flow of the trace we have included a Goal-Choice Path Tableau (see Table 2). Illustrated in it are the possible qualifiers and their val settings, relational qualifiers, and the appropriate rules, firing as they are enabled by responses to system queries or as the result of a new val setting by a rule firing.

First, we recall that for this system there are no initial start-up settings for qualifier vals or relationals. Qualifiers are evaluated as they occur within rule orderings. TR1, the first rule in the rule set, needs only qualifier Q1 and therefore the switch transition fires issuing a query for the qualifier. In response to the Q1 query, a val assignment of 1 or 3 will result in the selection of choice C1 or C6 respectively. Q1 with qualifier val set 1 effectively completes the session as the transition, TR1, assigns qualifier Q2 the val set 4. There is no rule space that will accept this resultant val set state, as an input, and the goal C1, set by TR1, is reached by the system. The response to qualifier Q1's query with val set 3 enables TR2. This rule has the effect of setting Q2 with

val set 2, as well as assigning choice C6 a certainty factor of 80. The significance here is that with Q2 at val set 2, the query transition for Q3 is enabled. This is due to TR14 which has become the sole rule that can be enabled. A Q3 query response with val set 5 will in turn generate a Q4 query. A val set response of 1, 2, or 3 to the Q4 query enables TR14. All other responses result in system closure. If TR14 does fire it has the effect of "modifying" C6, since it gives a weighting factor different from the current value. In this case we see the rule firing give this choice a greater weight. The previous firing had given C6 an 80 weight, while TR14 assigns a factor of 99. The certainty factor must be adjusted to reflect this newly applied goal weighting value.

Initial responses to qualifier Q1 with val set 2 result in a Q3 query transition with the subsequent enabling of TR3 or TR4. The first is enabled if val set 6 is indicated for Q3, and the second for any other response. As a result of a TR3 firing, Q2 gets a val setting of 1, whereas a TR4 firing marks Q2 with val set 3.

The Q2 val set 1 necessitates a query for the CUNY assessment score. TR10, TR11, TR12, or TR13 will be enabled depending upon the respondent's score entered for the ICUNY variable. With a passing score (ICUNY of 25 or higher), TR10 is enabled selecting choice C7. This goal indicates the entered assessment score to be inconsistent with prior responses. A score below 15 enables TR11 which selects choice C2. Scores ranging from 15 to 19 enable TR12

which in turn selects choice C4. Scores from 20 through 24 result in TR13 being enabled. This rule will cause choice C5 to be taken.

Qualifier Q2 having val set 3 results in qualifier Q3 being examined for its specific val setting. A val set of 3 or 4 enables TR7 with C4 the resultant choice. A val set of 2 enables TR8 with choice C3. A val set of 1 results in a query for qualifier Q4; a resultant val set for Q4 of 1, 2, or 3 enables TR5 with goal C3 taking effect. Any other val setting for Q4 enables TR6 with choice C2 as the goal result.

Lastly, rule TR9 examines qualifier Q3 for val set 5. This is done for both Q2 val sets 1 and 3. Under these conditions TR9 is enabled and selects goal C7 as the resultant choice. As stated earlier C7 refers to an inconsistency within the qualifiers val sets.

Goal-Choice Path Tableau

$\{Q_a \ldots \wedge Q_b \Rightarrow TR_x \Rightarrow C_j \ldots \wedge Q_c \ldots \Rightarrow \ldots \Rightarrow TR_y \Rightarrow C_j \ldots \wedge Q_d \ldots\} \Rightarrow TR_z \Rightarrow \sum C_n \wedge Q_e$

(1) Q1(1) ⇒ TR1 ⇒ [C1 ∧ Q2(4)]

(2) {Q1(2) ∧ Q3(1)} ⇒ [TR4 ⇒ Q2(3)] ∧ Q4(1,2,3) ⇒ TR5 ⇒ C3

(3) {Q1(2) ∧ Q3(1)} ⇒ [TR4 ⇒ Q2(3)] ∧ Q4(4+) ⇒ TR6 ⇒ C2

(4) {Q1(2) ∧ Q3(2)} ⇒ [TR4 ⇒ Q2(3)] ⊢ (Q2,Q3) ⇒ TR8 ⇒ C3

(5) {Q1(2) ∧ Q3(3)} ⇒ [TR4 ⇒ Q2(3)] ⊢ (Q2,Q3) ⇒ TR7 ⇒ C4

(6) {Q1(2) ∧ Q3(4)} ⇒ [TR4 ⇒ Q2(3)] ⊢ (Q2,Q3) ⇒ TR7 ⇒ C4

(7) {Q1(2) ∧ Q3(5)} ⇒ [TR4 ⇒ Q2(3)] ⊢ (Q2,Q3) ⇒ TR9 ⇒ C7

(8) {Q1(2) ∧ Q3(6)} ⇒ [TR3 ⇒ Q2(1)] ∧ *R1* : ICUNY ≥ 25 ⇒ TR10 ⇒ C7

(9) {Q1(2) ∧ Q3(6)} ⇒ [TR3 ⇒ Q2(1)] ∧ *R2* : ICUNY < 15 ⇒ TR11 ⇒ C2

(10) {Q1(2) ∧ Q3(6)} ⇒ [TR3 ⇒ Q2(1)] ∧ *R3* : 15 ≤ ICUNY < 20 ⇒ TR12 ⇒ C4

(11) {Q1(2) ∧ Q3(6)} ⇒ [TR3 ⇒ Q2(1)] ∧ *R4* : 20 ≤ ICUNY < 25 ⇒ TR13 ⇒ C5

(12) Q1(3) ⇒ TR2 ⇒ [}{ ∧ Q2(2)] ∧ Q3(5) ∧ Q4(1,2,3) ⇒ TR14 ⇒
C6,99 + {C6,80}[a] ≡ C6,89

(13) Q1(3) ⇒ TR2 ⇒ [}{ ∧ Q2(2)] ∧ Q3(1,2,3,4,6)[b] ⇒ {C6,80}

(14) Q1(3) ⇒ TR2 ⇒ [}{ ∧ Q2(2)] ∧ Q3(5) ∧ Q4(4+)[b] ⇒ {C6,80}

}{ Represents a choice which reappears with another value from a different rule later. Shown as }Cn{ when more than one choice is possible.

{Cn,wt} refers to the choice (}{) selected by a previous rule enabling.

≡ indicates an averaged or modified goal weighting.

⊢ (QX, . . ., QY) ⇒ Z These qualifiers val settings will enable rule Z.

[a] Multi-valued goal is numerically averaged.

[b] The qualifier val set is requested, but based on the given response set(s) it may not be implemented by this rule.

Table 2

2.7 Rule-Based Petri Net Graphs for the Small System

In this section we present the Rule-Based Petri Net Graphs for the small expert system used for mathematics remediation placement that we have described. In these graphs we distinguish between Qualifier, Relational, and Choice Places as indicated below:

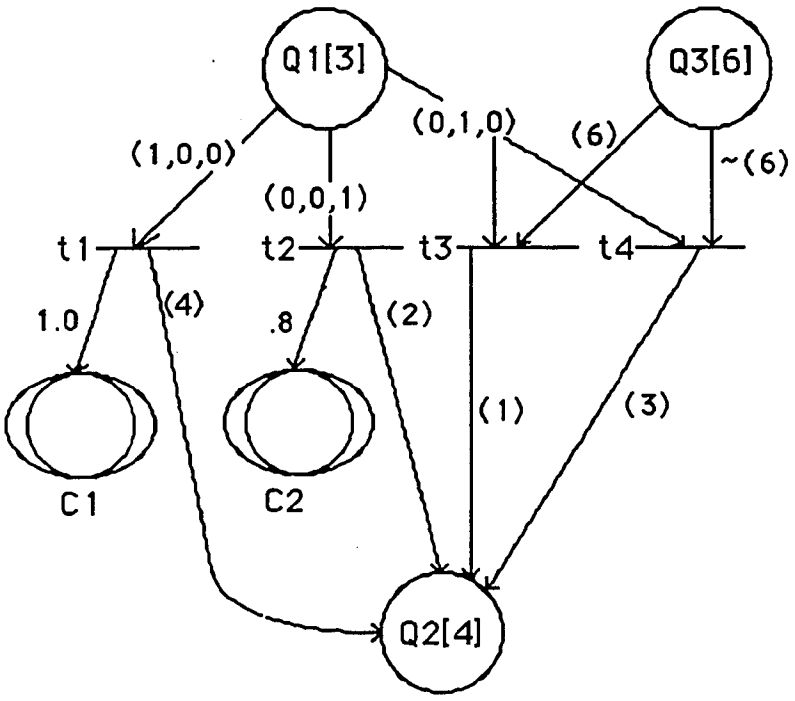
Relational Qualifier Place | Qualifier Place | Choice Place

We present two sets of graphs as well as the Reachability Graph (figure 2.5) for the small system. The query transitions associated with all of the qualifier places are not indicated in the first set of RBPN graphs (figure 2.1). This was done to improve the clarity of these graphs, as well as demonstrate their pictoral relationship with the Basic Petri Net. In both sets of graphs we use the following conventions:

* Choice place confidence factors are coincident with their transition arcs;
* Relational places contain the mathematical expressions to be evaluated;
* Qualifier places are represented in RBPN notation, Qn[m],

with their input and output arcs indicating the corresponding qualifier attribute requirements and settings, respectively.

In the second set of graphs (figures 2.2, 2.3, and 2.4) query transitions are indicated with the use of a switch. Bold lines for their transition arcs denote the path of a simple token during a query. For clarity, those transitions used for either query or relational expression evaluation are enclosed within rectangular transition boxes, in the style of Reisig [33].



Petri Net Graphs for the Small System

Figure 2.1 (Part 1 of 2)

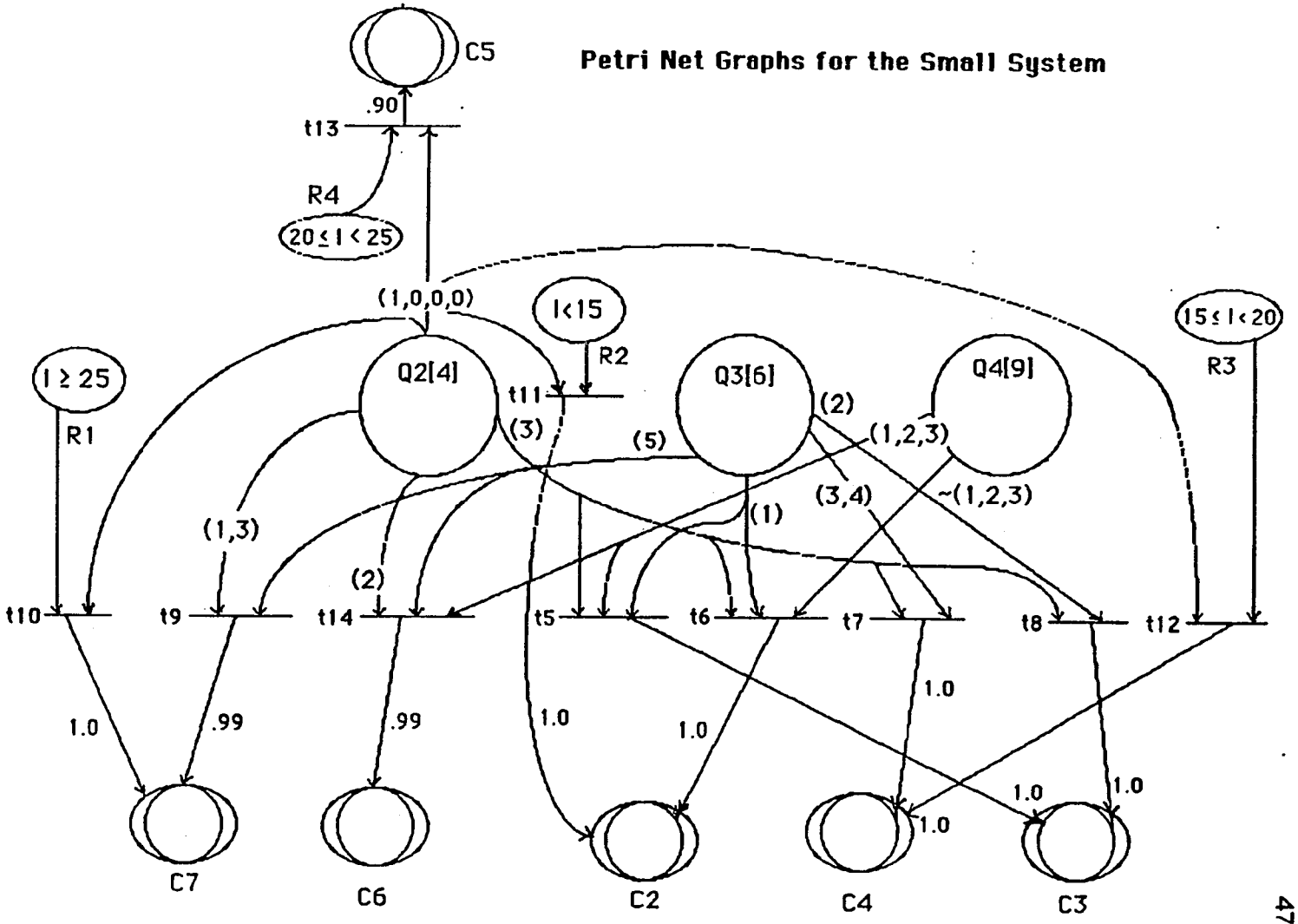
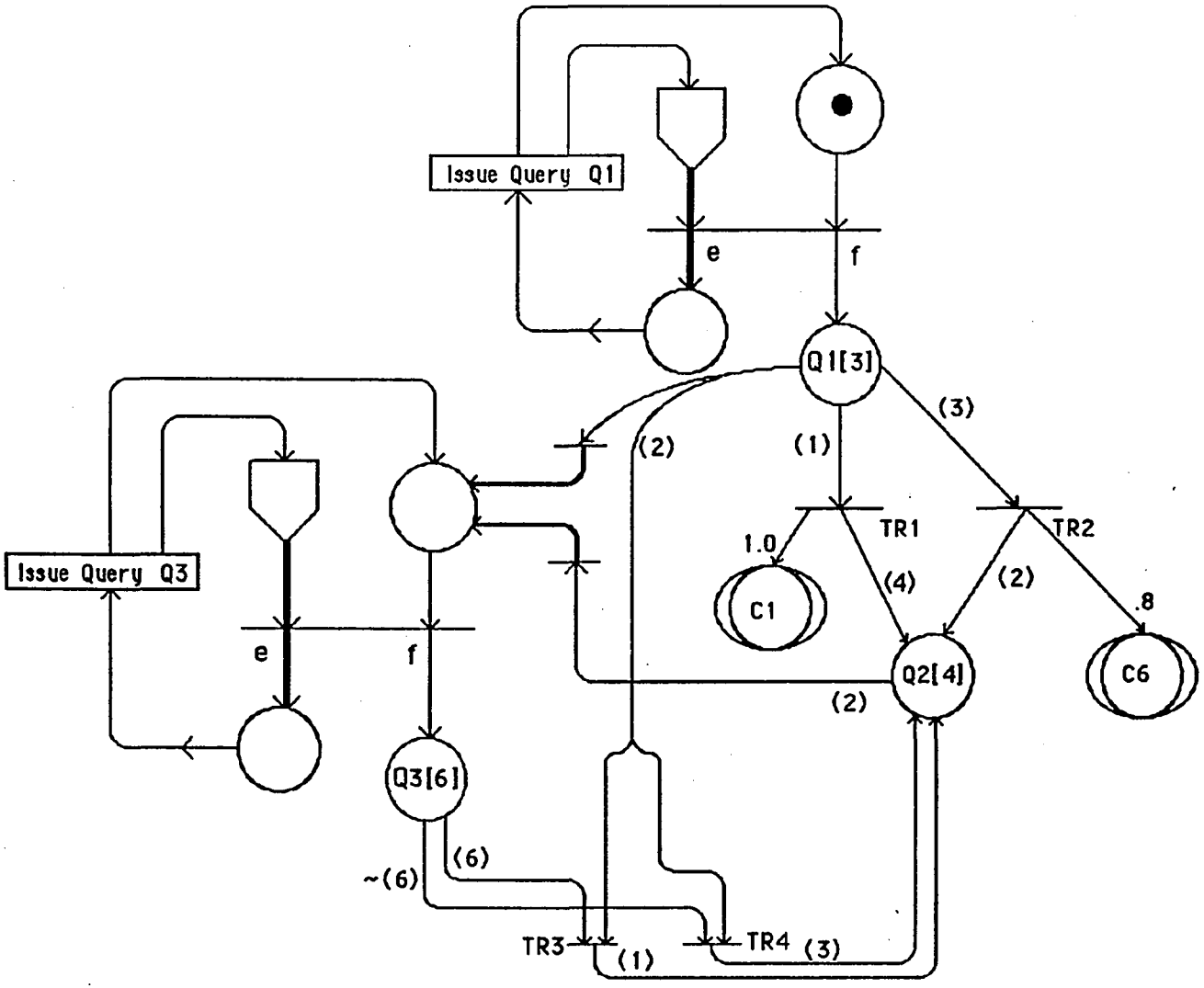


Figure 2.1 (Part 2 of 2)



Transition Rules 1-4 with Switches

Figure 2.2

Transition Rules 7-13 with Relational Val Switch

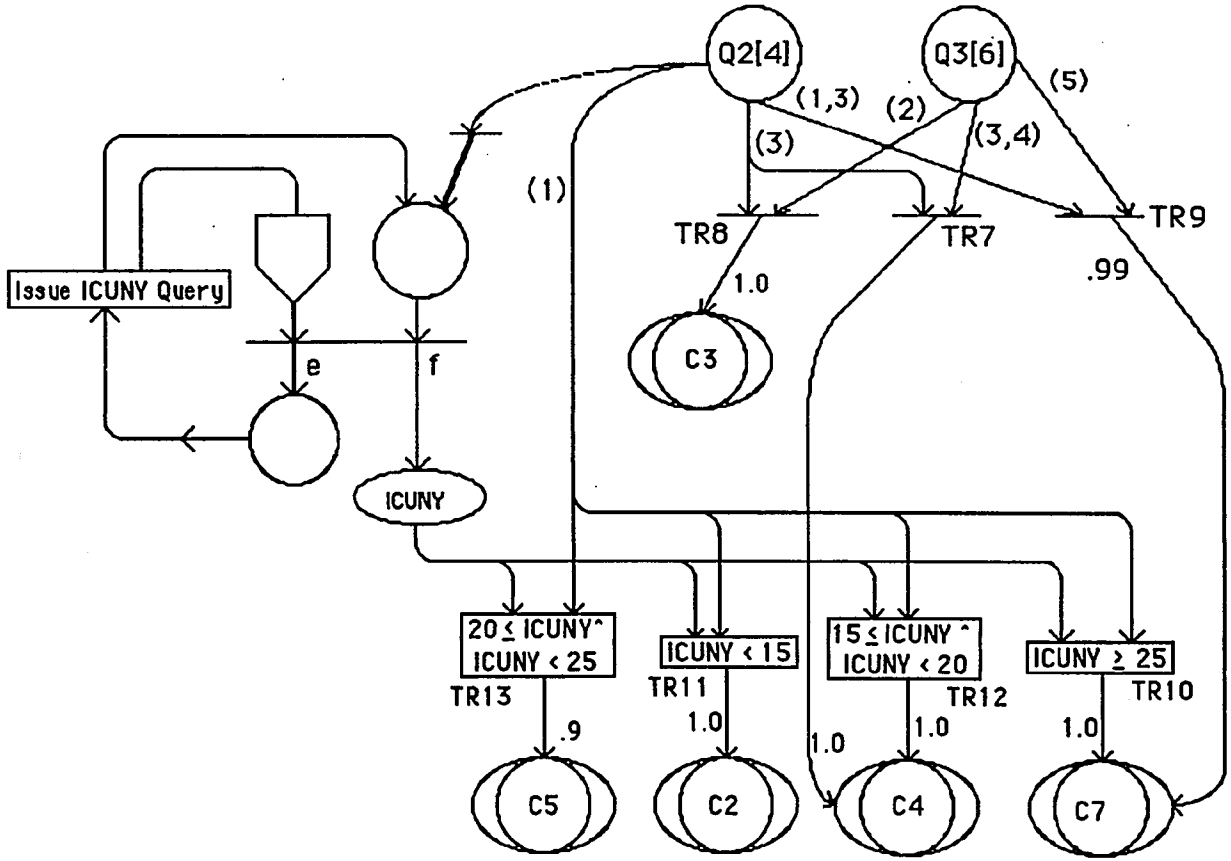


Figure 2.3

Transition Rules 5, 6, and 14 with Q4 Switch

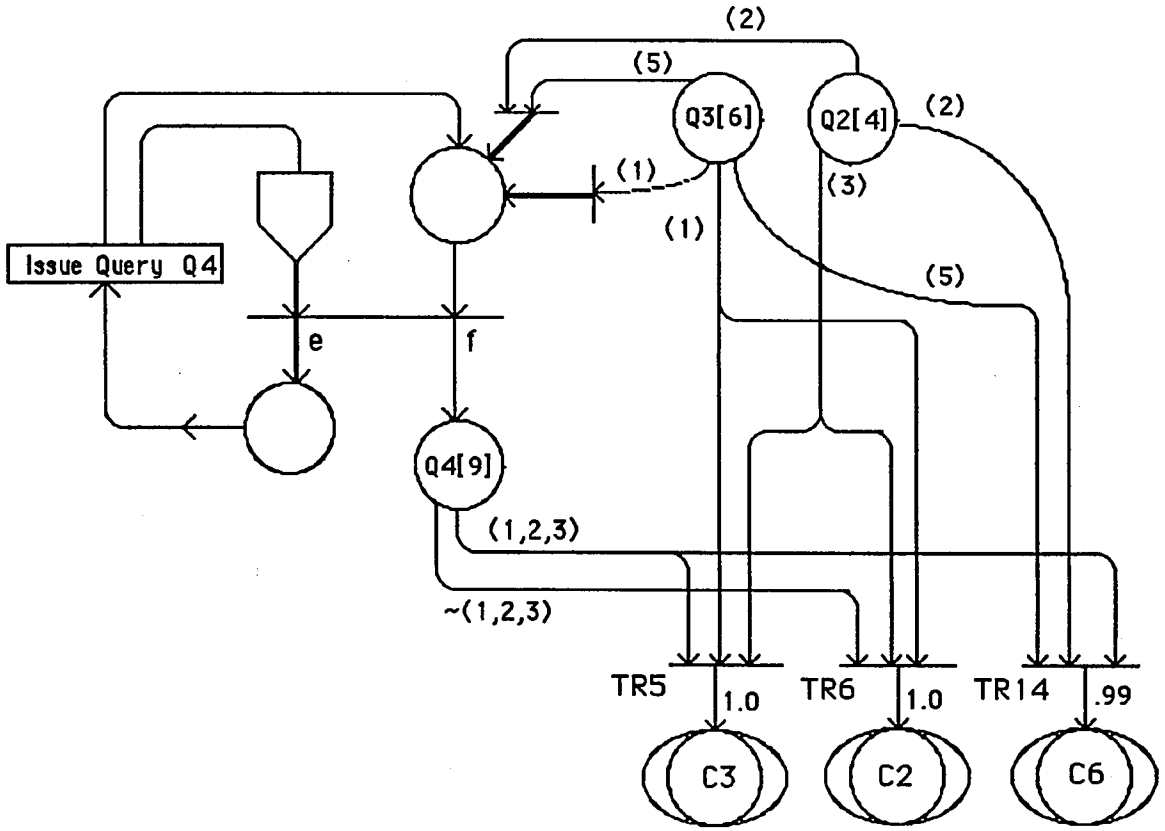


Figure 2.4

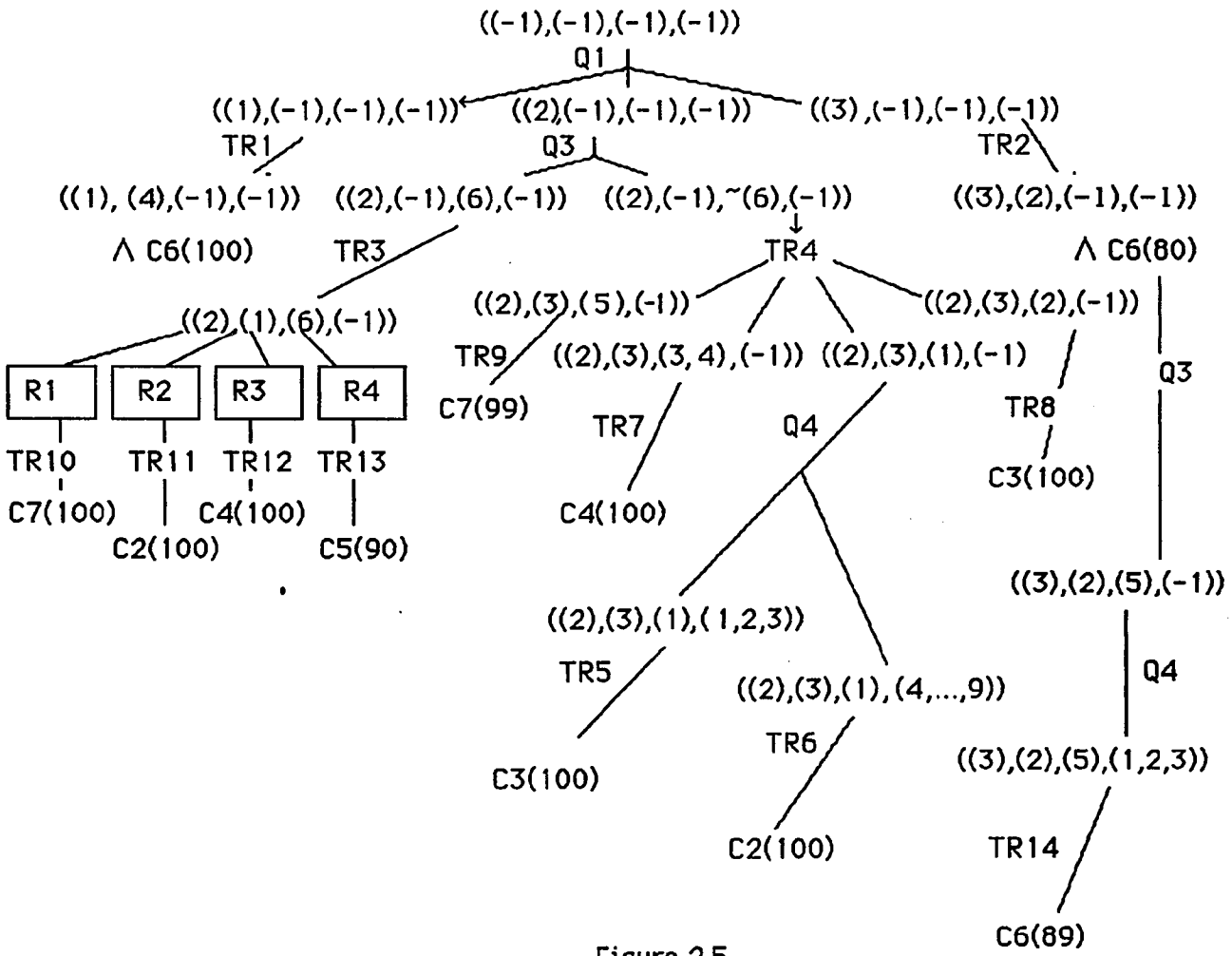

Reachability Graph for the Small Expert System
((-1),(-1),(-1),(-1))
Q1
((1),(-1),(-1),(-1))
((2),(-1),(-1),(-1))
((3),(-1),(-1),(-1))
TR1
Q3
TR2
((1), (4),(-1),(-1))
((2),(-1),(6),(-1))
((2),(-1),~(6),(-1))
((3),(2),(-1),(-1))
∧ C6(100)
TR3
TR4
∧ C6(80)
((2),(1),(6),(-1))
((2),(3),(5),(-1))
((2),(3),(2),(-1))
TR9
((2),(3),(3,4),(-1))
((2),(3),(1),(-1)
Q3
R1
R2
R3
R4
C7(99)
TR7
Q4
TR8
TR10
TR11
TR12
TR13
C3(100)
C7(100)
C4(100)
C4(100)
((3),(2),(5),(-1))
((2),(3),(1),(1,2,3))
Q4
TR5
((2),(3),(1),(4,...,9))
C2(100)
C5(90)
TR6
((3),(2),(5),(1,2,3))
C3(100)
TR14
C2(100)
C6(89)

Figure 2.5

Chapter 3 The Formal Model: Rule-Based Petri Nets

There have been many applications in which various types of Petri Nets have been used for modeling [1, 2, 4, 6, 37]. In this chapter we present a Petri Net formalism to be used in the modeling of rule-based systems. Our model's specificity to the area of expert systems makes it a unique tool for evaluating the performance and reliability of rule-based systems. We start with generic description definitions, which could be implemented by any typical language for expert system development. We then use these formalisms in the rule descriptions. The definitions of the Rule-Based Petri Net (RBPN) model will be developed from the rule descriptions.

3.1 The Rule Terms - Descriptions

We provide the following definitions and illustrations of the terms to be used in rule clauses of the form :

IF conditional THEN activity.

Terms enclosed within < >'s are required while those within { }'s are optional.

1) [< Qualifier > {NOT} < val_1 > { connector < val_2 >}]

Qualifier : a statement ending in a verb such as,

' The lighting is ' or, 'The weather outside is '.

val(s) : any phrase(s) that may complete a qualifier statement such as,

' The lighting is ' (1) sunlight
(2) indoor bright
(3) indoor dark
(4) night

When multiple values are used in the qualifier of a conditional clause they are connected by an OR. As an example we have,

IF ' The lighting is : sunlight OR indoor bright'.

In the conditional clause, NOT may appear before the val set, for example, IF ' The lighting is : NOT [sunlight OR night] '. Correspondingly, when the qualifier is used in an activity clause the val's are connected by an AND; as an example we have the following consequent,

THEN ' The weather outside is: foggy AND raining '.

2) [< id-name > := < rule-express >]

id-name : an identifier used to hold the result of one of the following: an evaluated expression or constant, the value of a user entered data item in response to a query, or a string data

value.

'_:=_' : the assignment operation. Assign the evaluated result of the expression on the right of the operator to the identifier on the left of the operator.

rule-express : any algebraic expression using integers, reals, or identifier names connected by any of the arithmetic operators (+, -,*, /,**), or any string of literal or character data. Specialized functions (such as the trigonometric functions, absolute value, EXP, LOG, or SQRT) may be implemented within a rule-express.

3) [< rule-express$_1$ > relop < rule-express$_2$ >]

relop : any one of the relational operators < , > , = , >= , <= , or <>. The entire clause is evaluated as being true or false.

4) [< choice, CF >]

choice : a possible solution or goal that is generated by a rule firing.

CF : certainty factor ; a probability or measure assigned to the choice when this particular rule is fired. The CF of a particular choice may be the average of its several CF's or a value computed by some other predetermined algorithm. The value of the CF gives the user a confidence level or gradient of

reliability with the resultant choice or goal.

Rule Terms: Usage within the Transition Rules

In the following we present a Petri Net formalism to be used in the modeling of rule-based systems. Each *Conditional* or *Activity* phrase will be given its corresponding Petri Net relation. Unless otherwise indicated, a Conditional will refer to input places, and an Activity will reference output places. The rule itself corresponds to the transition, and a rule that can be fired will be synonymous with an enabled transition.

IF *Conditional* THEN *Activity*

Where a *Conditional* clause is of the form :

$$[< \text{Qualifier}_1 > \{\text{NOT}\} < \text{val}_{11} > \{\text{OR} < \text{val}_{12} > \text{OR} \ldots \text{OR} < \text{val}_{1n} >\}] \text{ AND } \ldots$$

$$\text{AND } [< \text{Qualifier}_k > \{\text{NOT}\} < \text{val}_{k1} > \{\text{OR} < \text{val}_{k2} > \text{OR} \ldots \text{OR} < \text{val}_{km} >\}]$$

$$\{\text{AND } [< \text{rule-express}_{11} > \text{ relop } < \text{rule-express}_{12} >] \text{ AND } \ldots$$

$$\text{AND } [< \text{rule-express}_{j1} > \text{ relop } < \text{rule-express}_{j2} >]\}$$

An *Activity* clause is written in the form :

$$[< \text{id-name}_1 > := < \text{rule-express}_1 >] \text{ AND } \ldots$$

$$\text{AND } [< \text{id-name}_p > := < \text{rule-express}_p >]$$

$$\text{AND } [< \text{choice-1, CF}_1 >] \text{ AND} \ldots \text{ AND } [< \text{choice-r, CF}_r >]$$

$$\text{AND } [< \text{Qualifier}_1 > < \text{val}_{11} > \{\text{AND} \ldots \text{AND} < \text{val}_{1u} >\}] \text{ AND} \ldots$$

$$\text{AND } [< \text{Qualifier}_c > \{< \text{val}_{c1} > \text{AND} \ldots \text{AND} < \text{val}_{cs} >\}]$$

Conditional clauses : these preconditions must be satisfied prior to transition firing.

$$[< \text{Qualifier} > \{\text{NOT}\} < \text{val}_1 > \{\text{OR } < \text{val}_2 >\}]$$

For the conditional 'Qualifier/val' clause the input place represents the qualifier, and the token(s) it contains represents a val set. This val set might be a singleton or, as in the case of an as yet unanswered or undetermined query, an empty set (zero place). The transition will be enabled if any member of the set of val's for the place has been validated. This validation may be the result of a direct query or the consequence of an earlier inference.

A negated val, NOT v_i, is true under the following conditions:

1) v_i is false or

2) v_i has no attribute value, and there is at least one val other

than v_i which is valid. That is, v_c is true for some c, $c \neq i$. A negated val set, for example, NOT(v_k OR . . . OR v_n), is valid if and only if v_i is false or has no value for all corresponding $i = k, \ldots, n$, and for some j, v_j is true. For all other cases the val set would evaluate as false.

[< rule-express$_1$ > relop < rule-express$_2$ >]

In this case the relational condition is written into a place, and the indicated condition must be satisfied as a precondition for the associated transition to fire.

Activity clauses : these postconditions occur when all preconditions have been validated, and the transition is enabled and fireable.

[< id-name > := < rule-express >]

The id-name is assigned a value or string based upon the evaluated expression. This is considered an atomic action so that any data items which are used cannot be changed by the firing of any other transitions. There is no place associated with this clause, but it may be viewed as a transition event. Reisig [33] gives a further discussion of these transition events in his airline seat assignment and reservation system.

[< choice, CF >]

A choice place contains two token types. They are standard tokens and data tokens. Standard tokens have the usual meaning, while data tokens may contain attributes. In this instance, the attribute is the computed Certainty Factor (CF). Upon firing, the place receives a standard token indicating that the choice place is active, and the data token is marked with its CF attribute value or weight.

[< Qualifier > $< val_1 >$ {AND $< val_2 >$ AND . . . AND $< val_r >$}]

When used in the activity clause, the 'Qualifier/val' form has the output place representing the qualifier, and the token(s) it receives represents a non-empty val(s) set. Only after firing may the qualifier be examined to investigate whether additional preconditions are satisfied for other transitions.

Preconditions are the standard and data token set requirements and/or place conditions necessary for the determination of a live transition. Postconditions are the results affected by the firing on any place(s) and/or identifiers.

3.2 Rule-Based Petri Nets - Definitions

In this thesis a variation of Petri Nets has been developed to perform modeling and analysis of Rule-Based Systems. These new Rule-Based Petri Nets (RBPN's) attempt to overcome certain restraints associated with the basic Petri Net model. These limitations include the lack of token types and interactive time delays. We deal with both of these restrictions. The former in the specification of the net, and the latter in the design of qualifier attributes for the RBPN. In addition, we have added the use of a restriction on a set of specific precondition places, with the requirement on them that unless modified as a postcondition place they are not changed by the fired transition. In our design of RBPN's, we draw upon certain modifications made by Extended Petri Nets [1], Colored Petri Nets [15], and Extended Place/Transition Nets [26].

Definition 3.1.1 A *Standard Token* is a token which does not have any associated attributes, and will not contain any data. These tokens have typically been represented by dots in Petri Net Graphs.

Definition 3.1.2 A *Data Token* is a token which has associated attributes. The values of these attributes may

be modified by transition firings. A data token **D** with *n* attributes is indicated as **D**[*n*], with the i*th* attribute **D**(i). Data tokens associated with Qualifier places can be viewed as a vector of arity m, where m is the number of unique attributes that the qualifier may possess. Data tokens associated with Choice places contain the Certainty Factor attributes of the goal choices.

The presence of a standard token within a place connotes the usual meaning. The existence of a data token in a place implies a meaning which is place type dependent (e.g., val setting or quantity value).

Input Place Types

1) Condition Qualifier place: unless otherwise expressed the data tokens for these places are given the initial marking vector of $(-1_1, \ldots, -1_n)$. If a data token has this initial value when examined, an external request is generated for the data attributes. The vector marking will be modified according to the response, and a standard token entered in the place. When the i*th* attribute is validated, the i*th* position is set to +1, and any existing negatively marked positions are set to zero. All non-negative positions remain

unchanged. For example, from the vector (0,1,1,0, . . . ,0), abbreviated (2,3), we would infer that the second and third val attributes are simultaneously valid. Also implied is that NOT first attribute is true. Note that the condition NOT (first or second attributes) is invalid. The presence of a standard token indicates that the qualifier has been changed from the initial marking for its data token. Without a standard token in this place, any transition requiring use of this qualifier will not fire. Pictorially this is represented by Figure 3.1, where we have the following conditions:

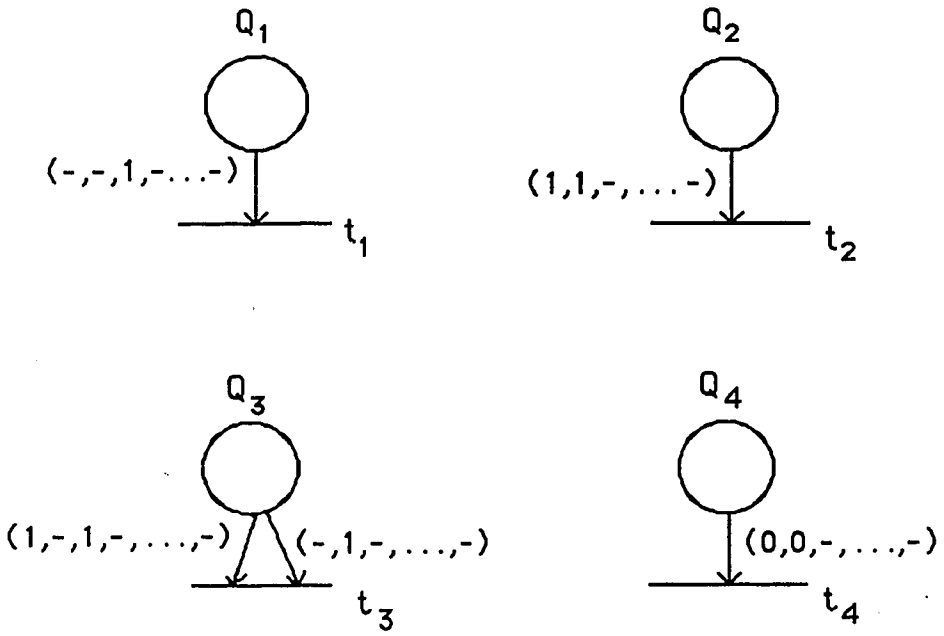
t_1 is enabled when Q_1 has its third attribute valid.

t_2 is enabled when Q_2 has its first or second attributes valid.

t_3 is enabled when Q_3 has its first or third attributes, and second attribute valid.

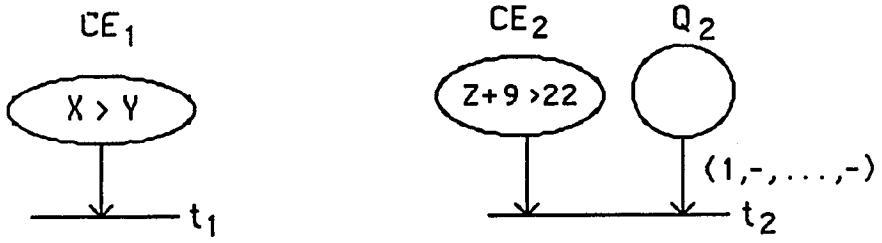
t_4 is enabled when Q_4 does NOT have its first or second attributes valid.

2) Conditional expression places: these places are set to true or false depending upon the current values used in evaluating the expression(s). A true setting permits the arc from place CE_1 to transition T, written (CE_1,T), to be active. Any additional input places for T, whether CE_i or qualifier type (Q_j), must also be active or validated for T to be enabled. If a variable used in a conditional expression is unknown, a request is generated and firing ceases



Qualifier Place Preconditions

Figure 3.1



Conditional Place Preconditions

Figure 3.2

until a response is received. As an example, in Figure 3.2 we have the following conditions :

t_1 is enabled when CE_1 (X > Y) evaluates as true.

t_2 is enabled when CE_2 (Z + 9 > 22) evaluates as true and Q_2 has its first attribute valid.

Output Place Types

1) Assignment place: upon firing, in this instance, the variable is given the evaluated expression value. This may be either an algebraic or string data value.

2) Choice place: upon firing the CF of the choice may be assigned a certainty value, as part of its postcondition, based upon the rule that was fired. If the CF has a value it may be reevaluated according to a predetermined algorithm. A CF once set to 0 or 100 remains constant throughout the entire run of the active system event. These special certainty factor values are assigned with the understanding that they are unchangeable, and they are given only under the most stringent conditions. A choice place also receives a standard token as part of its postcondition.

3) Activity Qualifier place: these places receive a standard token and the indicated qualifier val set vector marking. As an output place we restrict marking changes to one of the following:

a) either single or multiple valid attribute conditions, or

b) single or multiple negated attribute conditions.

Unlike an input qualifier place, when used as an output place for transition t_x, the qualifier place p_q may only have a single entry arc (t_x, p_q). That is, any attributes that are assigned (valid or invalid) are given to all designated vector positions, and the Qualifier name may only appear once in the Activity Clause. This is shown in Figure 3.3, which consists of the following three conditions:

if t_1 were to fire p_1 would get its first attribute set on as valid;

if t_2 were to fire p_2 would get its first and third attributes set on as valid;

if t_3 were to fire p_3 would get its first attribute negated (set as invalid);

Transition types:

RBPN's are distinguished from other Petri Net extensions by the manner in which val sets and quantifiers, of the input and output places, are used by the transitions. The combining of two types of threshold criteria, using both standard and data tokens, for establishing transition preconditions makes this a unique modeling tool for Expert System development. A val set place containing a standard token indicates the existence of a marked attribute

condition. That is, it confirms the presence of some attribute(s) for the corresponding data token. For a qualifier place the absence of a standard token implies that either a query will have to be issued, or the qualifier will be given attributes as a receiving (output) place of a transition firing. What we have is a net with a subset of places, incapable of receiving tokens (not contained in the set of Output places), that will be given a marking based upon external query responses. The determination of what ordering the queries will take, and the order in which transitions are fired in the system will be examined later.

Definition 3.1.3 Each transition, $t_i \in T$, in a RBPN is defined as the 5-tuple,

$t_i = [pr(t_i), q(t_i), r(t_i), s(t_i), z(t_i)]$, where,

$pr(t_i)$ are the preconditions that have to be satisfied before the transition t_i can be enable enabled;

$q(t_i)$ is the transition procedure specifying the effects on the token attributes of activating transition t_i ;

$r(t_i)$ is the output resolution procedure indicating the course of action to be followed in routing the tokens or val sets to alternative

places in the set of output places of t_i ;

s (t_i) is the transition schema; it specifies whether an action, *ts*, or query, *tr*, will result from the transition firing. This corresponds to a switch place representation;

z (t_i) refers to the execution time associated with each transition;

These components of transitions, including their use and implications, will be discussed in detail below.

1) Precondition - the specification of preconditions allows us to model cases where a place p_i participates in only a subset of the transitions (each of whose set of input places may include more than one entry from p_i). That is, specific arcs may have non-conflicting simultaneous conditions imposed on the data token (val set) of p_i.

As an example place p_j may represent the qualifier :

'The color of the flower is'

with corresponding positional val set attributes {red, white, violet, blue}. An arc (p_j, t_i) describes a precondition on the val set data token in qualifier place p_j for transition t_i. As illustrated in Figure 3.4a we have:

Arc (p_j,t_{i1}) marked (1,-,-,1) representing,

'The color of the flower is' red OR blue

Arc (p_j,t_{i2}) marked (-,1,-,-) representing,

'The color of the flower is' white

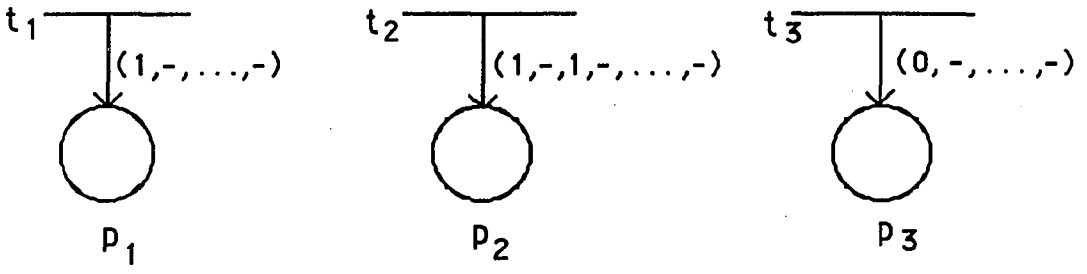
Each of the following markings would then be a permissible precondition in place p_j for transition t_i : (1,1,x,x) or (x,1,x,1) where x represents a don't care val. We then say that the data token vector satisfies the precondition marking.

The following examples illustrate Satisfiable Precondition states for an arbitrary input place p_i.

Case I: If p_i may only contain standard tokens, then $\mathbf{M}(p_i) \geq \mathbf{W}(p_i, t_j)$, where **W** is the token count required in place p_i to fire t_j, and $\mathbf{M}(p_i)$ is the standard token marking (number of tokens) in place p_i prior to firing transition t_j.

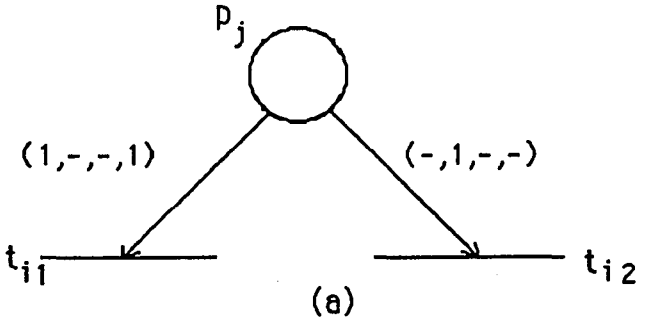
Case II: If p_i has a qualified data token with attributes (a val set), then the assigned attributes as present in the data token vector must give a permissible precondition for any data token arc marking.

Given a data token vector $(v_1, \ldots, v_n)$ with a precondition arc marking $(a_1, \ldots, a_n)$, for some $a_i = 1$, there must be at least one

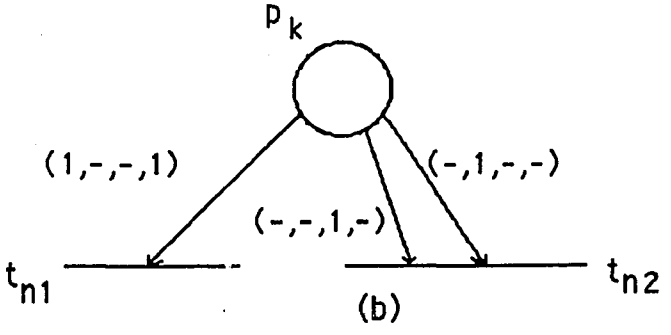


Qualifier Place Postconditions

Figure 3.3



$t_i = t_{i1} \wedge t_{i2}$



$t_n = t_{n1} \vee t_{n2}$

Logic Structures

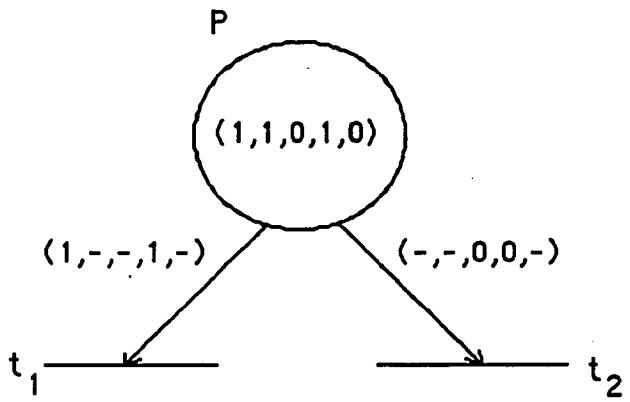
Figure 3.4

$v_i = 1$ for a satisfiable precondition. With a negated (NOT) arc marking, for all $a_k = 0$ then $v_k \neq 1$ and there exists a k, such that $v_k = 0$ to give a satisfiable precondition. As an example, in Figure 3.5, place p with val set (1,1,0,1,0) satisfies the arc (p, t_1) with marking (1,-,-,1,-), but does not satisfy the negated (p, t_2) arc with labeling (-,-,0,0,-).

Case III: If a qualifier data token in p_i has not been given any attributes then it will not satisfy any precondition arc marking. A standard token in this place should trigger a query transition firing.

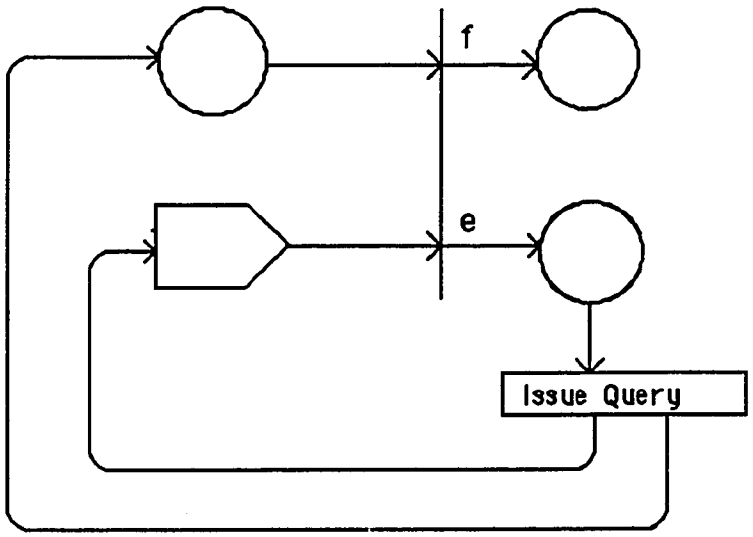
2) Transition Procedure - the transition procedure indicates how data token attributes are to be modified by each transition along with the evaluation of computed attributes as in the case of CF's. Any atomic actions which are inherent to the transition are also part of the transition procedure. These include identifier assignments, external message sending, as well as the execution of independent subroutine packages.

3) Output Resolution Procedure - the output resolution procedure specifies the movement of tokens to the places in $\mathbf{O}(t_i)$. This allows for the modeling of cases where tokens move to only a subset of places in $\mathbf{O}(t_i)$. There are several ways in which the



Precondition Arc Markings

Figure 3.5



Query Transition Structure

Figure 3.6

output procedure may be described.

a) As a set of atomic actions which may be indicated by an identifier and its associated expression.

b) As a procedure whose evaluation will indicate where the tokens should be placed.

c) As a function which uses a vector marking associated with an output place to determine its updated contents.

Let us look at the following examples :

I) Given $\mathbf{O}(t_i) = \{p_j, p_k\}$ and X := Y is the X identifier assignment for t_i. A standard token is to be moved to place p_j if Condition_1 is true and to place p_k if it is not true.

We can write this as :

$r(t_i) = [\, X := Y \;\wedge\; (\text{Condition\_1} \Longrightarrow \mathbf{M'}(p_j) \leftarrow \mathbf{M}(p_j) + 1)$

$\wedge\; (\neg\, \text{Condition\_1} \Longrightarrow \mathbf{M'}(p_k) \leftarrow \mathbf{M}(p_k) + 1)\,]$

where, **M'**(p) is the new marking (number of standard tokens) in place p upon completion of the transition firing.

II) Let $\mathbf{O}(t_a) = \{ p_b, p_c \}$ where p_b is a qualifier place with val set {r,s,t} and p_c is a choice place. We want t_a to validate the second of three possible attributes for p_b, and also to assign

a Certainty Factor value of y to the Choice place p_c. We can specify this as :

$$r(t_a) = [\, \mathbf{M_d}'(p_b) \leftarrow (\, fd(e,1,e) \Longrightarrow \mathbf{M_d}(p_b)\,) \wedge \mathbf{M_e}'(p_c) \leftarrow (\, fc(y) \Longrightarrow \mathbf{M_e}(p_c)\,)]$$

where, $\mathbf{M_d}(p)$ is the data token marking in qualifier place p;

fd is a data token change function for transition t on qualifier place p;

fd(i) is the value of the i*th* function component;

v_i is the corresponding value for the i*th* vector position.

The value v_i is identical to the data token value **D**(i) as previously stated (Def. 3.1.2).

if fd(i) = 1 then assign $v'_i = 1$ and simultaneously where,

fd(k) = e if $v_k < 0$ then mark $v'_k = 0$, else $v'_k = v_k$.

if fd(i) = 0 then assign $v'_i = 0$ and unless otherwise indicated all other $v'_k = v_k$.

$\mathbf{M_e}(p)$ is the data token marking in choice place p;

fc is the data token change function for transition t on choice place p.

In general we have :

if $\mathbf{M_e}(p)$ has a CF of 0 or 100, it will remain unchanged by fc;

if $\mathbf{M_e}(p)$ has not been given a CF, it is assigned the fc argument value;

if $\mathbf{M_e}(p)$ has a prior CF, a predetermined algorithm is used to evaluate the new CF. For example, a Bayesian product or averaging technique might be applied. It is also possible to use the transition function techniques as developed by McAloon [18] in determining the CF. For example if $\mathbf{M_e}(p) = 90$ and transition t assigns a CF of 80 to p, then

$$\mathbf{M_e}'(p) \leftarrow (\mathbf{M_e}(p) + CF) / 2 .$$

4) Transition Schema - in modeling a rule-based system there is a need to set apart those transitions which represent an internal action from those which require some type of external (or user interface) communication. We distinguish between these two transition types as follows:

tr - those transitions which may require a query response from the environment to communicate a val for a qualifier place

ts - those transitions which may be enabled solely by an internal activity and require no external communication.

We will implement Query Generations by using the concept of a special place called a 'switch' to be associated with any input qualifier place that requires interaction with a user. We have

initialized the qualifier val set as stated earlier, and concurrent with that is an initialization, if needed, of its switch place with a standard token. When a token is present in the switch, it functions as a request indicator for transition *tr.* Without the switch token present, the *ts* transition is enabled only if it meets all the precondition tests. Figure 3.6 illustrates this technique.

Every substantial Expert System will attempt to resolve an unanswered query in the course of its evaluation procedure. Because of this, most systems allow qualifiers to include 'don't know' or 'none of these' as possible attribute values. This will enable the expert system to continue with a limited or restricted knowledge base rather than halt. We have felt it necessary to require that our model also mimic this typical expert system behavior. That is, the model will also expect a response of some valid attribute type to its query transitions before resuming. A possible avoidance of this "wait for response" situation may be through the use of stochastic methods for queries with a corresponding time attribute value in the data tokens that would be used for communication purposes.

At this point in our discussion we have described the essentials necessary for understanding the concept of a Rule-Based Petri Net. In the following we give its formal definition.

Definition 3.1.4 A Rule-Based Petri Net is the 5-tuple,

RB = (P, T, A, **I**, **O**), such that,

(1) P = PQ ∪ PC ∪ PR where

PQ = $\{pq_1, \ldots, pq_q\}$, the set of qualifier places,

PC = $\{pc_1, \ldots, pc_c\}$, the set of choice places, and

PR = $\{pr_1, \ldots, pr_r\}$, the set of places holding relational expressions. Here q, c, r ≥ 0 and PQ, PR, and PC are mutually disjoint sets.

(2) T = TS ∪ TR is the union of two disjoint finite sets containing the transition types. They are,

TS = $\{ ts_1, \ldots, ts_s \}$, the set of transitions that are internally enabled and,

TR = $\{ tr_1, \ldots, tr_v \}$, the set of transitions that may require an external interaction.

We have TS ∩ TR = ∅ and s, v ≥ 0.

(3) A is a subset of B, where

B = { PR ∪ PQ } x T ∪ T x { PQ ∪ PC }

(4) For each $t_i, t_j \in T$, **I**(t_i) is the set of input

places, such that,

$$\mathbf{I}(t_i) = \{ pq_r \mid (pq_r, t_i) \in A \} \cup \{ pr_k \mid (pr_k, t_i) \in A \}$$

(5) For each $t_i , t_j \in T$, $\mathbf{O}(t_j)$ is the set of output places such that,

$$\mathbf{O}(t_j) = \{ pq_m \mid (t_j, pq_m) \in A \} \cup \{ pc_n \mid (t_j, pc_n) \in A \}$$

(6) We allow a place to have multiple inputs into a transition, but limit entries to an output place at a maximum of one of each token type (standard and data token) per place.

Note that (6) implies a qualifier place may have more than one input arc to a transition, indicating for example that the two unique val set entries, {a} and {b}, must both be true. This can only be accomplished with two entry arcs. As an output place, however, a qualifier may only receive a single entry arc. This arc may give any val set entries a new status (valid or negated). Similarly a choice place should receive only one entry with a single CF value. This value may be obtained from a function whose argument is dependent upon the current CF (if present) in the choice place.

3.3 Markings on Rule-Based Petri Nets

An RBPN marking $\mathfrak{M}$ is an assignment of standard and data tokens to the places of the net. These tokens describe the state of the system at any particular point in time. The number and position of tokens may change during transition firings (execution).

Definition 3.2.1 A Standard Token Marking μ of a RBPN, RB = (P, T, A, **I**, **O**), is the vector (μ_c, μ_q, μ_r), where μ_c, μ_q, and μ_r are functions defined as follows:

μ_c : PC --> $\mathcal{N}$

μ_q : PQ --> $\mathcal{N}$

μ_r : PR --> {0,1} 1= true, and 0=false

These markings can also be described as follows:

$\mu_c = (\mu_{c1} , \ldots , \mu_{cj})$ where j = | PC |, the cardinality of the set PC, and each $\mu_{ci} \in \mathcal{N}$, i=1, . . . ,j.

The vector μ_c gives for each choice place pc_i in the RBPN the number of standard tokens in that place.

$\mu_q = (\mu_{q1} , \ldots , \mu_{qm})$ where m = | PQ |, the cardinality of the set

PQ, and each $\mu_{qk} \in \mathcal{N}$, $k = 1, \ldots, m$.

The vector μ_q gives for each qualifier place pq_k in the RBPN the number of standard tokens in that place.

$\mu_r = (\mu_{r1}, \ldots, \mu_{rn})$ where $n = |$ PR $|$, the cardinality of the set PR, and each $\mu_{rl} \in \{0,1\}$ where $l=1, \ldots, n$.

The vector μ_r gives for each relational place pr_l in the RBPN the truth or falsity of the relational expression as evaluated for that place.

Definition 3.2.2 A Data Token Marking ∂ of a RBPN, RB= (P,T,A, **I**, **O**), is given by a pair of functions, ∂_c and ∂_q, defined on PC and PQ respectively, such that, $\partial_c(pc) \in \mathcal{N}_{100}$ for all $pc \in$ PC, and where $\partial_q(pq)$ is a vector $(\varepsilon_1, \ldots, \varepsilon_n)$ with $\varepsilon_i = 0,1$, or -1 and n is the number of attributes.

For all $i = 1, \ldots, n$, $\varepsilon_i = 1$ indicates the qualifier's *ith* val setting is valid (on), or $\varepsilon_i = -1$ indicates an unassigned (original) state for the val set, or $\varepsilon_i = 0$ indicates the qualifier's val setting attribute is false (off). We require that if we have a pq where

$\epsilon_i = 1$ for some i, then for all $j \neq i$, either $\epsilon_j = 1$ or 0.

As an example, let us look at some of the types markings available for RBPN's. A marking $\partial_q(pq_i) = (2,4)_5$ is equivalent to the vector (0,1,0,1,0) indicating that the second and fourth attributes of qualifier place $pq_i \in PQ$ are valid.

. Similarly, $\partial_q(pq_i) = \sim(3)_5$ indicates (-1,-1,0,-1,-1) while the initial marking for a qualifier place, pq_i, is written $\partial_q(pq_i) \equiv (-1,-1,-1,-1,-1)$ or alternatively $(-1)_5$.

Definition 3.2.3 A Marked Rule-Based Petri Net $\mathbf{M} = (RB, \mu, \partial)$ is a RBPN, RB = (P,T,A, **I**, **O**), with a standard token marking μ, and a data token marking ∂. This may also be written as the marked RBPN,

$$MRB = (P, T, A, \mathbf{I}, \mathbf{O}, \mu, \partial).$$

RBPN's execute by firing transitions. A transition fires by removing and/or creating standard tokens along with examining and/or setting attributes in data tokens.

Definition 3.2.4 A transition $t_i \in T$ in a marked RBPN, RB = (P,T,A, **I**, **O**) with the marking vector $\mu = (\mu_c, \mu_q, \mu_r)$, and ∂ the pair of functions

∂_c and ∂_q is enabled when all the $\mathbf{I}(t_j)$ satisfy $pr(t_j)$.

Definition 3.2.5 A transition $t_j \in T$ in a marked RBPN, $RB = (P,T,A, \mathbf{I}, \mathbf{O})$ with the marking vector $\mu = (\mu_c, \mu_q, \mu_r)$, and ∂ the pair of functions ∂_c and ∂_q may fire whenever it is enabled. Firing a transition t_j results in a new marking $\mu' = (\mu_c', \mu_q', \mu_r')$, and ∂' the pair $(\partial_c', \partial_q')$, defined by $r(t_j)$ on the elements of $\mathbf{O}(t_j)$.

From these definitions we may refer to the new markings as output of the resolution function. We therefore have,

$$\mu' = r(t_j)(\mu) \quad \text{and} \quad \partial' = r(t_j)(\partial).$$

Chapter 4 Anatomy of the Design of the Advisement Expert System

In this chapter, we explore the development of an expert advisement system for Data Processing students at Kingsborough Community College. We will give the reader an insight into the workings of the currently available manual (human expert) system, along with the newly developing automated expert system. This chapter is divided into two sections. In the first part we will describe the characteristics of the manual system. This will include the current types of problems encountered, guidelines, charts, and any other information and resources used by a human expert as part of the manual system. We will give the data flow tables, describe the rule extraction, and give the data base definitions as they are generated by the expert. It is from these elements of the human expert system that we develop the heuristics which form the basis of the computerized expert system. This will be followed by a standard Petri Net model for the manual system. In the second section we describe the process of automating the work of the human expert; that is, we give the generation of the expert system. Computerization will be discussed as it relates to how the rules are:

a) used by the expert system

b) kept in the system (design of the data base)

c) integrated within the data flow of the system

We have chosen to construct an expert system for student advisement directed at students majoring in Data Processing. This expert system will assist a candidate in selecting her/his appropriate DP course(s) if any, as well as the college core course requirements. The need for such a system is evident from the wide disparity in abilities among the many entering freshman and transfer students, as well as currently enrolled students. Many colleges spend vast amounts of time and money performing the functions of preregistration advisement, only to be followed by additional counseling at the actual time of registration. This system is designed to be of service not only at each of these counseling periods, but also throughout the entire year.

One goal of this system is to clarify the curriculum options that are available to students. Most two-year colleges today have a Data Processing curriculum with a minimum two tracks (i.e., Microcomputer Technology and Computer Programming), in addition to a Computer Science major. Consequently, many students come to preregistration or to registration unaware of the DP program options available. Some students have a predisposition towards a major in Computer Science when an investigation of the Data Processing programs might be more appropriate. Because of student misconceptions of these specialties, it is necessary for both the

Data Processing and Computer Science advisers to be aware of each other's programs and requirements so as to best serve the needs of the students requesting advice. Obviously the same awareness of these distinctions is required of the general counselors. Unfortunately, even with the counseling services available, many students are directed into inappropriate classes or programs. This is partly due to the continual change in program requirements and the creation of new curricula.

4.1 The Manual (Human Expert) System

The current economic and educational environment in which we find ourselves has had an impact upon what was once the rigid and rather sacrosanct process of course registration. Today many of our colleges offer walk-in registration for those students who, on the spur of the moment, have decided to enroll in one or more college courses. Another group of students have elected to transfer to this institution. The increase in such registrants has resulted in a demand for appropriate advisement in all disciplines to be available immediately at the time of registration. Regrettably, this occurs at the most difficult time to give extended individual attention to the student. The walk-in periods usually occur at the tail end of the registration sessions when course selections are

severely limited and departmental advisers may not be readily available. The advantages that would be offered through the use of an automated advisement system are quite apparent under these circumstances. We will now look at the preregistration process that occurs during the semester prior to formal registration.

4.1.1 Preregistration Procedures

We are able to divide new students into two groups: those who have been preregistered by the dean's office or an appropriate department, and those who have not been preregistered.

Preregistration is the standard beginning for student advisement. In our model for DP majors preregistration usually occurs in the previous semester for those students who will be enrolled as upper freshmen or higher. Lower freshmen may have been preregistered while still in high school, which may be from 2 to 5 months prior to the actual registration period. Those students currently enrolled in a DP course will be preregistered for the next semester by their Data Processing instructor. This is in the form of a class counseling session, where the different program requirements are specified along with the college core courses. The students reregistration cards are completed and approved by their instructor. Those students not currently enrolled in a DP course are

assigned a faculty counselor from within the DP Department. An appointment is arranged and the student meets with the adviser for academic counseling and preregistration course approval. Instructors may meet with as many as 20 advisees on an individual basis during this preregistration period. In both group and individual sessions the faculty adviser must be familiar with the requirements not only of the programs offered within his/her own department, but also the college-wide course requirements, and their prerequisites. For this to be done correctly, the adviser is furnished with a course survey booklet, college catalog, and degree requirements chart [Appendix C]. Along with these items, advisers are given placement packets for Mathematics and English. These interpret the codes used to assign the correct remedial course placement, if required. The codes are determined on the basis of the student's rating on the CUNY assessment exam in conjunction with any pertinent courses taken.

4.1.2 Formal Registration Procedures

In many cases the entire preregistration and advisement procedure is repeated again during the actual registration. Although the student should have the preprogrammed course registration information from his or her preregistration session, this

information may not be useful because classes have been closed, canceled, or not offered. Under these conditions student program planning must begin again at registration, a chaotic time for faculty and students. Oftentimes, these sessions mirror those of preregistration, with all the remediation, core, and departmental requirements checked over again. The advisement process is shown pictorially in Figure 4.1.

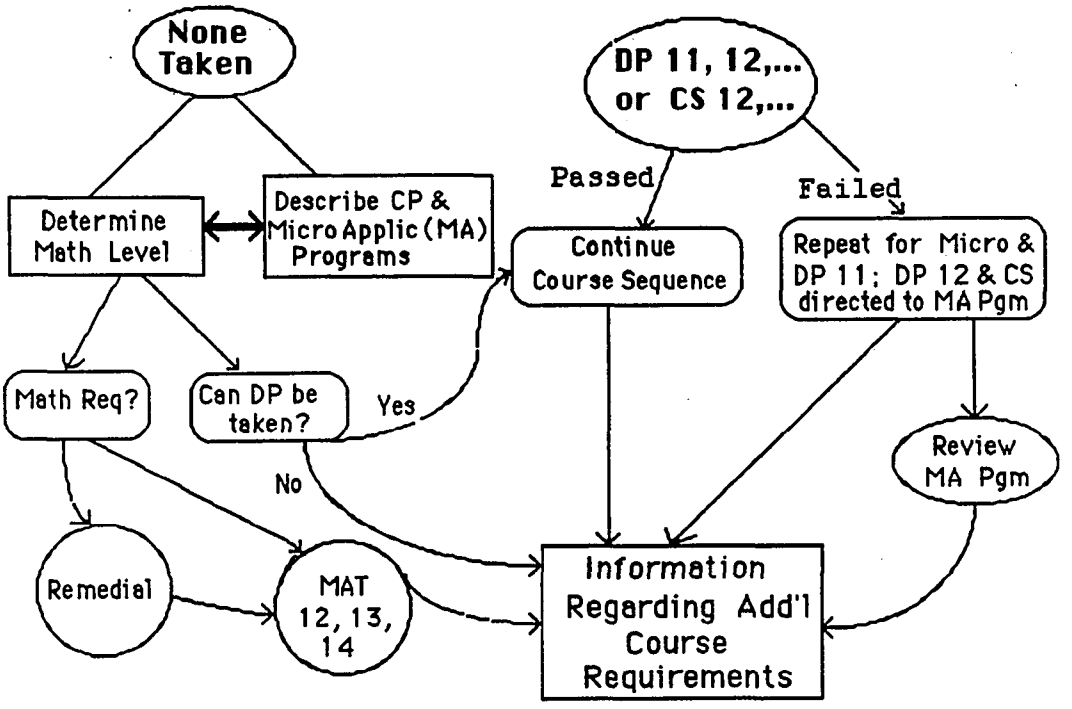
The following is a representative collection of questions that could be posed by an adviser to a student at registration.

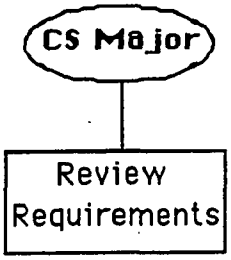
1) What is your major area? If undecided what are your interests?
2) What courses have you completed? Were any at another college? If so, have you been given credit or a waiver by the Registrar?
3) Do you have a copy of a transcript?
4) Are your CUNY assessment scores available? Evaluate Math and English levels.
5) Have you met any core group requirements? If applicable, what are the core groups you are interested in?
6) Have you met any of the college-wide requirements?
7) Have you considered taking an elective course? If so, which course?
8) A series of queries on days, hours, number of credits to be

Data Flow Graph of the Manual System

1) Student is directed to the appropriate adviser (based upon major field)

2) For D.P. majors, student is asked:
   What courses have been taken?





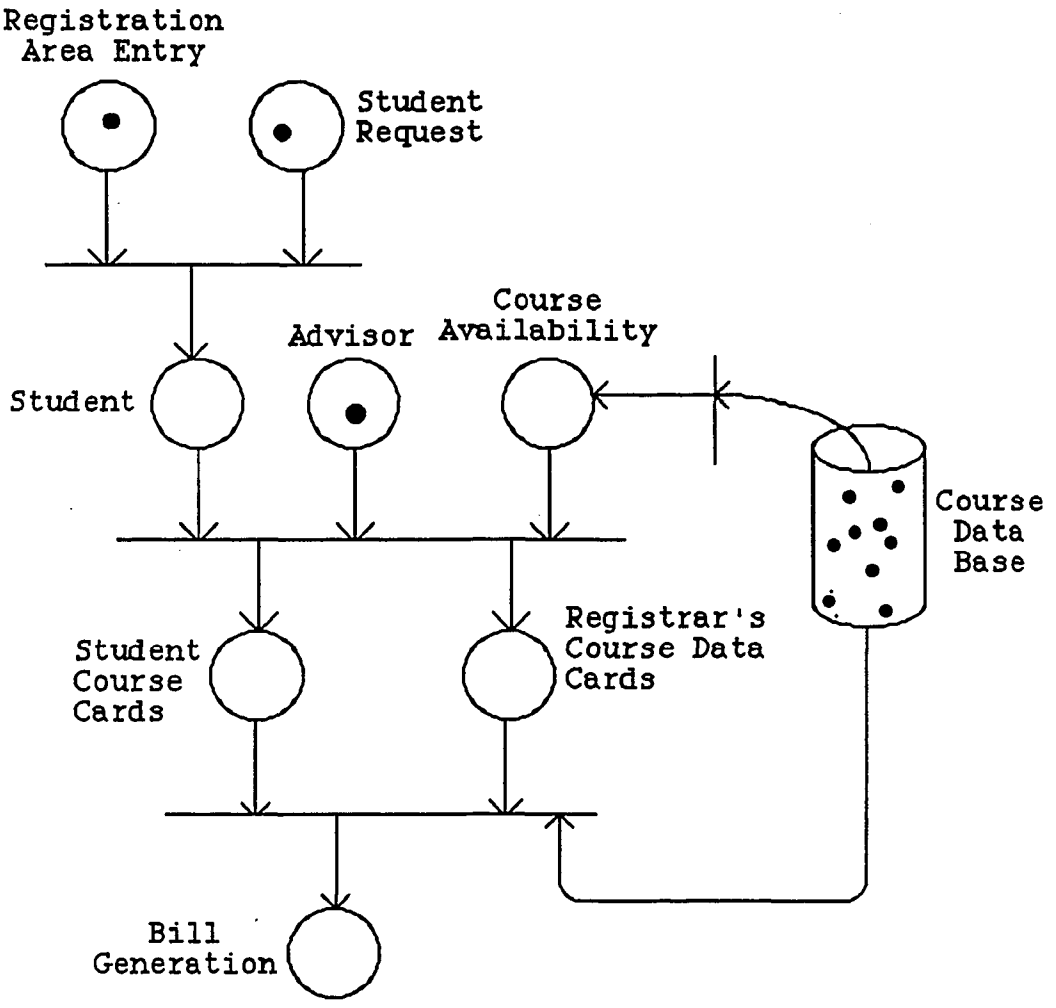
All prospective CS majors have a review of requisites to certify that CS is appropriate. If not student is directed to DP adviser.

Figure 4.1

taken, full-time or part-time, day or evening, and hours of employment.

Based upon the answers to these questions, the faculty adviser is able to assist the student in preparing an appropriate course program. Not indicated in this presentation are the problems encountered when an entering student has not taken either of the CUNY assessment exams in Mathematics or English. The limitations of advisement inherent under those circumstances will not be addressed here.

The manual registration process is modeled as a standard Petri Net Graph in Figure 4.2. In this graph we see a student entering the registration area upon being granted permission to enroll for courses. At this time, the student is assigned to a faculty adviser from either the student's major area or the academic advisement office for registration assistance. A program is planned based upon the curriculum requirements, the student's record of completed courses, and the current offering of available classes. The adviser helps the student find courses which complement his or her curriculum requirements, and thereafter the student receives course registration entry cards; simultaneously, the registrar's data base of course class size is updated. A bill is generated for the student based upon the total number of credits being taken, and the registration process is completed. Each of these events



Petri Net Model of the Registration Process

Figure 4.2

involves the movement of tokens within the given basic Petri Net, with the final event completion indicated by the presence of a token in the Bill Generation place.

4.1.3 Data Base for the Manual System

The collection and quantification of data, along with assigning meaning and determining the relevancy of the data, is a major component of any information processing system. Although we have the traditional form of student records from which to evaluate and advise enrolled students, there are currently several non-traditional components which may be included in determining the student's status.

The traditional types of data which may be found in a newly admitted student's record are high school and college transcripts, standardized test scores, and assessment test scores. Such factors as life experience credit, work related experiences, and other non-standard educational formats may yield credits to be included in the student record. In general, the student has these factors evaluated for equivalent course credits by the college Registrar, or another officiating agency such as N. Y. State's CLEP testing system.

Several situations may occur which prompt special consideration for advisement of the non-traditional student.

Transcripts from another school must be checked along with a catalog from that institution in determining the possibility of any course equivalencies or advanced standing. Final determination of equivalencies falls within the domain of the Registrar or Department Chairperson. At the time of advisement or registration, these equivalencies may not have been formalized, and the adviser must determine appropriate courses and placement for this particular student's situation. The adviser must be able to determine a level of competency in the area of Data Processing perhaps based upon these non-standard criteria. This is especially important when operating under the time constraints of registration. Typically these situations vary from a new or late registering student wishing to enroll for one or two courses, to the Registrar not immediately being able to make a course evaluation of studies completed at another institution.

The student data base also has components collected from traditional sources. Most records come from the current transcript at the college. Here it is essential that both the adviser and student be familiar with any specialized grading rules. In particular, the grade received in any remedial course(s) must be verified in evaluating the need for a follow-up course, or for any courses required in the curriculum.

High school records are another source for the data base. These are helpful in determining preparedness for college level work.

Many departments offer courses that are specifically designed for the student who did not have any related high school courses. For example, both the Biology and Chemistry Departments offer differing entry level courses. These alternatives are designed for those students who did or did not have either of those science courses in high school. Many college level Mathematics courses require ninth, tenth, or eleventh year math as a prerequisite. As a consequence the students need to be aware of past high school courses completed, or they should have a copy of their record available for review.

CUNY assessment scores are an integral part of the student data file. They are essential in determining if remediation is necessary. In fact, placement in Mathematics and English is directly related to these CUNY scores. The CUNY assessment scores are also required in deciding whether the student possesses the reading, writing, and critical thinking skills needed to undertake courses in other disciplines. The scores also function as a guide in evaluating whether a student meets minimal requirements to pursue higher-level courses in related disciplines. For example, a mathematics score can be useful in determining readiness for Computer Science or Data Processing.

Lastly, a copy of the catalogue and a curricula requirements outline are invaluable resources in assisting the advisers, both within and outside their specific areas of expertise.

4.1.4 Rule Extraction

The function of the human specialist is to coalesce the data base information received from the student together with the student's desired goals and capabilities, and to assist the student in producing a meaningful and challenging course selection. In order to do this, the expert analyzes the student's academic status, the prerequisites and corequisites of courses, college-wide requirements, and departmental regulations vis-a-vis the student's course requests. Additionally, the adviser may have to provide information regarding alternative or corollary programs of study, in addition to those currently under consideration. These are some of the capabilities required of the expert in order to assist the student in creating an appropriate program of classes. A major component in the advisement process is that of Rule Extraction. Determining whether a student may or may not take a course and validating that result with a plausible set of guidelines encompasses the Rule Extraction process.

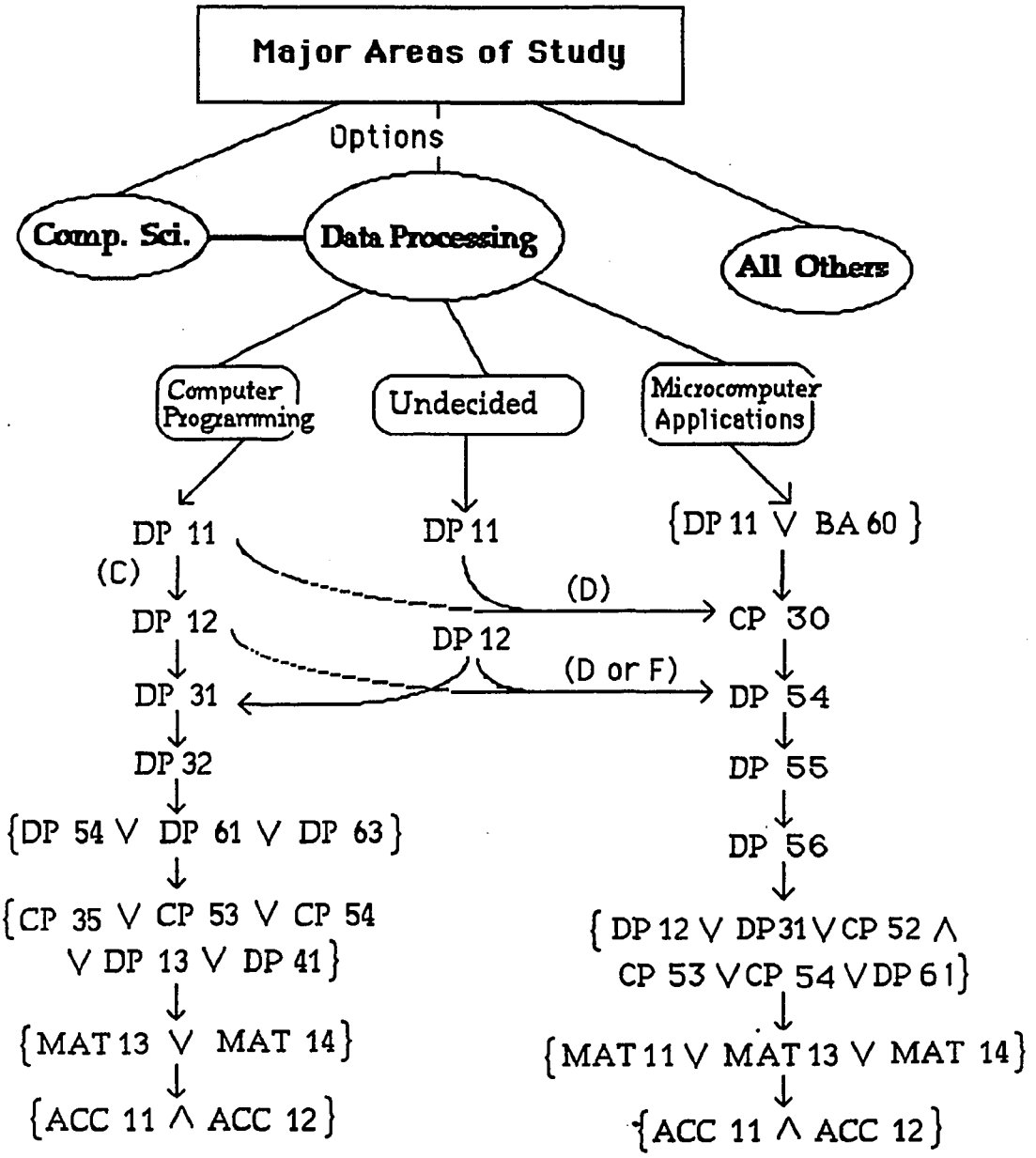
4.2 The Computerized Expert System

In the preceding section we have described the empirical research necessary for the human to develop an expert system for

student development. In this section, we describe our efforts to develop a computerized model of an expert system for Data Processing. We follow a pattern similar to the typical questioning exhibited in an individual student registration session.

The initial step in the advisement process is the determination of whether the student has a major area of study. The answer to this query will generate a specific set of questions dependent upon both courses completed within the major area and core requirements as determined by the major department. Students who have not selected an area of concentration are most often regarded as liberal arts majors for purposes of class registration and core requirements. This is all part of the knowledge acquisition function in the development of an Expert System. In Chapter Two we presented an example of a small system. This expert system serves as an adviser in evaluating the need for remedial mathematics placement, if any. This remediation adviser system is one component of the total Data Processing expert system. As the system developed, we were continuously evaluating the kinds of questions to be asked. Qualifiers changed along with the possible val sets they might assume. As a result, rules changed in scope and size. For example, was it really necessary to distinguish whether MAT 13 or 14 (see Figure 4.3) had been completed? Each satisfied the requirement, and any other math course would be an elective.

Individual small systems were set-up as prototypes and tested.



Horizontal lines indicate possible change routes from one curriculum to another. The corresponding letters in parentheses are the grades which can be used for changing tracks (but are not permitted for continuation in the curent program). Unless otherwise indicated, a passing grade is sufficient to continue down a track.

Partial Ordering of DP Courses

Figure 4.3

Some were able to be appended to the basis system while others created scenarios of irrelevant lines of questioning. In the development of the basis system, we went through a sequence of stages as part of Knowledge Acquisition. These have been formalized in the literature. They include, but are not limited to, the following: Identification, Conceptualization, Formalization, Implementation, and Testing. We will discuss each of these as they are related to the development of the expert system.

4.2.1 Identification Stage

In this step, the domain expert and knowledge engineer establish problem identification. These include defining what the problem is, itemizing its characteristics, and investigating sub-problems that may be encountered. Additionally one should determine the class of problem to be solved. Other questions include: Can the problem be handled through task partitions? What are the data? What is a solution to the problem, and what might be impediments to the solution? Resource and Goal identification must also be ascertained.

4.2.2 Conceptualization Stage

In this step, key data from the identification stage are made explicit. Concepts and relations are diagrammed to give a conceptual base for the prototype system. The types of data are examined as to what is given and what is to be inferred. Partial hypotheses are identified along with their corresponding subtasks. Relations between domain objects are extended to a diagrammatic hierarchy with causal relations, set inclusion, required constraints, and information flow graphs.

4.2.3 Formalization Process

In this stage the hypothesis space along with the underlying model of the process is formalized. Characteristics of the data are examined in greater detail. Sparse vs. plentiful amounts of data, uncertainty of the data, and logical interpretations of the data are investigated. Is meaning associated with the order of the data, so that the sequence in which questions are posed is significant? How are the data acquired? What are the classes of questions to be asked, and how reliable is the data? Is it hard or soft? What are the consistency and completeness of the data?

4.2.4 Implementation Stage

This step includes mapping the formalized knowledge into the representational framework associated with the tool chosen for the problem. As the requirements and specifications of the system and the shell implementation become well defined, the information flow develops into an executable program. This includes the use of domain knowledge and changes to it, refinements of the data structures (i.e., qualifier's val sets, relational expressions, choice sets) and local consistencies (e.g., managing both non-DP and DP majors within the larger Liberal Arts curriculum). Although accuracy may be achieved in a local environment, this does not guarantee global consistency when unified in the large system. All these factors contribute to the development of the prototype system.

4.2.5 Testing Stage

In order to evaluate the prototype and the representational forms used to implement it, we generated test cases through an arbitrary selection of students for preregistration advisement. This trial period revealed difficulties with some questions, and many had to be rephrased or reordered.

A major difficulty with the system was that of an incorrect response; an invalid data entry could only be remedied by restarting the system. This was easily achieved, however, and restarting the system was not deemed to be a significant problem. In fact, it allowed students unsure of their past academic history to generate several scenarios and make comparisons of the resulting recommendations. In most cases it was newly entering students who were initially advised through the system (most second semester students having been advised by a DP instructor), and as a result large variations of input data responses were achieved. In addition, data from those students in DP classes were collected to ascertain the validity of advisement vs. actual courses of study undertaken by students. Errors in the set of inference rules were able to be eliminated via the EXSYS capability of retracing questions through a 'Why' command when asked for a qualifier val set. Control strategies were able to be handled through the use of Rule analysis regarding the ordering of the rules. It was found that the system performed at maximal efficiency when related rules (those sharing two or more of the same qualifiers) were separated by at least one unrelated production. The test data were drawn from a large enough sample to produce an effective means of validating the accuracy of system responses when examined by the human expert. This close monitoring of the prototype enabled us to make effective changes for the development of the system. After

each change we revalidated and retested the entire system. This cycle would repeat itself within each stage of prototype revision and refinement.

Chapter 5 Production System Extensions to Rule-Based Petri Nets

As we have seen from the outset, most Petri Net models have undergone some form of extension or change as a way of providing additional capabilities and strengths, or as a means of creating a more generalized modeling tool. We foresee no less a need for expanding the RBPN model that we have described. In the following, we present several modifications to our basic net structure.

5.1 Production Systems

One area to be considered for modification is in modeling the form of responses to differing Conflict Resolution (CR) Strategies for Production Systems. Production Systems are a common type of Rule-Based System in which the control structure can be mapped into a relatively simple recognize-act paradigm. As stated earlier, these Production Systems typically consist of a set of production rules, written in the form '*antecedent* --> *consequent*' that make up the Production Memory (PM), and a data base of assertions called the Working Memory (WM). The assertions in the WM are called Working Memory Elements (WME).

The production system interpreter is the underlying mechanism that examines the data, and determines the set of valid productions. It then controls the execution of the production system program of rule-firing. The interpreter executes this type of production system by performing a series of recognize-act cycles. Each cycle involves deciding which rule to fire, and then executing the actions (consequents) associated with that rule. Since the data base may change as a result of a rule-firing, different rules may fire on successive cycles.

The Recognize-Act Cycle consists of 3 phases:

- Match - in this phase the antecedents of all productions are matched against the contents of the data base. The conflict set comprises all the rules which are instantiated by the current Working Memory elements.

- Conflict Resolution - in this phase, one of the satisfied production rules in the conflict set is selected for execution. If no production is satisfied (an empty conflict set), the interpreter halts.

- Act - in the final phase of the cycle, actions in the consequent of the production rule are executed. These actions may change the contents of Working Memory. At the end of this phase the cycle begins anew.

5.2 Conflict Resolution Classes

In this section we will describe the Conflict Resolution rules. They have been divided into classes based on the criteria by which they judge an instantiation. All of the rules use one or more of the unique knowledge resources consisting of Production Memory, Working Memory, or the state memory maintained by the interpreter. We will enumerate the most common CR rules used by Production Systems.

5.3 Production Ordering Rules

The Production Ordering (PO) rules use a predetermined priority ordering on the productions. As such they only use PM. We discuss two types of PO rules. In the first rule, the relation of dominance is based upon the order in which rules are entered into the system. The first rule entered dominates all others, the second rule dominates all but the first, etc.

The second PO rule is a dominance relation given by a digraph where the vertices represent productions and the edges represent dominance relations between productions. The first rule is said to be strongly selective since it selects from a completely ordered rule set, whereas the second ordering is much less selective.

As applied to Rule-Based Petri Nets we have established the following criteria for Production Ordering rules. Transitions are to be entered in sequence so that ordering is built into the Transition Rule (TR) number.

Let us consider the first case of PO in which the RBPN has at least two rules that are instantiated. In this situation we select for firing that transition which has been assigned the lowest TR number. For the second case, the RBPN has a corresponding directed graph of TRs so that comparable firings can be made. Again this allows for the same multiplicity of PR choices as described above, since the rules are not completely ordered.

5.4 Conflict Resolution Techniques

In this section we present current strategies for Conflict Resolution along with those which we have developed for RBPN's. It should be noted that our techniques are not restricted to RBPN's, and may be adopted within the general CR strategies.

5.4.1 Alternative Conflict Resolution Rules

Several different approaches to CR have been tried including

choosing:

1) the first rule that matches the context, where first is defined in terms of some explicit linear order of the Rule Base;
2) the highest priority rule, where priority is defined by the programmer according to the demands and characteristics of the task;
3) the most specific rule, that is the rule having the most detailed condition part (antecedent) matching the current context;
4) the rule that refers to the element most recently added to the context (WME);
5) a new rule (that is, an instantiation which has not occurred previously);
6) an arbitrary rule;
7) not to choose a single rule, but run in parallel.

5.4.2 RBPN Techniques for Conflict Resolution

The first two criteria regarding Production Ordering were discussed in Section 5.3 above. We now consider the case of specificity where the rule having the most detailed part is selected. This particular case will lead us to a new, and more specialized,

Conflict Resolution strategy that may be applied to RBPN's. This technique also has implications for general expert systems using production system type rules.

Let us assume we are given a collection of fireable productions (transitions). We enable and fire the production rule which has the largest number of val settings. This case corresponds to the strategy of selecting the production rule which satisfies the most criteria. This strategy differs from the one that selects the rule having the most input places however, in that an individual place may have more than one val setting. When used in the conjunctive form, each of the val settings contributes as an independent criterion to the total number. However, when given a disjunction of val settings in a single place, the contribution is equivalent to that of the single val setting criterion.

Prior to explaining the second type of strategy we need to describe the concepts of *place use*, *precondition set* and *precondition count*. We shall define these concepts using the following set theoretic descriptions:

For each transition t_b, let $P_b = \{ p_{b1}, \ldots, p_{bj} \}$ be the set of all its input places (possibly empty).

For each place p_c, let $T_c = \{ t_{c1}, \ldots, t_{cm} \}$ be the set of all transitions for which p_c is an input place.

The formal definitions follow:

Definition 5.1 Given a RBPN, RB = (P, T, A, **I**, **O**), for each place p_a, with either a standard or data token val setting, let $T_a = \{t_{a1}, \ldots, t_{aj}\}$ be the set of transitions for which place p_a is a precondition (input place).

We call T_a the *Precondition Set* of transitions for place p_a, and $j = |T_a|$, the cardinality of the set, is the *Precondition Count* of p_a.

Definition 5.2 We say u_a is the *Place Use* of transition t_a when the following are satisfied:

Given $P_k = \{p_{k1}, \ldots, p_{kn}\}$ is the set of all input places for t_a. Let $T_{k1}, \ldots, T_{kn}$ be the precondition sets for each $p_{ki} \in P_k$, $i = 1, \ldots, n$. Then we define the place use, u_a, for transition t_a as the cardinality of the union of precondition sets T_{ki}, $i = 1, \ldots, n$.

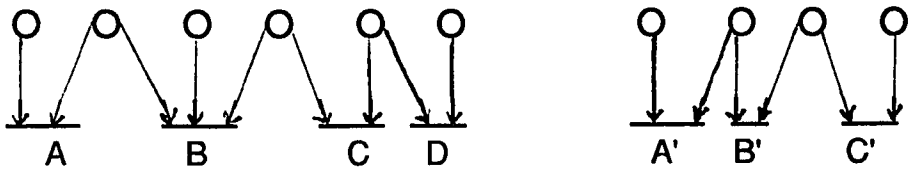
That is, $$u_a = \#\left(\bigcup_{p_{ki} \in P_k} T_{ki} \right)$$

For this second strategy, we select the transition whose *'place use'*

is minimal. In the manner of Priese [30], we define the concept of conflict connectedness which will also be used in CR strategies.

Definition 5.3 Given a RBPN, RB = (P, T, A, **I**, **O**), transitions t and t' are said to be in *conflict* if there is at least one place, p, that is an input to both t and t'; they are *conflict connected* if there exist transitions $t_1, \ldots, t_n$ such that $t = t_1$, $t' = t_n$ and for all i, $1 \leq i < n$, t_i and t_{i+1} are in conflict.

In Figure 5.1a, transitions A and D are said to be conflict connected since we have the transition pairs {A,B}, {B,C}, and {C,D} in conflict. This does not imply however that conflict is a transitive relation, as can be seen in Figure 5.1b. Here we have A' and B' in conflict, along with B' and C' in conflict, without having A' and C' conflict.



(a) (b)

Figure 5.1

We can prove however that Conflict Connectedness is a transitive relation. Given the conflict connected transition pairs t and t', plus t' and t", then t and t" are conflict connected.

Proof: Let $t_1, \ldots, t_j$ be the conflict sequence of transitions, for t and t', such that $t = t_1$ and $t' = t_j$, and let $t_k, \ldots, t_n$ be the conflict sequence for t' and t", such that $t' = t_k$ and $t'' = t_n$. We have $t_j = t_k$ and therefore there is a sequence of transitions $t_1, \ldots, t_n$, such that $t = t_1$ and $t'' = t_n$ and for all i, $1 \leq i < n$, t_i and t_{i+1} are in conflict. This demonstrates that t and t" are conflict connected. We therefore have Conflict Connectedness as as a transitive relation.

5.4.3 Place Use and Place Count Techniques in Conflict Resolution

We can continue to specify with more granularity the conflict resolution process when there are two or more transitions having the same degree of place use. In those cases we enable the transition having a choice place as an output place. As a result, any possible decisions that might be employed are made available to the

user at a time when other transitions are still being considered. In the event that there is more than one such transition (having an output Choice Place), we fire that transition which assigns the maximum Certainty Factor for its Choice. In the event that there is no Choice Place as an output we enable that transition having the greatest number of output place settings. This has the effect of allowing for the maximal number of possible new matchings in the next cycle.

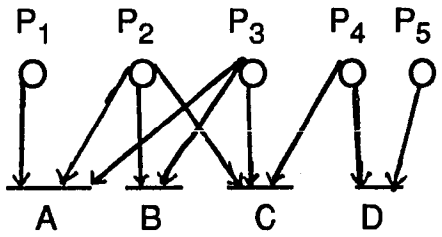


Figure 5.2

We illustrate the function of place use in conflict resolution with the following example. In Figure 5.2 we first evaluate the Precondition Counts as follows:

$P_1 = 1\ \{ t_A \}$; $P_2 = 3\ \{ t_A, t_B, t_C \}$; $P_3 = 3\ \{ t_A, t_B, t_C \}$;

$P_4 = 2\ \{ t_C, t_D \}$; $P_5 = 1\ \{ t_D \}$.

We have the corresponding transition Place Uses given by:

$t_A = 3$; $t_B = 3$; $t_C = 4$; $t_D = 2$.

In Figure 5.2, transitions t_A and t_C have the largest number of input places. In evaluating the choice of which transition to fire when both transitions are firable, we determine their place uses as $t_A = 3$, $\{ t_A, t_B, t_C \}$ and $t_C = 4$, $\{t_A, t_B, t_C, t_D \}$. Since t_A's place use is minimal, the conflict is resolved. Similarly when choosing between t_B and t_D we have place use for $t_B = 3$, $\{ t_A, t_B, t_C \}$ and $t_D = 2$, $\{ t_C, t_D \}$; t_D is selected since its place use is minimal.

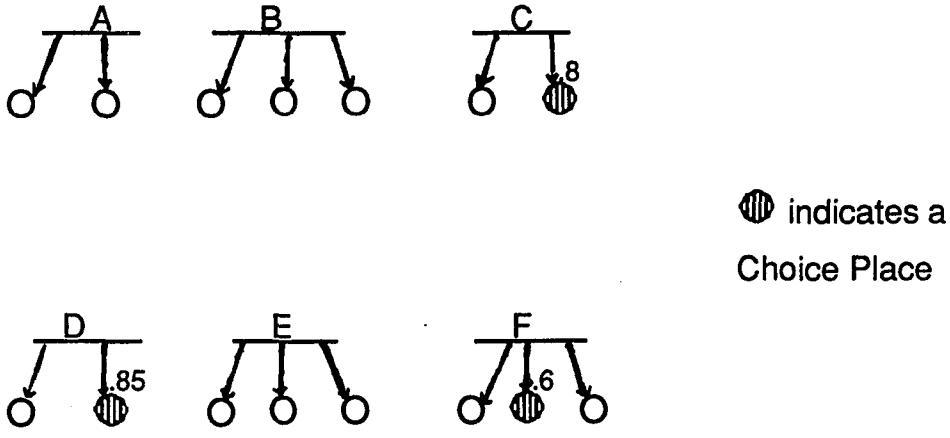


Figure 5.3

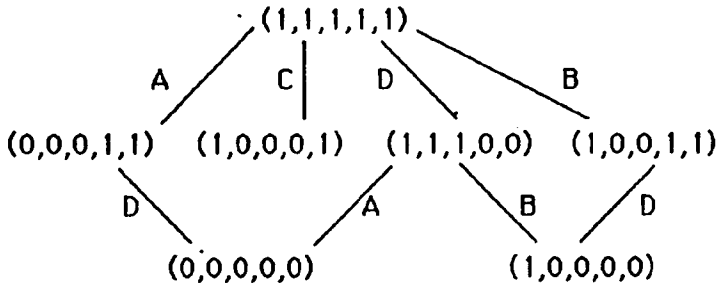
In Figure 5.3, we assume the transitions $\{t_A, t_B, t_C, t_D, t_E, t_F\}$ all have similar input conditions and the firing selection will be output place (consequent) dependent. When the choice is between t_A and t_B, t_B is selected since it has the greater number of output

places (and the other rules do not apply). When selecting between t_D and t_E, t_D is chosen since it contains a choice place in its output set. When given the pair t_C and t_F, t_C is selected due to the larger Certainty Factor value for its choice place. In those instances where the output places do not provide any additional resolution capabilities in the ordering of transition selection, we can filter down to alternative Conflict Resolution procedures or perform a random transition selection. In Figure 5.4, we present Reachability Graphs for the net model pictured in Figure 5.2. We can see that the CR techniques have described the transition sequence A,D which gives a maximal use of place markings in Figure 5.4a.

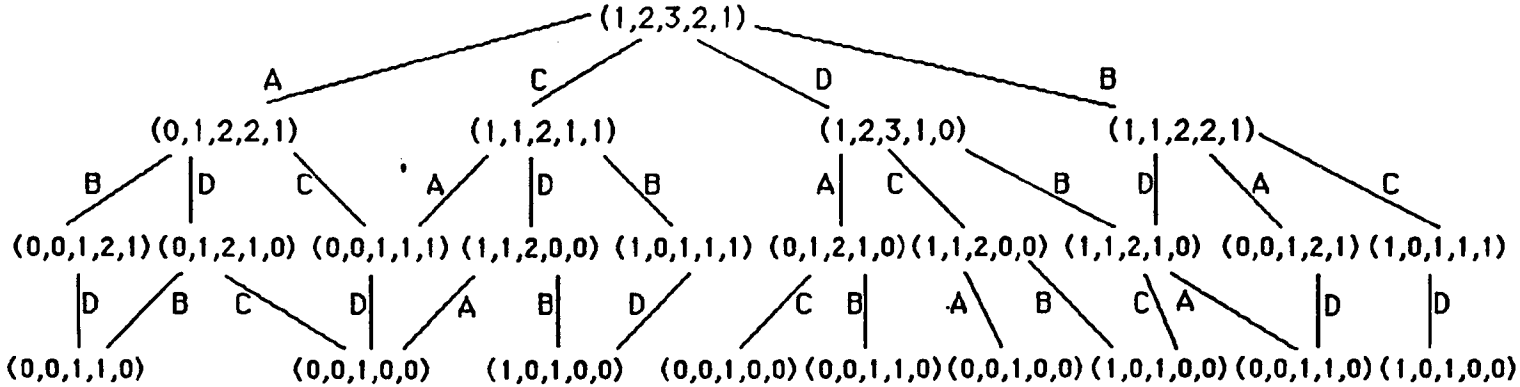
5.5 Additional CR Techniques for RBPN's

As stated earlier, we will demonstrate how RBPN's can model the variety of additional CR methods we have described. We continue to go through the remaining techniques for CR that were described in 5.4.1. We begin with the option of selecting the rule that has the most recently added element. As we have seen, any element that has been requested as an input to a transition, and is to be entered by the user, can be signaled through the use of the entry switch to receive a value. This place inherently becomes the

Reachability Graphs of fig. 5.2



(a)



(b)

Figure 5.4

most recently added element and will result in a transition firing if the appropriate criteria are met. In the event of a non-external place type being the most recently activated place, we propose the use of historical traces containing the current firing transition number whenever an place receives a value. Thereby we have an automatic means of detecting when a change has occurred for comparison purposes. This allows for all places to be related as to the most recent time of activity. It also will be used for transition firing histograms so that transitions can be monitored in order that we may compare two or more transitions for the most recent activity. Obviously a transition that has not been fired will not be found in the transition histogram of firings. This will give us sufficient criteria for the Conflict Resolution technique of firing the most recently enabled transition. This however does not cover the case of instantiations which have not previously occurred. For this CR technique, we shall describe the extension of instantiation variables to RBPN's.

5.6 Instantiation Modeling for RBPN's

Although it is possible to limit our RBPN's to Qualifier Places that have a preset allowable range of attribute values, we consider the extension of instances to a variable. We recall that in the

standard model of a production system when the Left Hand Side (LHS) of a production, containing a conjunction of Condition Elements (CE's), is satisfied the production is eligible to fire, that is, the actions in the Right Hand Side (RHS) may be performed. The RHS may contain a sequence of actions which can result in Working Memory (WM) changes [34]. In OPS5 [8] type systems, these may take the form of creating a new Working Memory Element (WME) and adding it to WM (Make), changing values in a WME (Modify), or deleting a WME from WM (Remove). What we will show is how RBPN's can be extended to mimic the operations contained in these production systems.

We begin by examining the Make and Modify operations. These may correspond to internal settings for a val set attribute. Although val set attributes can be first set via external data entry (query transitions), they may also be set internally for their initial marking. The Make can also be accomplished for a variable type object where a simple data place is assigned a token value. Modify is treated as a special case of Make in which the data val attributes are changed as indicated.

The Remove is a special case of the Modify in which the attributes are all reset to the initial (-1, . . . ,-1) status for a val attribute, but with the query indicator (empty switch) off; it is apparent that the attribute has already been through an initial val setting and is no longer in use. An example of an OPS5 type

production is given in Figure 5.5.

The LHS CE's are allowed to contain variables, the predicate symbols (<, >, =, <=, >=, <>) and others as well as constants. Typically constants must match identically. A variable may match to any value, unless there is more than one variable of the same type on the LHS; in this case, all occurrences of the variable must be identical to the corresponding WME.

```
(p  p1   (C1   ^attr1 <x>      ^attr2   12 )
         (C2   ^attr1 <=15     ^attr2   <x> )
         -(C3  ^attr1 <x> )
   -->
         (remove  2) )
```

Figure 5.5 A Sample OPS5 Production

Chapter 6 Validation of the Expert System

After producing an operational expert system, with rules for the proposed application, we ask the natural question: Is it working correctly? This is one question expert systems have in common with any conventional computer system. In the same sense that a conventional system must be verified, so too must an expert system. It may be said that the need here is even greater. This is due to the fact that most conventional systems are operating from a standard of formalized algorithms functioning within an established business or scientific community. In environments such as these, verification and testing are long, established procedures whose tenets are the bedrock of the total system development process within the organization.

In contrast, expert systems are based upon a collection of heuristics and rules emerging from one or more individual's sense of expertise and practical experiences, perhaps never before having been subjected to an extensive rigorous examination. In addition, most standard systems are used and tested in such a way that when they are not working properly those occurrences can be detected easily. There are of course exceptions, such as those large complex systems which may fail in subtle or unexpected ways, but these too still produce effects that can be labeled as errors. In general,

because of the non-procedural nature of these expert systems, the usual inclusion of bells and whistles is not appropriate. The major question for these systems is: At what point can they be declared correct? We employ several basic requirements that can be used for determining this level of correctness for an expert system. They are consistency, completeness, soundness, precision, and usability. We address each of these issues in turn.

A standard measure for consistency is the degree to which the system results match those of the expert. In any student advisement system it is highly possible for the counselor to recommend one required course over another. In our system we indicate all of the pertinent required courses; the level of consistency then, is not a direct match. Rather consistency here is determined when the expert adviser's course projections form a subset of the computerized recommendations. This is the criterion we have used for consistency. Similarly within the system test process we had cases that differed from each other by one or two responses. We examined these and found our results to be consistent. That is similar input gave us similar output which is what was sought. As an example, two rules should not come to opposing results from the same premise. This alone is not a proof of consistency, since it does not establish that similar cases are treated similarly. By having a choice set classified as a mutually exclusive set, we can further verify consistency. Unfortunately,

choice sets of this type are not always possible. What we have done, however, is generate a graph of the rule set and through analysis of the graph we can determine the paths that exist between the input data and conclusions. With this tool we are able to ensure that similar preconditions do not result in conflicting postconditions.

The meaning of completeness of a system is how well the domain is covered by the knowledge base. In this sense, we have completeness when everything that is a logical consequence in the domain from the given data will be derived. In another sense, completeness requires that the knowledge base be extensive enough to allow the system to handle any problem within its purview. In our system we make allowances for the advice to be incomplete under circumstances that require special attention, for example, career advice, and academic or disciplinary counseling. Also we have considered individual systems so that students may still elect courses outside the limited department areas without specifically having the system acknowledge or allow for those choices. For each major or core area the system must be complete. Even when the area of concentration is undecided, results must be complete as they pertain to college-wide policies. It is possible to work backwards from the potential goals to the generating data, in order to determine that these paths exist, and to create support for those goal choices that are not supported. This was established by the

reachability trees that could be created for the goals and their corresponding coverings.

A system is said to be sound when all derivable choices and goals are valid. In other words, the correct conclusions are reached in relation to expert judgment. In our system, we took an empirical approach and soundness was established through test cases where known outcomes were compared to those generated by the system.

The precision requirement is relevant when there are certainty factors present in the output of the expert system. In these cases we want precision regarding the certainty of any conclusions reached by the system. As such, precision is an extension of the soundness criteria. A system that may be overly confident or overly pessimistic can be tolerated, but one which varies between these two states cannot. Testing for precision means that results fall within a range deemed accurate. This range might be as limited as a few cases: absolute, strong, little, or no certainty. We have established the values of 100 and 0 to be unchangeable, in the sense that once assigned either of these values the certainty factor remains fixed.

Improvements to the precision of the system can be effected through several techniques. First we may adjust whatever algorithm is being used to combine different certainty factors assigned to the same goal. This is the technique we employed in our system. In some cases this technique may have a limited effect

since it may not reflect the sensitivity of the system. This second method of sensitivity analysis allows for variations in the conclusions as a function of variations to the data. How well does the system react to deviations in query responses? If the reactions are insignificant for differing answers, the question is not sensitive, whereas if small changes in the answers lead to large differences in the conclusions we can say the question is very sensitive. By locating these sensitive areas we can fine tune the system to give more than tolerable results for those parameters that have a considerable effect on the outcome.

To maximize usability, we want the interaction between the user and the system to progress along the lines intended by the developer. We want the users to understand what the questions are asking of them, and to respond accordingly. Toward this end we have limited the number of queries that request the user to supply an answer. Instead, we have tried to let the user select from a list (menu) of possible answers. By grouping together answers in this way, we relieve some pressure in the degree of certainty that is asked of the user. The questions take on a more objective meaning rather than calling on the impressions or attitudes of the user in their responses. We want the user interface to be clean and unambiguous. As a result, answers to certain questions must be checked for consistency, which can be domain dependent.

Additionally, we tested the system by allowing users to run

through the prototype system by themselves. We were able to get their reactions and responses to this form of advisement, and to answer any questions when assistance was required. By doing this we were able to access the weak points in the system, and those areas that created difficulty were highlighted and addressed in later versions of the system.

A final point regarding validity of the system pertains to the response time of the system. We didn't want the user to wait more than 15 seconds between any occurrences of system responses. Longer response times are known to create an ineffective system environment, resulting in a loss of involvement on the part of the user. This would defeat one of the major goals of the system, and therefore must be avoided.

Chapter 7 Human Expert Results vs Expert System Output

In this chapter we present a comparison of our prototype expert system with the current human adviser system. We will discuss the results generated by the expert system for student advisement in Data Processing, and those results produced by the expert faculty advisers. We compare the results of the two systems for consistency, accuracy and completeness.

In order to evaluate the efficacy of the expert system, we designed the following experimental procedure: a random group of students majoring in DP were selected for the expert system test after they were pre-registered by their academic adviser. The students' faculty advisement had been done in either a classroom environment or in an office interview by one of several advisers familiar with the DP program as well as the general college curriculum requirements. After their advisement, these students were asked to write down the program of recommended courses they were given. We then asked the same students to interface with the expert system for DP advisement. The results of these two advisement procedures are tabulated in Appendix A. These results are presented with the faculty advisement data followed by the expert system (ASARS) generated advisement responses. We have

also included, in the appendix, several of the response form printouts generated by the expert system. These response forms give a clear picture of the type of data from which the tabular results were culled.

We were able to establish the following results from the prototype testing described above.

1) It was apparent in every instance that the course list of the expert system almost always included the faculty recommended course list as a subset. If there was a course missing the reason for its absence was apparent from the user responses.

2) In many cases the expert system was more definitive in its course selection than the human expert. We found this to be surprising since we thought the human expert would be more definitive in their course selections.

3) The presence of confidence factors in the expert system was not matched by any corresponding feature in the human system. As a result, all course offerings appeared to have the same importance.

4) Student reaction to the expert system was generally positive. Individual sessions lasted only a few minutes with most of the time taken up with printouts of the session. These listings included both course recommendations and the student's responses to the qualifier attribute queries.

APPENDIX A

DESCRIPTIVE INDEX FOR TEST CASE RESULTS:

CASE Number	Course1 Course2 Course3 Course4 Course5 .
special comments	These courses are recommended by the faculty adviser during counseling sessions.
ASARS special comments	These course selections were generated by the computerized expert system. They are further divided into the course areas used by Data Processing majors.

Any course matching a faculty recommendation is followed with an asterisk (*).

The number of Group I-IV courses taken is indicated below the group level.

An X in a group column indicates a course in that area may be taken to fulfill the group requirement.

Special course recommendations and/or requirements are stated in the ASARS comment area.

CF represents the Certainty Factor that was assigned to a recommendation. It is only indicated if the course assignment is questionable.

An @ represents a course generated by the system but not given by the adviser.

COMP refers to a message generated by the system indicating the student has completed the requirement(s) in the area indicated.

2trks, macc, PRE, are abbreviations for system generated messages.

APPENDIX A

TEST CASE RESULTS

CASE#001	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
	ENG 22	DP 54	DP 12	CP 54	
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
	ENG 22 *		1	1	2
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
.DP 12 *			1		
CASE #02	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
	DP 11	MAT M1	HPE 12	ENG 12	
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
CP & MAT	Adv03/04C1	CU=9*MATM1	1, X	X	X
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
MAT M1 PRE	2trksCP,MA		X	HPE 12 *	
CASE #03	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
	DP 54	MAT 13	ENG 22	HIST 11	
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
11yr math	22 *	MA13*CU=37	X	X	HIST 11 *
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
DP 11@	macc		1, X		
CASE #04	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
ACC11wmod	DP 32	ACC 12	DP 55	HPE12	
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
		COMP *	X	X	1
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
DP 32 *	DP 55 *	ACC 12 *	2	HPE 12 *	

CASE #05	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
	DP 32	DP 61	MAT 15	DP56(smod)	
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
		COMP *	2	ENG 78 *	X
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
DP32 *	DP61*, 56 *	COMP	1		

CASE #06	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
	DP 32	CP 54	ENG 78	ART 51	HE 35,33
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
	COMP	COMP	1	1, ENG 78*	1
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
DP 32 *	CP 54 *	COMP	1	*	*

CASE #07	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
	DP 32	DP 54	SOC 31	SPE11smod	ECO12smod
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
	COMP	COMP	SPE 11 *	X	ECO12 *
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
DP 32 *	DP 54 *	COMP	SOC 31 *		

CASE #08	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
	DP 32	ENG 22	SPE 21		
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
	ENG 22 *	COMP	SPE 21 *	1, X	X
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
DP 32 *	2 GROUPS	COMP	X		

CASE #09	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
EN12 C2wm	DP 32	CP 54	ENG 22	FR 2	
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
	ADVISOR	COMP	1, X	1, FR 2 *	X
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
DP 32 *	2 GROUPS	COMP	X		

CASE #10	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
	DP 32	DP 54	PSY 33	CP 54	
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
GRP I-IV	COMP	COMP	1		2
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
DP 32 *	DP54*CP54*	COMP	1	PSY 33 *	

CASE #11	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
CP -> MA	DP 54	DP 55	DP 56	ACC 12	
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
new curric	COMP	COMP	1,X	1,X	1,X
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
DP 54 *	DP32not req	ACC 12 *	1,X		

CASE #12	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
	DP 32	CP 54	ENG 22		
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
1 of a GRP	ENG 22 *	COMP	1, X	1, X	1, X
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
DP 32 *	CP 54 *	COMP	1, X		

CASE #13	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
TRANSFER	ENG 22	DP 54	DP 12	CP 54	
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
	ENG 22 *	COMP	1	1	1
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
DP 12 *	CP54*DP54*		2		

CASE #014	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
failed DP12	DP 61	ACC 11	MAT 14	ENG 12	SPA 1
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
Micro-appl	ENG12 *REP	MAT 14 *	2	1,*	*
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
DP 61 *	DP12msg*	ACC 11 *	*		

CASE #015	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
DP 54 COMP	ACC 12	CP 30	DP 55	DP 56	
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
DP 12 'D'	COMP	COMP	Grp COMP 1	2	
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
DP54*CF=.8	DP31,CF=.7	ACC 12 *	1		

CASE #016	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
	DP 12	CP 30	ENG 22	PSY 11	MUS 31
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
Micro-appl	ENG 22 *	COMP	MUS 31 *	X	X
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
DP 12 *			PSY 11 *		

CASE #017	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
DP 54 COMP	DP 55	CP 54	MAT 13	CP 30	
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
DP11 C+msg	COMP	MAT 13 *	Grp COMP 1	2	1
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
DP 55 *	CP 54 *	COMP	1		

CASE #018	COURSE #1	COURSE #2	COURSE #3	COURSE #4	COURSE #5
LA major	ART 31	ENG 42			
ASARS	ENGLISH	MATH	GROUP I	GROUP II	GROUP III
Jun grad	COMP	COMP	1,*	1, ENG42 *	
DATA PROC I	DATA PROC 2	ACCTG	GROUP IV	ELECTIVE	ELECTIVE
		COMP	1		

```
1      DP 11 recommended  :10/10
2      DP Majors are required to take ONE course from Math 11 (Micro-
         Application Group only), 13 or 14, and BOTH ACC 11 and 12.  :10/10
3      Group requirements (I - IV) have been met. Additional electives may be
         taken.  :10/10
4      HPE 12 may be taken  :10/10
5      ENG 22 may be taken  :10/10
  score on CUNY Math exam enter zero '0' if not taken = 11.000000
  the number of Group I courses passed = 0.000000
  the number of Group II courses passed = 1.000000
  The number of Group IV courses passed = 1.000000
  the number of Group III courses passed = 2.000000
```

SAMPLE
OUTPUT

The Last Math Course Taken or CUNY result was MATH M1

Last DP course taken was NONE at this time

Major field is DATA PROCESSING

your status with last KCC math course is PASSED

Number of courses PASSed in Foreign Language, Literature or Philosophy is ONE course.

Number of courses PASSed in History, Economics, or Political Science is TWO or MORE courses

Number of courses PASSed in Anthropology, Psychology, or Sociology is ONE course

Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE ARTS is NONE as of this semester

Has HPE 12 been completed: NO

Last English course PASSed was ENG C2 or 12

MATHCHK = 2.000000

NEXTDP = 0.000000

CUNYTEST = 11.000000

ADVISE = 2.000000

GROUPREQ = 4.000000

GRP1 = 0.000000

GRP2 = 1.000000

GRP4 = 1.000000

GRP3 = 2.000000

GRPCT = 3.000000

SAMPLE
OUTPUT

APPENDIX B

Rule-Based Petri Net Model

EXPERT SYSTEM FOR DATA PROCESSING ADVISEMENT
QUALIFIER ATTRIBUTES:-

Q1[9] Last MATH course taken was:
1) NONE and NO CUNY EXAM 2) MAT M1 3) MAT M2 4) MAT R2
5) PASSED CUNY EXAM 6) Failed CUNY score below 15
7) MAT 03 8) MAT 10, 11, or 12 9) MAT 13 or above

Q2[5] Last DP course taken was:
1) NONE 2) DP11 3) DP12 4) DP31
5) DP13 or numbered above 31

Q3[3] Major field is:
1) Data Processing 2) Computer Science
3) Other or undecided

Q4[3] Advisement status is:
1) Not advised 2) In progress 3) Completed

Q5[3] Mathematics Evaluation is:
1) Not started 2) In progress 3) Done

Q6[3] Your status with last KCC MATH course is:
1) PASSED 2) NONE TAKEN 3) FAILED or R

Q7[4] Last grade in a DP course was:
1) PASSED with 'C' or higher 2) INC or W
3) FAILED 4) PASSED with 'D'

Q8[4] Status of Computer Science courses is:
1) NONE taken yet 2) CS 12 passed with 'C' or better
3) CS 12 passed with 'D' 4) CS 12 with 'INC', 'W', or 'F'

Q9[2] MAT 15 grade was:
1) Passing 2) Failing or W or INC

Q10[3] Last Accounting course Passed was:
1) NONE 2) ACC 11 3) ACC 12

Q11[3] Number of GROUP II (For. Language, Literature and Philosophy) Courses passed:
1) ONE 2) TWO or more 3) NONE as of yet

Q12[3] Number of GROUP III (Econ, History and Political Science) Courses passed:
1) ONE 2) TWO or more 3) NONE as of yet

Q13[3] Number of GROUP IV (Anthropology, Psychology, and Sociology) Courses passed:
1) ONE 2) TWO or more 3) NONE as of yet

Q14[3] Number of GROUP I (Non-studio Art, Music, Speech and Theater Arts) Courses passed:
1) ONE 2) TWO or more 3) NONE as of yet

Q15[2] HPE 12 Completed:
1) YES b) NO

Q16[5] Last ENGLISH course Passed was:
1) ENG 03/04/C1 2) ENG C2 or 12
3) ENG 22 or ABOVE 4) NONE PASSed
5) NONE Taken or NO ENGLISH CUNY Placement

Q17[4] Follow up area is:
1) DP 2) CS 3) Other 4) Undecided

Q18[6] Last H.S. Math Course Passed was:
1) 9th yr or elementary algebra 2) 10th yr or geometry
3) 11th yr algebra only 4)11th yr with trigonometry
5)12th yr 6) none passed or don't recall

Q19[8] Electives taken are:
1) DP 54 2) DP 61 3) DP 63 or 55 4) CP 35
5) CP 53 6) CP 54 7) DP 13 8) None of these

RULE EXPRESSION VARIABLES USED BY THE EXPERT SYSTEM

Machk- variable indicating the mathematics course standing and status the of the student.

Adv- variable indicating status within advisement session.

nexDP- variable of student's status within the DP curricula.

Grp1, Grp2, Grp3, Grp4- variables indicating the student's status within the Group Requirements.

Gct- variable for the number of Groups areas checked

Greq- variable indicating status relative status of all Groups

EXPERT SYSTEM FOR DP ADVISEMENT : CHOICE PLACE DATA

C1: DP 11 Recommended

C2: DP 12 Recommended

C3: DP 31 Recommended

C4: DP 32 Recommended

C5: DP 41 Recommended

C6: DP 61

C7: DP 63

C8: No DP until MAT M1 Passed

C9: DP 54 Microcomputer Applications Group Recommended

C10: Student may continue DP only upon completion of DP 11

C11: All INC grades must be removed this semester: DP registration by dept. approval only.

C12: The student must pass DP 12 to continue with DP (Comp.Prog) Group.

C13: DP majors must take one from MAT 11,13,14, and Both ACC11 and 12.

C14: Requirements in CS include Calculus and Diff. Eq.

C15: DP (Comp Prog) req's include DP 31, 32, one of DP 54, 61, 63, and one of DP 13, 41, CP 35, 53, or 54.

C16: A major in DP should be considered.

C17: CS 12 may only be taken with MAT 15

C18: CS 12 must be passed before going on in CS

C19: CS 13 has MAT 16 as a corequisite.

C20: Assistance for your curriculum is not available at this time.

C21: The Failed course must be passed to satisfy the curriculum req's.

C22: DP 54 (Microcomputer Applications)

C23: CP 35 (APL)

C24: CP 53 (PL/I)

C25: ONE from DP 13, 41, or CP 35, 53, or 54.

C26: ONE from DP 54, 61, or 63.

C27: A grade of 'C' or better in DP 11 is required to continue in the Computer Programming Concentration.

C28: ACC 11 followed by ACC 12

C29: ACC 12

C30: DP (Comp. Prog.) majors must take MAT 13 or 14.

C31: DP has two group areas: Computer Programming and Microcomputer Applications

C32: Group requirements (I-IV) have been met. Additional electives may be taken.

C33: Student may choose a course from one of the 4 areas to satisfy Group Requirements.

C34: A Group III (History, Eco., Pol.Sci.) course may be taken.

C35: A Group IV (Anthro., Psych., Sociol.) course may be taken.

C36: A Group II (For.Lang., Lit., Philosophy) course may be taken.

C37: A Group I (non-studio Art, Music, Speech, Theater.Arts) course may be taken.

C38: HPE 12 may be taken.

C39: ENG 22 may be taken.

C40: ENGLISH requirement completed.

C41: SEE English Department to determine placement.

C42: Repeat FAILED English course.

C43: MAT M1 based upon CUNY exam.

C44: Repeat MAT M1.

C45: MAT R2 may be taken.

C46: Sign-up in Math Lab to get assistance for CUNY exam.

C47: Mathematics requirement is fulfilled.

C48: MAT 03 may be taken.

Transition Rules for the Expert Advisement System in DP

TR1	Q3[3](1)	=> Q17[4](1)
TR2	Q2[5](1) ∧ Q17[4](1)	=> Machk=1 ∧ Adv=1
TR3	~Q2[5](1) ∧ Q17[4](1)	=> Machk=2 ∧ Adv=1 ∧ nexDP=1
TR4	Q6[3](1) ∧ Machk=1 ∧ Adv=1	=>Q5[3](2) ∧ Machk=2 ∧ Adv=2 ∧ (C1,100) ∧ (C13,100)
TR5	Q6[3](2) ∧ Machk=1 ∧ Adv=1 ∧ ICUNY < 15	=>Q5[3](3) ∧ Machk=2 ∧ Adv=2 ∧ (C8,100) ∧ (C31,100) ∧ (C43,100)
TR6	Q2[5](2) ∧ Q7[4](1) ∧ nexDP=1	=> Adv=2 ∧ (C2,100)
TR7	Q2[5](2) ∧ Q7[4](4) ∧ nexDP=1	=> Adv=2 ∧ (C9,90) ∧ (C13,100) ∧ (C27,100)
TR8	Q2[5](2) ∧ Q7[4](2,3) ∧ nexDP=1	=> Adv=2 ∧ (C10,100)
TR9	Q2[5](3) ∧ Q7[4](2) ∧ nexDP=1	=> Adv=2 ∧ (C11,100) ∧ (C12,100)
TR10	Q6[3](3) ∧ Q1[7](2) ∧ Machk=1 ∧ Adv=1 ∧ ICUNY < 15	=> Adv=2 ∧ Machk=2 ∧ Q5[3](3) ∧ (C8,100) ∧ (C9,70) ∧ (C13,100)∧ (C44,100)

TR11	Q2[5](3) ∧ Q7[4](3) ∧ nexDP=1	=> Adv=2 ∧ (C9,80) ∧ (C12,90)
TR12	Q6[3](3) ∧ ~Q1[7](9) ∧ Machk=1	=> Machk=2 ∧ Adv=2 ∧ (C13,100)∧(C9,100) ∧ (C1,100) ∧ (C21,90)
TR13	Q6[3](3) ∧ Q1[7](9) ∧ Machk=1 ∧ Adv=1	=> Machk=2 ∧ Adv=2 ∧ (C1,90) ∧ (C2,90) ∧ (C13,100)
TR14	Q3[3](2) ∧ Q8[4](1) ∧ 15 ≤ ICUNY ∧ ICUNY ≤ 32	=> Adv=2 ∧ (C14,100) ∧ (C16,80)
TR15	Q3[3](2) ∧ Q8[4](1) ∧ ICUNY < 15	=> Adv=2 ∧ (C14,100) ∧ (C16,70) ∧ (C9,90) ∧ (C8,100)
TR16	Q3[3](2) ∧ Q8[4](1) ∧ ICUNY > 32	=> Adv=2 ∧ (C14,100) ∧ (C17,100)
TR17	Q3[3](2) ∧ Q8[4](4)	=> Adv=2 ∧ (C14,100) ∧ (C16,90) ∧ (C18,100)
TR18	Q3[3](2) ∧ Q8[4](2) ∧ Q9[2](1)	=> Adv=2 ∧ (C14,100) ∧ (C19,100)
TR19	Q3[3](2) ∧ Q8[4](3) ∧ Q9[2](1)	=> Adv=2 ∧ (C14,100) ∧ (C16,80) ∧ (C19,70)
TR20	Q3[3](2) ∧ Q8[4](2,3) ∧ Q9[2](2)	=> Adv=2 ∧ (C14,100) ∧ (C16,90) ∧ (C19,100)
TR21	Q3[3](3)	=> Adv=2 ∧ (C20,100)

TR22	Q2[5](4) ∧ Q7[4](2) ∧ nexDP=1	=> Adv=2 ∧ (C11,100) ∧ (C15,100)
TR23	Q2[5](4) ∧ Q7[4](3) ∧ nexDP=1	=> Adv=2 ∧ (C21,100) ∧ (C9,90) ∧ (C3,70) ∧ (C15,100)
TR24	Q2[5](5) ∧ Q10[3](1) ∧ nexDP=1	=> Adv=2 ∧ (C13,100) ∧ (C15,100) ∧ (C28,90)
TR25	Q2[5](5) ∧ Q10[3](2) ∧ nexDP=1	=> Adv=2 ∧ (C13,100) ∧ (C15,100) ∧ (C29,90)
TR26	Q2[5](5) ∧ Q10[3](3) ∧ nexDP=1	=> Adv=2 ∧ (C15,100) ∧ (C30,100)
TR27	Q2[5](3) ∧ Q7[4](1) ∧ Q10[3](1) ∧ nexDP=1	=> Adv=2 ∧ (C3,100) ∧ (C15,100) ∧ (C28,90) ∧ (C30,100)
TR28	Q2[5](3) ∧ Q7[4](1) ∧ Q10[3](2) ∧ nexDP=1	=> Adv=2 ∧ (C3,100) ∧ (C15,100) ∧ (C29,90) ∧ (C30,100)
TR29	Q2[5](3) ∧ Q7[4](1) ∧ Q10[3](3) ∧ nexDP=1	=> Adv=2 ∧ (C3,100) ∧ (C15,100) ∧ (C30,100)
TR30	Q2[5](4) ∧ Q7[4](1) ∧ Q10[3](1) ∧ nexDP=1	=> Adv=2 ∧ (C4,100) ∧ (C28,100) ∧ (C30,100) ∧ (C25,80) ∧ (C26,90)
TR31	Q2[5](4) ∧ Q7[4](1) ∧ Q10[3](2) ∧ nexDP=1	=> Adv=2 ∧ (C4,100) ∧ (C29,100) ∧ (C30,100) ∧ (C25,80) ∧ (C26,90)

TR32	Q2[5](4) ∧ Q7[4](1) ∧ Q10[3](3) ∧ nexDP=1	=> Adv=2 ∧ (C4,100) ∧ (C30,100) ∧ (C25,80) ∧ (C26,90)
TR33	Q2[5](4) ∧ Q7[4](4) ∧ Q10[3](1) ∧ nexDP=1	=> Adv=2 ∧ (C5,90) ∧ (C9,90) ∧ (C15,100) ∧ (C28,80) ∧ (C4,100) ∧ (C30,100)
TR34	Q2[5](4) ∧ Q7[4](4) ∧ Q10[3](2) ∧ nexDP=1	=> Adv=2 ∧ (C5,90) ∧ (C9,90) ∧ (C15,100) ∧ (C29,100) ∧ (C4,100) ∧ (C30,100)
TR35	Q2[5](4) ∧ Q7[4](4) ∧ Q10[3](3) ∧ nexDP=1	=> Adv=2 ∧ (C5,90) ∧ (C9,90) ∧ (C15,100) ∧ (C4,100) ∧ (C30,100)
TR36	Q2[5](3) ∧ Q7[4](4) ∧ Q10[3](1) ∧ nexDP=1	=> Adv=2 ∧ (C3,70) ∧ (C5,70) ∧ (C9,80) ∧ (C13,100) ∧ (C28,80)
TR37	Q2[5](3) ∧ Q7[4](4) ∧ Q10[3](2) ∧ nexDP=1	=> Adv=2 ∧ (C3,70) ∧ (C5,70) ∧ (C9,80) ∧ (C13,100) ∧ (C29,100)
TR38	Q2[5](3) ∧ Q7[4](4) ∧ Q10[3](3) ∧ nexDP=1	=> Adv=2 ∧ (C3,70) ∧ (C5,70) ∧ (C9,80) ∧ (C13,100)
TR39	Q6[3](2) ∧ Machk=1 ∧ Adv=1 ∧ ICUNY > 24	=> Q5[3](2) ∧ Machk=2 ∧ Adv=2 ∧ (C1,100) ∧ (C13,100)

TR40	Q6[3](2) ∧ Machk=1 ∧ Adv=1 ∧ ICUNY < 25 ∧ ICUNY ≥ 20	=> Q5[3](3) ∧ Adv=2 ∧ Machk=2 ∧ (C1,100) ∧ (C13,100) ∧ (C31,100) ∧ (C46,100)
TR41	Q6[3](2) ∧ Machk=1 ∧ Adv=1 ∧ ICUNY < 20 ∧ ICUNY ≥ 15	=> Q5[3](3) ∧ Adv=2 ∧ Machk=2 ∧ (C1,100) ∧ (C13,100) ∧ (C31,100) ∧ (C45,100)
TR42	Q11[3](1) ∧ Adv=2	=> Grp2=1 ∧ Gct=Gct+1 ∧ Greq=Greq+1
TR43	Q11[3](2) ∧ Adv=2	=> Grp2=2 ∧ Gct=Gct+1 ∧ Greq=Greq+2
TR44	Q13[3](1) ∧ Adv=2	=> Grp4=1 ∧ Gct=Gct+1 ∧ Greq=Greq+1
TR45	Q13[3](2) ∧ Adv=2	=> Grp4=2 ∧ Gct=Gct+1 ∧ Greq=Greq+2
TR46	Q12[3](1) ∧ Adv=2	=> Grp3=1 ∧ Gct=Gct+1 ∧ Greq=Greq+1
TR47	Q12[3](2) ∧ Adv=2	=> Grp3=2 ∧ Gct=Gct+1 ∧ Greq=Greq+2
TR48	Q14[3](1) ∧ Adv=2	=> Grp1=1 ∧ Gct=Gct+1 ∧ Greq=Greq+1
TR49	Q14[3](2) ∧ Adv=2	=> Grp1=2 ∧ Gct=Gct+1 ∧ Greq=Greq+2

TR50 ~Q3[3](2) ∧ Greq > 4 => (C32,100)

TR51 ~Q3[3](2) ∧ Greq = 4 ∧ Gct = 4 => (C33,90)

TR52 Q3[3](2) ∧ Greq = 4 ∧ Gct = 4 => (C32,100)

TR53 Q14[3](2) ∧ Q11[3](2) ∧ Greq = 4 =>(C34,100)∧(C35,100)

TR54 Q14[3](2) ∧ Q12[3](2) ∧ Greq = 4 =>(C36,100)∧(C35,100)

TR55 Q14[3](2) ∧ Q13[3](2) ∧ Greq = 4 =>(C36,100)∧(C34,100)

TR56 Q11[3](2) ∧ Q12[3](2) ∧ Greq = 4 => (C37,100)∧(C35,100)

TR57 Q11[3](2) ∧ Q13[3](2) ∧ Greq = 4 =>(C37,100)∧(C34,100)

TR58 Q12[3](2) ∧ Q13[3](2) ∧ Greq = 4 => (C37,100)∧(C36,100)

TR59 Q14[3](1) ∧ Q13[3](1) ∧ ~Q3[3](2) => (C32,100)
∧ Greq=4 ∧ Gct=3

TR60 Q14[3](1) ∧ Q12[3](1) ∧ ~Q3[3](2) => (C32,100)
∧ Greq=4 ∧ Gct=3

TR61 Q14[3](1) ∧ Q11[3](1) ∧ ~Q3[3](2) => (C32,100)
∧ Greq=4 ∧ Gct=3

TR62 Q13[3](1) ∧ Q12[3](1) ∧ ~Q3[3](2) => (C32,100)
∧ Greq=4 ∧ Gct=3

TR63 Q13[3](1) ∧ Q11[3](1) ∧ ~Q3[3](2) => (C32,100)
∧ Greq=4 ∧ Gct=3

TR64 Q12[3](1) ∧ Q11[3](1) ∧ ~Q3[3](2) => (C32,100)
∧ Greq=4 ∧ Gct=3

TR65 Q3[3](1) ∧ Greq = 4 ∧ Gct = 3 => (C37,100) ∧ (C36,100)
∧ (C34,100)∧(C35,100)

TR66 Grp1 = 2 ∧ Gct = 1 => (C36,100) ∧ (C34,100)
∧ (C35,100)

TR67 Grp2 = 2 ∧ Gct = 1 =>(C37,100) ∧ (C34,100) ∧ (C35,100)

TR68 Grp3 = 2 ∧ Gct = 1 =>(C37,100) ∧ (C36,100) ∧ (C35,100)

TR69 Grp4 = 2 ∧ Gct = 1 => (C37,100) ∧ (C36,100) ∧ (C34,100)

TR70 Greq=3 ∧ Gct=2 ∧ => (C34,100) ∧ (C35,100)
Q14[3](1,2) ∧ Q11[3](1,2)

TR71 Greq=3 ∧ Gct=2 ∧ => (C36,100) ∧ (C35,100)
Q14[3](1,2) ∧ Q12[3](1,2)

TR72 Greq=3 ∧ Gct=2 ∧ => (C36,100) ∧ (C34,100)
Q14[3](1,2) ∧ Q13[3](1,2)

TR73 Greq=3 ∧ Gct=2 ∧ => (C37,100) ∧ (C35,100)
Q11[3](1,2) ∧ Q12[3](1,2)

TR74 Greq=3 ∧ Gct=2 ∧ => (C37,100) ∧ (C34,100)
Q11[3](1,2) ∧ Q13[3](1,2)

TR75 Greq=3 ∧ Gct=2 ∧ => (C37,100) ∧ (C36,100)
Q12[3](1,2) ∧ Q13[3](1,2)

TR76 Q15[2](2) => (C38,100)

TR77	Q16[5](2)	=> (C39,100)
TR78	Q16[5](3)	=> (C40,100)
TR79	Q16[5](4)	=> (C42,100)
TR80	Q16[5](1,5)	=> (C41,100)
TR81	~Q3[3](1) ∧ Gct=1 ∧ Grp1=1	=> (C36,100) ∧ (C34,100) ∧ (C35,100)
TR82	~Q3[3](1) ∧ Gct=1 ∧ Grp2=1	=> (C37,100) ∧ (C34,100) ∧ (C35,100)
TR83	~Q3[3](1) ∧ Gct=1 ∧ Grp3=1	=> (C37,100) ∧ (C36,100) ∧ (C35,100)
TR84	~Q3[3](1) ∧ Gct=1 ∧ Grp4=1	=> (C37,100) ∧ (C36,100) ∧ (C34,100)
TR85	Q3[3](1) ∧ Gct=2 ∧ Greq=2	=>(C37,10) ∧ (C36,100) ∧ (C34,100) ∧ (C35,100)
TR86	~Q3[3](1) ∧ Gct=2 ∧ Q14[3](3) ∧ Q13[3](3)	=> (C37,100) ∧ (C35,100)
TR87	~Q3[3](1) ∧ Gct=2 ∧ Q14[3](3)) ∧ Q12[3](3)	=> (C37,100) ∧ (C34,100)
TR88	~Q3[3](1) ∧ Gct=2 ∧ Q14[3](3) ∧ Q11[3](3)	=> (C37,100) ∧ (C36,100)
TR89	~Q3[3](1) ∧ Gct=2 ∧ Q13[3](3)	=> (C34,100) ∧ (C35,100)

∧ Q12[3](3)

TR90 ~Q3[3](1) ∧ Gct=2 ∧ Q13[3](3) ∧ Q11[3](3) => (C36,100) ∧ (C35,100)

TR91 ~Q3[3](1) ∧ Gct=2 ∧ Q12[3](3) ∧ Q11[3](3) => (C36,100) ∧ (C34,100)

TR92 Gct=2 ∧ Q12[3](3) ∧ Q11[3](3) ∧ Q13[3](3) ∧ Q14[3](3) => (C37,100) ∧ (C36,100) ∧ (C34,100)∧(C35,100)

TR93 Q6[3](1) ∧ Q17[4](1) ∧ Q1[9](9) => Q5[3](3) ∧ (C47,100)

TR94 Q6[3](1) ∧ Q17[4](1) ∧ Q1[9](3,4) => Q5[3](3) ∧ (C48,100)

TR95 Q6[3](1) ∧ Q17[4](1) ∧ Q1[9](7) => Q5[3](3) ∧ (C13,100)

TR96 ICUNY ≥ 25 ∧ Q17[4](1) ∧ Q18[6](1,2) => Q5[3](3) ∧ (C48,90)

TR97 ICUNY ≥ 25 ∧ Q17[4](1) ∧ Q18[6](3,4,5) => Q5[3](3) ∧ (C13,90) ∧ (C15,100)

TR98 ~Q2[5](1) ∧ Q19[8](1,2,3) ∧ ~Q19[8](4,5,6,7) => (C25,100)

TR99 ~Q2[5](1) ∧ ~Q19[8](1,2,3) ∧ Q19[8](4,5,6,7) => (C26,100)

TR100 ~Q2[5](1) ∧ Q19[8](8) => (C15,100)

RULES IN EXPLICIT FORMAT

Subject:
DP Advisement for Computer Programming and Microcomputer Applications Groups

Author:
Gordon Bassen

Starting text:
Advisement for the Computer Programming and Microcomputer Application Curriculums in Data Processing. It is important that the student have their score on the CUNY Mathematics Assessment Exam (CUNYTEST), as well as, course numbers and grades of pertinent Data Processing and Mathematics classes. A Grade of R in MAT M1, M2, or R2 is not to be considered a Passing Grade. For Group Requirement (I-IV) evaluation the student must know all group courses passed. It is not necessary to know which group they belong to, only the subject area(i.e.SPEECH, HISTORY, etc.) is needed. Studio courses in ART and MUSIC do not fulfill group requirements.

Ending text:
The following information is meant to be used only as a guideline and is NOT to be construed as an actual registration for classes or an obligatory assignment to specific courses. Registration can only be completed with the approval of a counselor or faculty advisor.

Uses all applicable rules in data derivations.

RULES:

RULE NUMBER: 1

IF:
Major field is DATA PROCESSING

THEN:
Follow up area is DP

RULE NUMBER: 2

IF:
Follow up area is DP
and Last DP course taken was NONE at this time

THEN:
[MATHCHK] IS GIVEN THE VALUE 1
and [ADVISE] IS GIVEN THE VALUE 1

NOTE:
Beginning of advisement period

```
RULE NUMBER: 3

IF:
        Follow up area is DP
  and   Last DP course taken was NOT NONE at this time


THEN:
        [MATHCHK] IS GIVEN THE VALUE 2
  and   [NEXTDP] IS GIVEN THE VALUE :
  and   [ADVISE] IS GIVEN THE VALUE 1


NOTE:
  Check for having taken a prior DP course, and reduce advice
  accordingly.

----------------------------------------


RULE NUMBER: 4

IF:
        [MATHCHK] = 1
  and   your status with last KCC math course is PASSED
  and   [ADVISE] = 1


THEN:
        [MATHCHK] IS GIVEN THE VALUE 2
  and   DP 11 recommended - Probability= 10/10
  and   [ADVISE] IS GIVEN THE VALUE 2
  and   DP Majors are required to take ONE course from Math 11 (Micro-
        Application Group only), 13 or 14, and BOTH ACC 11 and 12. -
        Probability= 10/10
  and   Mathematics evaluation is IN PROGRESS


NOTE:
  Student put in DP 11 since they have passed some math and no DP was
  taken.
```

```
RULE NUMBER: 5

IF:
        [MATHCHK] = 1
  and   [CUNYTEST] < 15
  and   your status with last KCC math course is HAVEN'T TAKEN ANY
  and   [ADVISE] = 1


THEN:
        [MATHCHK] IS GIVEN THE VALUE 2
  and   No DP until MAT M1 Passed - Probability= 10/10
  and   [ADVISE] IS GIVEN THE VALUE 2
  and   DP has 2 group areas: Computer Programming or Microcomputer
        Applications - Probability= 10/10
  and   MAT M1 based on CUNY exam - Probability= 10/10
  and   Mathematics evaluation is DONE


NOTE:
  It is suggested that the student consider the curriculum in
  microcomputers.


----------------------------------------




RULE NUMBER: 6

IF:
        [NEXTDP] = 1
  and   Last DP course taken was DP 11
  and   Last grade in a DP course  was PASSED with 'C' or Higher


THEN:
       DP 12 recommended - Probability= 10/10
  and   [ADVISE] IS GIVEN THE VALUE 2


NOTE:
  Going from DP 11 to DP 12
```

```
RULE NUMBER: 7

IF:
        [NEXTDP] = 1
  and   Last DP course taken was DP 11
  and   Last grade in a DP course  was Passed with 'D'


THEN:
        [ADVISE] IS GIVEN THE VALUE 2
  and   A grade of C or better in DP 11 is required to continue in this
        concentration (Computer Prog) - Probability= 10/10
  and   DP 54 - Micro-computer Applications group is recommended. -
        Probability=  9/10
  and   DP Majors are required to take ONE course from Math 11 (Micro-
        Application Group only), 13 or 14, and BOTH ACC 11 and 12. -
        Probability= 10/10


NOTE:
  This student is referred to the program in microcomputer applications.
  Additional advisement may be required.


REFERENCE:
  Current requirements are a Grade of C or better to continue in this
  concentration (Computer Programming).

----------------------------------------




RULE NUMBER: 8

IF:
        [NEXTDP] = 1
  and   Last DP course taken was DP 11
  and   Last grade in a DP course  was INComplete or Withdrawal or Failed


THEN:
        [ADVISE] IS GIVEN THE VALUE 2
  and   Student may only continue in DP upon completion of DP 11; See counselor
        for further assistance. - Probability= 10/10


NOTE:
  Data processing (Computer Programming) majors must complete DP 11
  before continuing any further.


REFERENCE:
  See counselor for assistance.
```

```
RULE NUMBER: 9

IF:
        [NEXTDP] = 1
  and   Last DP course taken was DP 12
  and   Last grade in a DP course  was INComplete or Withdrawal


THEN:
        [ADVISE] IS GIVEN THE VALUE 2
  and   The student must PASS DP 12 to continue with DP (Computer Programming)
        group - Probability= 10/10
  and   All INC grades must be removed this semester; DP registration by
        departmental approval only. Mat 13 or 14 is required of majors. -
        Probability= 10/10


NOTE:
  INC must be removed from record or it will be replaced by a grade of
  `F`.

----------------------------------------



RULE NUMBER: 10

IF:
        [CUNYTEST] <15
  and   [MATHCHK] = 1
  and   [ADVISE] =1
  and   your status with last KCC math course is FAILED
  and   The Last Math Course Taken or CUNY result was MATH M1


THEN:
        [MATHCHK] IS GIVEN THE VALUE 2
  and   [ADVISE] IS GIVEN THE VALUE 2
  and   No DP until MAT M1 Passed - Probability= 10/10
  and   DP 54 - Micro-computer Applications group is recommended. -
        Probability=  7/10
  and   Repeat MATH M1 - Probability= 10/10
  and   DP Majors are required to take ONE course from Math 11 (Micro-
        Application Group only), 13 or 14, and BOTH ACC 11 and 12. -
        Probability= 10/10
  and   Mathematics evaluation is DONE


NOTE:
  Mathematically weak student should be directed away from DP entirely
  (FAILED M1).
```

```
RULE NUMBER: 11

IF:
      [NEXTDP] = 1
  and Last DP course taken was DP 12
  and Last grade in a DP course  was Failed


THEN:
      [ADVISE] IS GIVEN THE VALUE 2
  and The student must PASS DP 12 to continue with DP (Computer Programming)
      group - Probability=  9/10
  and DP 54 - Micro-computer Applications group is recommended. -
      Probability=  8/10


NOTE:
  Most students do not repeat DP 12 after an `F` is recieved.  These
  students should be directed toward the MICROCOMPUTER APPLICATIONS
  Major.

----------------------------------------



RULE NUMBER: 12

IF:
      [MATHCHK] = 1
  and your status with last KCC math course is FAILED
  and The Last Math Course Taken or CUNY result was NOT MATH 13 or HIGHER


THEN:
      [MATHCHK] IS GIVEN THE VALUE 2
  and [ADVISE] IS GIVEN THE VALUE 2
  and DP 11 recommended - Probability= 10/10
  and DP 54 - Micro-computer Applications group is recommended. -
      Probability= 10/10
  and DP Majors are required to take ONE course from Math 11 (Micro-
      Application Group only), 13 or 14, and BOTH ACC 11 and 12. -
      Probability= 10/10
  and The FAILed course must be PASSed to satisfy the curriculum
      requirements. - Probability=  9/10


NOTE:
  Student must be made aware that Mat 13 or 14 must be completed.
```

RULE NUMBER: 13

IF:
[MATHCHK] = 1
and your status with last KCC math course is FAILED
and The Last Math Course Taken or CUNY result was MATH 13 or HIGHER
and [ADVISE] = 1

THEN:
[ADVISE] IS GIVEN THE VALUE 2
and [MATHCHK] IS GIVEN THE VALUE 2
and DP 11 recommended - Probability= 9/10
and DP Majors are required to take ONE course from Math 11 (Micro-Application Group only), 13 or 14, and BOTH ACC 11 and 12. - Probability= 10/10
and The FAILed course must be PASSed to satisfy the curriculum requirements. - Probability= 9/10

NOTE:
This makes the student aware that if they failed MAT 13 or 14 they will have to repeat the course.

RULE NUMBER: 14

IF:
Major field is COMPUTER SCIENCE
and Status of Computer Science courses is NONE Taken yet
and [CUNYTEST] >= 15
and [CUNYTEST] <= 32

THEN:
[ADVISE] IS GIVEN THE VALUE 2
and Requirements in Computer Science include Calculus and Differential Equations. - Probability= 10/10
and A major in Data Processing should be considered - Probability= 8/10

NOTE:
Computer science students may really belong in Data Processing.

```
RULE NUMBER: 15

IF:
       Major field is COMPUTER SCIENCE
  and  Status of Computer Science courses is NONE Taken yet
  and  [CUNYTEST] < 15


THEN:
       [ADVISE] IS GIVEN THE VALUE 2
  and  A major in Data Processing should be considered - Probability=  7/10
  and  DP 54 - Micro-computer Applications group is recommended. -
       Probability=  9/10
  and  No DP until MAT M1 Passed - Probability= 10/10
  and  Requirements in Computer Science include Calculus and Differential
       Equations. - Probability= 10/10

NOTE:
  CUNY math exam requiring M1 ; Computer Science is not recommended.

----------------------------------------



RULE NUMBER: 16

IF:
       Major field is COMPUTER SCIENCE
  and  Status of Computer Science courses is NONE Taken yet
  and  [CUNYTEST] >32


THEN:
       Requirements in Computer Science include Calculus and Differential
       Equations. - Probability= 10/10
  and  CS 12  may only be taken with MAT 15 - Probability= 10/10
  and  [ADVISE] IS GIVEN THE VALUE 2


NOTE:
  this is for a possible CS major.
```

RULE NUMBER: 17

IF:
Major field is COMPUTER SCIENCE
and Status of Computer Science courses is CS 12 was INComplete or Withdrawal or FAILed

THEN:
[ADVISE] IS GIVEN THE VALUE 2
and CS 12 must be PASSed before going on in Computer Science. - Probability= 10/10
and Requirements in Computer Science include Calculus and Differential Equations. - Probability= 10/10
and A major in Data Processing should be considered - Probability= 9/10

NOTE:
A failure in CS 12 does not imply continuing with a CS curriculum.

RULE NUMBER: 18

IF:
Status of Computer Science courses is CS 12 PASSed with C or better
and Major field is COMPUTER SCIENCE
and MAT 15 grade was PASSing

THEN:
[ADVISE] IS GIVEN THE VALUE 2
and CS 13 has MAT 16 as a corequisite - Probability= 10/10
and Requirements in Computer Science include Calculus and Differential Equations. - Probability= 10/10

REFERENCE:
CS curriculum begins in the fall with CS 12 and continues into the spring semester with CS 13.

RULE NUMBER: 19

IF:
Major field is COMPUTER SCIENCE
and Status of Computer Science courses is CS 12 Grade of 'D'
and MAT 15 grade was PASSing

THEN:
[ADVISE] IS GIVEN THE VALUE 2
and Requirements in Computer Science include Calculus and Differential Equations. - Probability= 10/10
and CS 13 has MAT 16 as a corequisite - Probability= 7/10
and A major in Data Processing should be considered - Probability= 8/10

REFERENCE:
Student should be told of alternate areas of study relating to computers (i.e. Data Processing).

RULE NUMBER: 20

IF:
Major field is COMPUTER SCIENCE
and Status of Computer Science courses is CS 12 PASSed with C or better or CS 12 Grade of 'D'
and MAT 15 grade was FAILing or W or INC

THEN:
[ADVISE] IS GIVEN THE VALUE 2
and Requirements in Computer Science include Calculus and Differential Equations. - Probability= 10/10
and CS 13 has MAT 16 as a corequisite - Probability= 10/10
and A major in Data Processing should be considered - Probability= 9/10

NOTE:
Cannot continue since MAT 16 is Corequisite for CS 13.

```
RULE NUMBER: 21

IF:
       Major field is SOME OTHER MAJOR


THEN:
       [ADVISE] IS GIVEN THE VALUE 2
  and  assistance for your curriculum is not available at this time; please
       speak to an advisor. - Probability= 10/10


NOTE:
  not ready yet

----------------------------------------



RULE NUMBER: 22

IF:
       [NEXTDP] = 1
  and  Last DP course taken was DP 31
  and  Last grade in a DP course  was INComplete or Withdrawal


THEN:
       [ADVISE] IS GIVEN THE VALUE 2
  and  All INC grades must be removed this semester; DP registration by
       departmental approval only. Mat 13 or 14 is required of majors. -
       Probability= 10/10
  and  DP (Computer Programming) Major requirements include: DP 31 and 32, and
       one from DP 54, 61, 63, and one of DP 13, 41 or CP 35(APL), 53(PL/I),
       54(Pascal) - Probability= 10/10

----------------------------------------



RULE NUMBER: 23

IF:
       [NEXTDP] = 1
  and  Last DP course taken was DP 31
  and  Last grade in a DP course  was Failed


THEN:
       [ADVISE] IS GIVEN THE VALUE 2
  and  The FAILed course must be PASSed to satisfy the curriculum
       requirements. - Probability= 10/10
  and  DP 54 - Micro-computer Applications group is recommended. -
       Probability=  9/10
  and  DP 31 recommended - Probability=  7/10
  and  DP (Computer Programming) Major requirements include: DP 31 and 32, and
       one from DP 54, 61, 63, and one of DP 13, 41 or CP 35(APL), 53(PL/I),
       54(Pascal) - Probability= 10/10
```

```
RULE NUMBER: 24

IF:
       [NEXTDP] = 1
  and  Last DP course taken was DP 13, or numbered above 31
  and  Last Accounting course PASSed was NONE


THEN:
       ACC 11 followed by ACC 12. - Probability=  9/10
  and  [ADVISE] IS GIVEN THE VALUE 2
  and  DP (Computer Programming) Major requirements include: DP 31 and 32, and
       one from DP 54, 61, 63, and one of DP 13, 41 or CP 35(APL), 53(PL/I),
       54(Pascal) - Probability= 10/10
  and  DP Majors are required to take ONE course from Math 11 (Micro-
       Application Group only), 13 or 14, and BOTH ACC 11 and 12. -
       Probability= 10/10

----------------------------------------


RULE NUMBER: 25

IF:
       [NEXTDP] = 1
  and  Last DP course taken was DP 13, or numbered above 31
  and  Last Accounting course PASSed was ACC 11


THEN:
       ACC 12 - Probability=  9/10
  and  DP (Computer Programming) Major requirements include: DP 31 and 32, and
       one from DP 54, 61, 63, and one of DP 13, 41 or CP 35(APL), 53(PL/I),
       54(Pascal) - Probability= 10/10
  and  DP Majors are required to take ONE course from Math 11 (Micro-
       Application Group only), 13 or 14, and BOTH ACC 11 and 12. -
       Probability= 10/10
  and  [ADVISE] IS GIVEN THE VALUE 2
```

RULE NUMBER: 26

IF:
[NEXTDP] = 1
and Last DP course taken was DP 13, or numbered above 31
and Last Accounting course PASSed was ACC 12

THEN:
DP (Computer Programming) Major requirements include: DP 31 and 32, and one from DP 54, 61, 63, and one of DP 13, 41 or CP 35(APL), 53(PL/I), 54(Pascal) - Probability= 10/10
and DP (Computer Programming) Majors are required to take Math 13 or 14. - Probability= 10/10
and [ADVISE] IS GIVEN THE VALUE 2

RULE NUMBER: 27

IF:
[NEXTDP] = 1
and Last DP course taken was DP 12
and Last grade in a DP course was PASSED with 'C' or Higher
and Last Accounting course PASSed was NONE

THEN:
[ADVISE] IS GIVEN THE VALUE 2
and DP 31 recommended - Probability= 10/10
and ACC 11 followed by ACC 12. - Probability= 9/10
and DP (Computer Programming) Major requirements include: DP 31 and 32, and one from DP 54, 61, 63, and one of DP 13, 41 or CP 35(APL), 53(PL/I), 54(Pascal) - Probability= 10/10
and DP (Computer Programming) Majors are required to take Math 13 or 14. - Probability= 10/10

REFERENCE:
Student has completed DP 12 with C or better and is being informed of additional req's.

```
RULE NUMBER: 28

IF:
       [NEXTDP] = 1
  and  Last DP course taken was DP 12
  and  Last grade in a DP course  was PASSED with 'C' or Higher
  and  Last Accounting course PASSed was ACC 11


THEN:
       [ADVISE] IS GIVEN THE VALUE 2
  and  DP 31 recommended - Probability= 10/10
  and  DP (Computer Programming) Major requirements include: DP 31 and 32, and
       one from DP 54, 61, 63, and one of DP 13, 41 or CP 35(APL), 53(PL/I),
       54(Pascal) - Probability= 10/10
  and  ACC 12 - Probability= 10/10
  and  DP (Computer Programming) Majors are required to take Math 13 or 14. -
       Probability= 10/10

----------------------------------------


RULE NUMBER: 29

IF:
       [NEXTDP] = 1
  and  Last DP course taken was DP 12
  and  Last grade in a DP course  was PASSED with 'C' or Higher
  and  Last Accounting course PASSed was ACC 12


THEN:
       [ADVISE] IS GIVEN THE VALUE 2
  and  DP 31 recommended - Probability= 10/10
  and  DP (Computer Programming) Major requirements include: DP 31 and 32, and
       one from DP 54, 61, 63, and one of DP 13, 41 or CP 35(APL), 53(PL/I),
       54(Pascal) - Probability= 10/10
  and  DP (Computer Programming) Majors are required to take Math 13 or 14. -
       Probability= 10/10
```

```
RULE NUMBER: 30

IF:
        [NEXTDP] = 1
  and   Last DP course taken was DP 31
  and   Last grade in a DP course  was PASSED with 'C' or Higher
  and   Last Accounting course PASSed was NONE


THEN:
        [ADVISE] IS GIVEN THE VALUE 2
  and   DP 32 recommended - Probability= 10/10
  and   ACC 11 followed by ACC 12. - Probability= 10/10
  and   ONE from DP 54, 61, or 63. - Probability=  9/10
  and   ONE from DP 13, 41 or CP 35, 53, or 54. - Probability=  8/10
  and   DP (Computer Programming) Majors are required to take Math 13 or 14. -
        Probability= 10/10

----------------------------------------


RULE NUMBER: 31

IF:
        [NEXTDP] = 1
  and   Last DP course taken was DP 31
  and   Last grade in a DP course  was PASSED with 'C' or Higher
  and   Last Accounting course PASSed was ACC 11


THEN:
        [ADVISE] IS GIVEN THE VALUE 2
  and   DP 32 recommended - Probability= 10/10
  and   ACC 12 - Probability= 10/10
  and   ONE from DP 54, 61, or 63. - Probability=  9/10
  and   ONE from DP 13, 41 or CP 35, 53, or 54. - Probability=  8/10
  and   DP (Computer Programming) Majors are required to take Math 13 or 14. -
        Probability= 10/10
```

```
RULE NUMBER: 32

IF:
        [NEXTDP] = 1
  and   Last DP course taken was DP 31
  and   Last grade in a DP course  was PASSED with 'C' or Higher
  and   Last Accounting course PASSed was ACC 12


THEN:
        [ADVISE] IS GIVEN THE VALUE 2
  and   DP 32 recommended - Probability= 10/10
  and   ONE from DP 54, 61, or 63. - Probability=  9/10
  and   ONE from DP 13, 41 or CP 35, 53, or 54. - Probability=  8/10
  and   DP (Computer Programming) Majors are required to take Math 13 or 14. -
        Probability= 10/10

----------------------------------------



RULE NUMBER: 33

IF:
        [NEXTDP] = 1
  and   Last DP course taken was DP 31
  and   Last grade in a DP course  was Passed with 'D'
  and   Last Accounting course PASSed was NONE


THEN:
        [ADVISE] IS GIVEN THE VALUE 2
  and   DP 32 recommended - Probability= 10/10
  and   DP 41 recommended - Probability=  9/10
  and   DP 54 (Microcomputer Applications) - Probability=  9/10
  and   DP (Computer Programming) Major requirements include: DP 31 and 32, and
        one from DP 54, 61, 63, and one of DP 13, 41 or CP 35(APL), 53(PL/I),
        54(Pascal) - Probability= 10/10
  and   ACC 11 followed by ACC 12. - Probability=  8/10
  and   DP (Computer Programming) Majors are required to take Math 13 or 14. -
        Probability= 10/10

NOTE:
  Suggest DP 41(RPG) for requirement after completing DP 32.
```

```
RULE NUMBER: 34

IF:
       [NEXTDP] = 1
  and  Last DP course taken was DP 31
  and  Last grade in a DP course  was Passed with 'D'
  and  Last Accounting course PASSed was ACC 11

THEN:
       [ADVISE] IS GIVEN THE VALUE 2
  and  DP 32 recommended - Probability= 10/10
  and  ACC 12 - Probability= 10/10
  and  DP 41 recommended - Probability=  9/10
  and  DP 54 (Microcomputer Applications) - Probability=  9/10
  and  DP (Computer Programming) Major requirements include: DP 31 and 32, and
       one from DP 54, 61, 63, and one of DP 13, 41 or CP 35(APL), 53(PL/I),
       54(Pascal) - Probability= 10/10
  and  DP (Computer Programming) Majors are required to take Math 13 or 14. -
       Probability= 10/10

----------------------------------------


RULE NUMBER: 35

IF:
       [NEXTDP] = 1
  and  Last DP course taken was DP 31
  and  Last grade in a DP course  was Passed with 'D'
  and  Last Accounting course PASSed was ACC 12

THEN:
       [ADVISE] IS GIVEN THE VALUE 2
  and  DP 32 recommended - Probability= 10/10
  and  DP 41 recommended - Probability=  9/10
  and  DP 54 (Microcomputer Applications) - Probability=  9/10
  and  DP (Computer Programming) Major requirements include: DP 31 and 32, and
       one from DP 54, 61, 63, and one of DP 13, 41 or CP 35(APL), 53(PL/I),
       54(Pascal) - Probability= 10/10
  and  DP (Computer Programming) Majors are required to take Math 13 or 14. -
       Probability= 10/10
```

```
RULE NUMBER: 36

IF:
        [NEXTDP] = 1
  and   Last DP course taken was DP 12
  and   Last grade in a DP course  was Passed with 'D'
  and   Last Accounting course PASSed was NONE

THEN:
        [ADVISE] IS GIVEN THE VALUE 2
  and   DP 54 - Micro-computer Applications group is recommended. -
        Probability=  8/10
  and   ACC 11 followed by ACC 12. - Probability=  8/10
  and   DP 31 recommended - Probability=  7/10
  and   DP 41 recommended - Probability=  7/10
  and   DP Majors are required to take ONE course from Math 11 (Micro-
        Application Group only), 13 or 14, and BOTH ACC 11 and 12. -
        Probability= 10/10

NOTE:
  Switch to micro-application group(DP 54) or continue with DP
  curriculum, COBOL and or RPG.

----------------------------------------


RULE NUMBER: 37

IF:
        [NEXTDP] = 1
  and   Last DP course taken was DP 12
  and   Last grade in a DP course  was Passed with 'D'
  and   Last Accounting course PASSed was ACC 11

THEN:
        [ADVISE] IS GIVEN THE VALUE 2
  and   DP 54 - Micro-computer Applications group is recommended. -
        Probability=  8/10
  and   ACC 12 - Probability= 10/10
  and   DP 31 recommended - Probability=  7/10
  and   DP 41 recommended - Probability=  7/10
  and   DP Majors are required to take ONE course from Math 11 (Micro-
        Application Group only), 13 or 14, and BOTH ACC 11 and 12. -
        Probability= 10/10
```

```
RULE NUMBER: 38

IF:
        [NEXTDP] = 1
  and   Last DP course taken was DP 12
  and   Last grade in a DP course  was Passed with 'D'
  and   Last Accounting course PASSed was ACC 12


THEN:
        [ADVISE] IS GIVEN THE VALUE 2
  and   DP 54 - Micro-computer Applications group is recommended. -
        Probability=  8/10
  and   DP 31 recommended - Probability=  7/10
  and   DP 41 recommended - Probability=  7/10
  and   DP Majors are required to take ONE course from Math 11 (Micro-
        Application Group only), 13 or 14, and BOTH ACC 11 and 12. -
        Probability= 10/10

----------------------------------------



RULE NUMBER: 39

IF:
        [MATHCHK] = 1
  and   your status with last KCC math course is HAVEN'T TAKEN ANY
  and   [ADVISE] = 1
  and   [CUNYTEST] > 24


THEN:
        [MATHCHK] IS GIVEN THE VALUE 2
  and   [ADVISE] IS GIVEN THE VALUE 2
  and   DP 11 recommended - Probability= 10/10
  and   DP Majors are required to take ONE course from Math 11 (Micro-
        Application Group only), 13 or 14, and BOTH ACC 11 and 12. -
        Probability= 10/10
  and   Mathematics evaluation is IN PROGRESS


NOTE:
  Student has passed CUNY Math test and may enroll in DP 11, and begin
  curriculum.
```

```
RULE NUMBER: 40

IF:
        [MATHCHK] = 1
  and   your status with last KCC math course is HAVEN'T TAKEN ANY
  and   [ADVISE] = 1
  and   [CUNYTEST] >= 20
  and   [CUNYTEST] < 25


THEN:
        [MATHCHK] IS GIVEN THE VALUE 2
  and   [ADVISE] IS GIVEN THE VALUE 2
  and   DP 11 recommended - Probability= 10/10
  and   DP Majors are required to take ONE course from Math 11 (Micro-
        Application Group only), 13 or 14, and BOTH ACC 11 and 12. -
        Probability= 10/10
  and   DP has 2 group areas: Computer Programming or Microcomputer
        Applications - Probability= 10/10
  and   Sign up in Math Lab to get assistance for CUNY test - Probability=
        10/10
  and   Mathematics evaluation is DONE


NOTE:
  Student with a CUNY test score between 15 and 24 is being told of the
  two DP concentrations.

----------------------------------------



RULE NUMBER: 41

IF:
        [MATHCHK] = 1
  and   your status with last KCC math course is HAVEN'T TAKEN ANY
  and   [ADVISE] = 1
  and   [CUNYTEST] >= 15
  and   [CUNYTEST] <= 19


THEN:
        [MATHCHK] IS GIVEN THE VALUE 2
  and   [ADVISE] IS GIVEN THE VALUE 2
  and   DP 11 recommended - Probability= 10/10
  and   DP Majors are required to take ONE course from Math 11 (Micro-
        Application Group only), 13 or 14, and BOTH ACC 11 and 12. -
        Probability= 10/10
  and   DP has 2 group areas: Computer Programming or Microcomputer
        Applications - Probability= 10/10
  and   MAT R2 may be taken - Probability= 10/10
  and   Mathematics evaluation is DONE


NOTE:
  CUNY score between 15 and 19 implies Mat R2 and DP 11
```

```
RULE NUMBER: 42

IF:
       [ADVISE] = 2
  and  Number of courses PASSed in Foreign Language, Literature or Philosophy
       is ONE course.


THEN:
       [GROUPREQ] IS GIVEN THE VALUE [GROUPREQ] - 1
  and  [GRP2] IS GIVEN THE VALUE 1
  and  [GRPCT] IS GIVEN THE VALUE [GRPCT] + 1

----------------------------------------



RULE NUMBER: 43

IF:
       [ADVISE] = 2
  and  Number of courses PASSed in Foreign Language, Literature or Philosophy
       is TWO or MORE courses


THEN:
       [GROUPREQ] IS GIVEN THE VALUE [GROUPREQ] + 2
  and  [GRP2] IS GIVEN THE VALUE 2
  and  [GRPCT] IS GIVEN THE VALUE [GRPCT] + 1

----------------------------------------



RULE NUMBER: 44

IF:
       [ADVISE] = 2
  and  Number of courses PASSed in Anthropology, Psychology, or Sociology is
       ONE course


THEN:
       [GROUPREQ] IS GIVEN THE VALUE [GROUPREQ] - 1
  and  [GRP4] IS GIVEN THE VALUE 1
  and  [GRPCT] IS GIVEN THE VALUE [GRPCT] - 1
```

```
RULE NUMBER: 45

IF:
       [ADVISE] = 2
  and  Number of courses PASSed in Anthropology, Psychology, or Sociology is
       TWO or MORE courses


THEN:
       [GROUPREQ] IS GIVEN THE VALUE [GROUPREQ] + 2
  and  [GRP4] IS GIVEN THE VALUE 2
  and  [GRPCT] IS GIVEN THE VALUE [GRPCT] + 1

----------------------------------------



RULE NUMBER: 46

IF:
       [ADVISE] = 2
  and  Number of courses PASSed in History, Economics, or Political Science is
       ONE course


THEN:
       [GROUPREQ] IS GIVEN THE VALUE [GROUPREQ] - 1
  and  [GRP3] IS GIVEN THE VALUE 1
  and  [GRPCT] IS GIVEN THE VALUE [GRPCT] + 1

----------------------------------------



RULE NUMBER: 47

IF:
       [ADVISE] = 2
  and  Number of courses PASSed in History, Economics, or Political Science is
       TWO or MORE courses


THEN:
       [GROUPREQ] IS GIVEN THE VALUE [GROUPREQ] + 2
  and  [GRP3] IS GIVEN THE VALUE 2
  and  [GRPCT] IS GIVEN THE VALUE [GRPCT] - 1
```

```
RULE NUMBER: 48

IF:
       [ADVISE] = 2
  and  Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE
       ARTS is ONE course

THEN:
       [GROUPREQ] IS GIVEN THE VALUE [GROUPREQ] + 1
  and  [GRP1] IS GIVEN THE VALUE 1
  and  [GRPCT] IS GIVEN THE VALUE [GRPCT] + 1

----------------------------------------



RULE NUMBER: 49

IF:
       [ADVISE] = 2
  and  Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE
       ARTS is TWO or MORE courses

THEN:
       [GROUPREQ] IS GIVEN THE VALUE [GROUPREQ] + 2
  and  [GRP1] IS GIVEN THE VALUE 2
  and  [GRPCT] IS GIVEN THE VALUE [GRPCT] + 1

----------------------------------------



RULE NUMBER: 50

IF:
       [GROUPREQ] > 4
  and  Major field is NOT COMPUTER SCIENCE

THEN:
       Group requirements (I - IV) have been met. Additional electives may be
       taken. - Probability= 10/10
```

RULE NUMBER: 51

IF:
[GROUPREQ] = 4
and Major field is NOT COMPUTER SCIENCE
and [GRPCT] = 4

THEN:
Student may choose a course from one of the 4 areas to satisfy Group Requirements - Probability= 9/10

RULE NUMBER: 52

IF:
[GROUPREQ] = 4
and Major field is COMPUTER SCIENCE
and [GRPCT] = 4

THEN:
Group requirements (I - IV) have been met. Additional electives may be taken. - Probability= 10/10

REFERENCE:
CS majors are required to take at least one course from each group area for A.S. Degree.

RULE NUMBER: 53

IF:
[GROUPREQ] = 4
and Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE ARTS is TWO or MORE courses
and Number of courses PASSed in Foreign Language, Literature or Philosophy is TWO or MORE courses

THEN:
A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability= 10/10
and A Group IV (Anthro.,Psych., Sociology) course may be taken - Probability= 10/10

RULE NUMBER: 54

IF:
[GROUPREQ] = 4
and Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE ARTS is TWO or MORE courses
and Number of courses PASSed in History, Economics, or Political Science is TWO or MORE courses

THEN:
A Group II (For.Lang.,Lit., Philosophy) course may be taken - Probability= 10/10
and A Group IV (Anthro.,Psych., Sociology) course may be taken - Probability= 10/10

RULE NUMBER: 55

IF:
[GROUPREQ] = 4
and Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE ARTS is TWO or MORE courses
and Number of courses PASSed in Anthropology, Psychology, or Sociology is TWO or MORE courses

THEN:
A Group II (For.Lang.,Lit., Philosophy) course may be taken - Probability= 10/10
and A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability= 10 10

RULE NUMBER: 56

IF:
[GROUPREQ] = 4
and Number of courses PASSed in Foreign Language, Literature or Philosophy is TWO or MORE courses
and Number of courses PASSed in History, Economics, or Political Science is TWO or MORE courses

THEN:
A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be taken - Probability= 10/10
and A Group IV (Anthro.,Psych., Sociology) course may be taken - Probability= 10/10

RULE NUMBER: 57

IF:
[GROUPREQ] = 4
and Number of courses PASSed in Foreign Language, Literature or Philosophy is TWO or MORE courses
and Number of courses PASSed in Anthropology, Psychology, or Sociology is TWO or MORE courses

THEN:
A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be taken - Probability= 10/10
and A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability= 10/10

RULE NUMBER: 58

IF:
[GROUPREQ] = 4
and Number of courses PASSed in History, Economics, or Political Science is TWO or MORE courses
and Number of courses PASSed in Anthropology, Psychology, or Sociology is TWO or MORE courses

THEN:
A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be taken - Probability= 10/10
and A Group II (For.Lang.,Lit., Philosophy) course may be taken - Probability= 10/10

RULE NUMBER: 59

IF:
[GROUPREQ] = 4
and Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE ARTS is ONE course
and Number of courses PASSed in Anthropology, Psychology, or Sociology is ONE course
and Major field is NOT COMPUTER SCIENCE
and [GRPCT]=3

THEN:
Group requirements (I - IV) have been met. Additional electives may be taken. - Probability= 10/10

RULE NUMBER: 60

IF:
[GROUPREQ] = 4
and Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE ARTS is ONE course
and Number of courses PASSed in History, Economics, or Political Science is ONE course
and Major field is NOT COMPUTER SCIENCE
and [GRPCT]=3

THEN:
Group requirements (I - IV) have been met. Additional electives may be taken. - Probability= 10/10

RULE NUMBER: 61

IF:
[GROUPREQ] = 4
and Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE ARTS is ONE course
and Number of courses PASSed in Foreign Language, Literature or Philosophy is ONE course.
and Major field is NOT COMPUTER SCIENCE
and [GRPCT]=3

THEN:
Group requirements (I - IV) have been met. Additional electives may be taken. - Probability= 10/10

RULE NUMBER: 62

IF:
[GROUPREQ] = 4
and Number of courses PASSed in Anthropology, Psychology, or Sociology is ONE course
and Number of courses PASSed in History, Economics, or Political Science is ONE course
and Major field is NOT COMPUTER SCIENCE
and [GRPCT]=3

THEN:
Group requirements (I - IV) have been met. Additional electives may be taken. - Probability= 10/10

RULE NUMBER: 63

IF:
[GROUPREQ] = 4
and Number of courses PASSed in Anthropology, Psychology, or Sociology is ONE course
and Number of courses PASSed in Foreign Language, Literature or Philosophy is ONE course.
and Major field is NOT COMPUTER SCIENCE
and [GRPCT]=3

THEN:
Group requirements (I - IV) have been met. Additional electives may be taken. - Probability= 10/10

RULE NUMBER: 64

IF:
[GROUPREQ] = 4
and Number of courses PASSed in History, Economics, or Political Science is ONE course
and Number of courses PASSed in Foreign Language, Literature or Philosophy is ONE course.
and Major field is NOT COMPUTER SCIENCE
and [GRPCT]=3

THEN:
Group requirements (I - IV) have been met. Additional electives may be taken. - Probability= 10/10

```
RULE NUMBER: 65

IF:
        Major field is DATA PROCESSING
  and   [GRPCT]=1
  and   [GROUPREQ]=1


THEN:
        A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be
        taken - Probability= 10/10
  and   A Group II (For.Lang.,Lit., Philosophy) course may be taken -
        Probability= 10/10
  and   A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability=
        10/10
  and   A Group IV (Anthro.,Psych., Sociology) course may be taken -
        Probability= 10/10

----------------------------------------



RULE NUMBER: 66

IF:
        [GRPCT] = 1
  and   [GRP1] = 2


THEN:
        A Group II (For.Lang.,Lit., Philosophy) course may be taken -
        Probability= 10/10
  and   A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability=
        10/10
  and   A Group IV (Anthro.,Psych., Sociology) course may be taken -
        Probability= 10/10


NOTE:
  Two courses from one group have been taken, so no others from that area
  will count toward requirement (DP major)
```

RULE NUMBER: 67

IF:
[GRPCT] = 1
and [GRP2] = 2

THEN:
A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be taken - Probability= 10/10
and A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability= 10/10
and A Group IV (Anthro.,Psych., Sociology) course may be taken - Probability= 10/10

NOTE:
Two courses from one group, all other areas ok.

RULE NUMBER: 68

IF:
[GRPCT] = 1
and [GRP3] = 2

THEN:
A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be taken - Probability= 10/10
and A Group II (For.Lang.,Lit., Philosophy) course may be taken - Probability= .0/10
and A Group IV (Anthro.,Psych., Sociology) course may be taken - Probability= 10/10

NOTE:
same as preceding rule

RULE NUMBER: 69

IF:
[GRPCT] = 1
and [GRP4] = 2

THEN:
A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be taken - Probability= 10/10
and A Group II (For.Lang.,Lit., Philosophy) course may be taken - Probability= 10/10
and A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability= 10/10

NOTE:
same as preceding

RULE NUMBER: 70

IF:
[GRPCT] = 2
and [GROUPREQ] = 3
and Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE ARTS is ONE course or TWO or MORE courses
and Number of courses PASSed in Foreign Language, Literature or Philosophy is ONE course, or TWO or MORE courses

THEN:
A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability= 10/10
and A Group IV (Anthro.,Psych., Sociology) course may be taken - Probability= 10/10

NOTE:
one course needed from Group III or IV to fulfill requirement (DP major)

RULE NUMBER: 71

IF:
[GRPCT] = 2
and [GROUPREQ] = 3
and Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE ARTS is ONE course or TWO or MORE courses
and Number of courses PASSed in History, Economics, or Political Science is ONE course or TWO or MORE courses

THEN:
A Group II (For.Lang.,Lit., Philosophy) course may be taken - Probability= 10/10
and A Group IV (Anthro.,Psych., Sociology) course may be taken - Probability= 10/10

NOTE:
At least one course from Group II or IV must be taken.

RULE NUMBER: 72

IF:
[GRPCT] = 2
and [GROUPREQ] = 3
and Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE ARTS is ONE course or TWO or MORE courses
and Number of courses PASSed in Anthropology, Psychology, or Sociology is ONE course or TWO or MORE courses

THEN:
A Group II (For.Lang.,Lit., Philosophy) course may be taken - Probability= 10/10
and A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability= 10/10

NOTE:
At least one course from Group II or III may be taken.

RULE NUMBER: 73

IF:
[GRPCT] = 2
and [GROUPREQ] = 3
and Number of courses PASSed in Foreign Language, Literature or Philosophy is ONE course, or TWO or MORE courses
and Number of courses PASSed in History, Economics, or Political Science is ONE course or TWO or MORE courses

THEN:
A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be taken - Probability= 10/10
and A Group IV (Anthro.,Psych., Sociology) course may be taken - Probability= 10/10

NOTE:
one course needed from Group I or IV to fulfill requirement (DP major)

RULE NUMBER: 74

IF:
[GRPCT] = 2
and [GROUPREQ] = 3
and Number of courses PASSed in Foreign Language, Literature or Philosophy is ONE course, or TWO or MORE courses
and Number of courses PASSed in Anthropology, Psychology, or Sociology is ONE course or TWO or MORE courses

THEN:
A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be taken - Probability= 10/10
and A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability= 10/10

NOTE:
one course needed from Group I or III to fulfill requirement (DP major)

RULE NUMBER: 75

IF:
[GRPCT] = 2
and [GROUPREQ] = 3
and Number of courses PASSed in History, Economics, or Political Science is ONE course or TWO or MORE courses
and Number of courses PASSed in Anthropology, Psychology, or Sociology is ONE course or TWO or MORE courses

THEN:
A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be taken - Probability= 10/10
and A Group II (For.Lang.,Lit., Philosophy) course may be taken - Probability= 10/10

NOTE:
one course needed from Group I or II to fulfill requirement (DP major)

RULE NUMBER: 76

IF:
Has HPE 12 been completed: NO

THEN:
HPE 12 may be taken - Probability= 10/10

NOTE:
HPE 12 is a requirement for all curriculums

RULE NUMBER: 77

IF:
Last English course PASSed was ENG C2 or 12

THEN:
ENG 22 may be taken - Probability= 10/10

RULE NUMBER: 78

IF:
Last English course PASSed was ENG 22

THEN:
ENGLISH requirement completed - Probability= 10/10

RULE NUMBER: 79

IF:
Last English course PASSed was NONE PASSed

THEN:
Repeat FAILed English course - Probability= 10/10

RULE NUMBER: 80

IF:
Last English course PASSed was ENG 03/04/C1 or NONE taken or NO CUNY ENGLISH placement

THEN:
SEE Advisor for correct English placement if any - Probability= 10/10

```
RULE NUMBER: 81

IF:
        Major field is NOT DATA PROCESSING
  and   [GRPCT]=1
  and   [GRP1]=1


THEN:
        A Group II (For.Lang.,Lit., Philosophy) course may be taken -
        Probability= 10/10
  and   A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability=
        10/10
  and   A Group IV (Anthro.,Psych., Sociology) course may be taken -
        Probability= 10/10

----------------------------------------



RULE NUMBER: 82

IF:
        Major field is NOT DATA PROCESSING
  and   [GRPCT]=1
  and   [GRP2] = 1


THEN:
        A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be
        taken - Probability= 10/10
  and   A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability=
        10/10
  and   A Group IV (Anthro.,Psych., Sociology) course may be taken -
        Probability= 10/10

---------- ----------



RULE NUMBER: 83

IF:
        Major field is NOT DATA PROCESSING
  and   [GRPCT]=1
  and   [GRP3] = 1


THEN:
        A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be
        taken - Probability= 10/10
  and   A Group II (For.Lang.,Lit., Philosophy) course may be taken -
        Probability= 10/10
  and   A Group IV (Anthro.,Psych., Sociology) course may be taken -
        Probability= 10/10
```

```
RULE NUMBER: 84

IF:
       Major field is NOT DATA PROCESSING
  and  [GRPCT]=1
  and  [GRP4]=1


THEN:
       A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be
       taken - Probability= 10/10
  and  A Group II (For.Lang.,Lit., Philosophy) course may be taken -
       Probability= 10/10
  and  A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability=
       10/10

----------------------------------------



RULE NUMBER: 85

IF:
       Major field is DATA PROCESSING
  and  [GRPCT] = 2
  and  [GROUPREQ] = 2


THEN:
       A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be
       taken - Probability= 10/10
  and  A Group II (For.Lang.,Lit., Philosophy) course may be taken -
       Probability= 10/10
  and  A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability=
       10/10
  and  A Group IV (Anthro.,Psych., Sociology) course may be taken -
       Probability= 10/10


NOTE:
  Any group area may be used to select a course.
```

RULE NUMBER: 86

IF:
Major field is NOT DATA PROCESSING
and [GRPCT] = 2
and Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE ARTS is NONE as of this semester
and Number of courses PASSed in Anthropology, Psychology, or Sociology is NONE as of this semester

THEN:
A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be taken - Probability= 10/10
and A Group IV (Anthro.,Psych., Sociology) course may be taken - Probability= 10/10

NOTE:
Two groups have been met, two more are needed.

RULE NUMBER: 87

IF:
Major field is NOT DATA PROCESSING
and [GRPCT] = 2
and Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE ARTS is NONE as of this semester
and Number of courses PASSed in History, Economics, or Political Science is NONE as of this semester

THEN:
A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be taken - Probability= 10/10
and A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability= 10/10

NOTE:
Two groups have been met, two more are needed.

RULE NUMBER: 88

IF:
Major field is NOT DATA PROCESSING
and [GRPCT] = 2
and Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE ARTS is NONE as of this semester
and Number of courses PASSed in Foreign Language, Literature or Philosophy is None as of this semester

THEN:
A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be taken - Probability= 10/10
and A Group II (For.Lang.,Lit., Philosophy) course may be taken - Probability= 10/10

NOTE:
Two groups have been met, two more are needed.

RULE NUMBER: 89

IF:
Major field is NOT DATA PROCESSING
and [GRPCT] = 2
and Number of courses PASSed in Anthropology, Psychology, or Sociology is NONE as of this semester
and Number of courses PASSed in History, Economics, or Political Science is NONE as of this semester

THEN:
A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability= 10/10
and A Group IV (Anthro.,Psych., Sociology) course may be taken - Probability= 10/10

NOTE:
Two groups have been met, two more are needed.

RULE NUMBER: 90

IF:
Major field is NOT DATA PROCESSING
and [GRPCT] = 2
and Number of courses PASSed in Anthropology, Psychology, or Sociology is NONE as of this semester
and Number of courses PASSed in Foreign Language, Literature or Philosophy is None as of this semester

THEN:
A Group II (For.Lang.,Lit., Philosophy) course may be taken - Probability= 10/10
and A Group IV (Anthro.,Psych., Sociology) course may be taken - Probability= 10 10

NOTE:
Two groups have been met, two more are needed.

RULE NUMBER: 91

IF:
Major field is NOT DATA PROCESSING
and [GRPCT] = 2
and Number of courses PASSed in History, Economics, or Political Science is NONE as of this semester
and Number of courses PASSed in Foreign Language, Literature or Philosophy is None as of this semester

THEN:
A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability= 10/10
and A Group II (For.Lang.,Lit., Philosophy) course may be taken - Probability= 10/10

NOTE:
Two groups have been met, two more are needed.

RULE NUMBER: 92

IF:
[GRPCT] = 0
and Number of courses PASSed in History, Economics, or Political Science is NONE as of this semester
and Number of courses PASSed in Foreign Language, Literature or Philosophy is None as of this semester
and Number of courses PASSed in (non-studio) ART, MUSIC, SPEECH, or THEATRE ARTS is NONE as of this semester
and Number of courses PASSed in Anthropology, Psychology, or Sociology is NONE as of this semester

THEN:
A Group I (non-studio Art, Music, Speech, Theatre Arts) course may be taken - Probability= 10/10
and A Group II (For.Lang.,Lit., Philosophy) course may be taken - Probability= 10/10
and A Group III (Hist.,Econ.,Pol.Sci) course may be taken - Probability= 10/10
and A Group IV (Anthro.,Psych., Sociology) course may be taken - Probability= 10/10

NOTE:
No group courses have been taken yet

RULE NUMBER: 93

IF:
your status with last KCC math course is PASSED
and Follow up area is DP
and The Last Math Course Taken or CUNY result was MATH 13 or HIGHER

THEN:
Mathematics evaluation is DONE
and Math requirement is fulfilled - Probability= 10/10

```
RULE NUMBER: 94

IF:
       your status with last KCC math course is PASSED
  and  Follow up area is DP
  and  The Last Math Course Taken or CUNY result was MATH M2 or MATH R2

THEN:
       Mathematics evaluation is DONE
  and  MATH 03 to be taken - Probability= 10/10

----------------------------------------


RULE NUMBER: 95

IF:
       your status with last KCC math course is PASSED
  and  Follow up area is DP
  and  The Last Math Course Taken or CUNY result was MATH 03

THEN:
       Mathematics evaluation is DONE
  and  DP Majors are required to take ONE course from Math 11 (Micro-
       Application Group only), 13 or 14, and BOTH ACC 11 and 12. -
       Probability= 10/10

----------------------------------------


RULE NUMBER: 96

IF:
       Follow up area is DP
  and  Last H.S. Course passed was 9th year or elementary algebra or 10th year
       or Geometry
  and  [CUNYTEST] >= 25

THEN:
       Mathematics evaluation is DONE
  and  MATH 03 to be taken - Probability=  9/10
```

RULE NUMBER: 97

IF:
Follow up area is DP
and Last H.S. Course passed was 11th year algebra only or 11th year w/ trigonometry or 12th year
and [CUNYTEST] >= 25

THEN:
Mathematics evaluation is DONE
and DP Majors are required to take *ONE* course from Math 11 (Micro-Application Group only), 13 or 14, and BOTH ACC 11 and 12. - Probability= 10/10
and DP (Computer Programming) Major requirements include: DP 31 and 32, and one from DP 54, 61, 63, and one of DP 13, 41 or CP 35(APL), 53(PL/I), 54(Pascal) - Probability= 10/10

RULE NUMBER: 98

IF:
Last DP course taken was NOT NONE at this time
and Electives taken are DP 54 or DP 61 or DP 63
and Electives taken are NOT CP 35 or CP 53 or CP 54 or DP 13

THEN:
ONE from DP 13, 41 or CP 35, 53, or 54. - Probability= 10/10

RULE NUMBER: 99

IF:
Last DP course taken was NOT NONE at this time
and Electives taken are NOT DP 54 or DP 61 or DP 63
and Electives taken are CP 35 or CP 53 or CP 54 or DP 13

THEN:
ONE from DP 54, 61, or 63. - Probability= 10/10

RULE NUMBER: 100

IF:
Last DP course taken was NOT NONE at this time
and Electives taken are NONE OF THESE

THEN:
DP (Computer Programming) Major requirements include: DP 31 and 32, and one from DP 54, 61, 63, and one of DP 13, 41 or CP 35(APL), 53(PL/I), 54(Pascal) - Probability= 10/10

APPENDIX C

Knowledge Data Base for Human and Automated Expert Systems

A.A.S. DATA PROCESSING** (Computer Programming)

Total credits: 64

Requirements for Matriculants

COLLEGE REQUIREMENTS*

(*Successful completion of the CUNY Freshman Skills Assessment Tests with passing scores on the Reading, Writing, and Mathematics Placement Examinations, or remedial courses may be required in these subjects.)

ENG 03 &/or 04 (if required)	0 credits
ENG C1 (if required)	0
ENG C2 or ENG 12 (based on CUNY Test scores, students may be exempt)	4
ENG 22	4
HPE 12	3
Corrective Speech (if required)	0
MAT M1 & M2 or, MAT R2 proficiency	0

DEPARTMENT REQUIREMENTS‡

Introduction to Data Processing (DP 11 pg. 212)	4 credits
Assembler Language Programming (DP 12)	5
Advanced Assembler Programming (DP 13) **OR** any other Data Processing Course numbered above 32	3-4
Cobol Programming (DP 31)	5
Advanced Programming (DP 32)	4
Survey of Mathematics (MAT 13 pg. 216) OR Analytic Geometry and Pre-Calculus Math (MAT 14)	4
Fundamentals of Accounting I and II (ACC 11-12 pg. 155)	8

GROUP REQUIREMENTS

Course selections, from groups I through IV, must be from at least three different groups
All credits required from groups I, II, III & IV must be in Basic Courses

I. Performing and Visual Arts
(Excluded from this group are Art and Music Studio courses, Theatre Production and Technique courses)
- Art
- Music
- Speech
- Theatre Arts

II. Language and Literature
- Foreign Language
- Literature
- Philosophy

III. Social Sciences
- Economics
- History
- Political Science

IV. Behavioral Sciences
- Anthropology
- Psychology
- Sociology

12 credits

V. Mathematics and Science
- Biological Science
- Mathematics and Computer Science
- Physical Science

4 credits

ELECTIVES—Sufficient to meet required total of **64 credits**

Departmental requirements may be used to satisfy groups I-V requirements where applicable.
‡Consultation with the Department Advisor is required.
**This program is within the Mathematics and Computer Science Department.

MICROCOMPUTER APPLICATIONS CONCENTRATION
(Office Automation Technology)

Introduction to Data Processing (DP 11 pg. 237)	4	credits
Introduction to BASIC (CP 30 pg. 233)	4	
Microcomputer Applications I, II, III, (DP 54, 55, 56)	9	
Choice: ONE of following Programming courses:	3-5	
Introduction to COBOL Programming (CP 52)		
Assembler Language Programming (DP 12)		
Systems Analysis (DP 61)		
Choice: ONE of following Mathematics courses:	4	
Finite Mathematics (MAT 11 pg. 242)		
Survey of Mathematics (MAT 13)		
Analytic Geometry and Pre-Calculus Math (MAT 14)		
Fundamentals of Accounting I and II (ACC 11, 12 pg. 163)	8	

COMPUTER PROGRAMMING CONCENTRATION

Introduction to Data Processing (DP 11 pg. 237)	4
(NOTE: C grade or better in DP 11 required to continue in this concentration.)	
Assembler Language Programming (DP 12)	5
Cobol Programming (DP 31)	5
Advanced Programming (DP 32)	4
Choice: ONE of following Data Processing courses:	3-4
Microcomputer Applications I (DP 54)	
Systems Analysis (DP 61)	
Database Processing (DP 63)	
Choice: ONE of following Programming courses:	3-4
Introduction to APL (CP 35 pg. 234)	
PL/1 Programming (CP 53)	
Introduction to PASCAL (CP 54)	
Advanced Assembler Language Programming (DP 13)	
RPG Programming (DP 41)	
Choice: ONE of following Mathematics courses:	4
Survey of Mathematics (MAT 13 pg. 243) **OR**	
Analytic Geometry and Pre-Calculus Math (Mat 14)	
Fundamentals of Accounting I and II (ACC 11, 12 pg. 163)	8

DATA PROCESSING
(COMPUTER PROGRAMMING)

DATA PROCESSING courses *do not* satsify Group V requirements.

+ DP 11 INTRODUCTION to DATA PROCESSING

4 crs. 4 hrs.

Detailed introduction to the concepts, structure and operation of electronic data processing systems including such topics as components of a computer system; input/output devices; principles of arithmetical operation, decimal, binary and hexadecimal; coding representations of information in computers; flowcharting and coding of problems; machine language and assembler language programming; a brief introduction to procedure-oriented languages such as FORTRAN and COBOL.
An introduction to IBM System 370 assembler language programming.
Prerequisite: MAT 01, 05 or Elementary Algebra.

DP 12 ASSEMBLER LANGUAGE PROGRAMMING

5 crs. 5 hrs.

Detailed study of assembler language programming. Typical business applications will be considered. Programs written by the students will be run on the remote job entry communication network connected to the University Computer Center.
Prerequisite: DP 11

DP 13 ADVANCED ASSEMBLER LANGUAGE PROGRAMMING: IBM SYSTEM/360

4 crs. 4 hrs.

Detailed study of advanced assembler language programming techniques and sophisticated applications. Programs written by the students will be run on the remote job entry communication network connected to the University Computer Center.
Prerequisite: DP 12

DP 21 INTRODUCTION to COMPUTERS and INFORMATION PROCESSING

3 crs. 3 hrs.

This course presents an overview of Computer concepts as an area of general knowledge for the informed individual. Students will be introduced to computer systems touching on such topics as: uses of computers in society; components of a computer system; input/output devices; external and internal data representation; storage and retrieval of information; flowcharting and programming in BASIC; other programming languages. A very brief introduction to other selected topics such as: word processing, systems analysis and design, operating systems, will be included.
COURSE NOT OPEN TO DATA PROCESSING MAJORS OR STUDENTS WHOSE CURRICULUM REQUIRE BA 60.
Prerequisite: Passing grade on CUNY mathematics exam.

DP 22 COMPUTER LITERACY: INTRODUCTION to COMPUTERS and COMPUTER APPLICATIONS

3 crs. 3 hrs.

A computer literacy course to introduce various computer hardware and software. Students will concentrate on basic applications with emphasis on the computer as a tool at home, at school, and at work. Topics to be included: Relational Database-Management (d BASE II or CONDOR), Word Processing, Spread Sheets, Graphics, Games and an introduction to BASIC.
COURSE NOT OPEN TO DATA PROCESSING MAJORS OR STUDENTS WHOSE CURRICULUM REQUIRE BA 60.
Prerequisite: Passing grade on CUNY mathematics exam.

+Basic Course

DP 31 COBOL PROGRAMMING

5 crs. 5 hrs.

Detailed study of COBOL programming. Typical business applications will be considered. Programs written by the students will be run on the remote job entry communication network connected to the University Computer Center.
Prerequisite: DP 12

DP 32 ADVANCED PROGRAMMING

4 crs. 4 hrs.

Magnetic tape and disk programming in COBOL
A thorough introduction to operating systems including such topics as: system control and system service programs such as the Supervisor, Job Control, and the Linkage Editor; interaction of the control program and the problem program within the systems environment; Job Control Language; program libraries; channels and interrupts; data management and IOCS; Sequential Access Method; Indexed Sequential Access Method; Direct Access Method; utility programs; multiprogramming considerations; introduction to virtual storage.
Prerequisites: DP 12 and 31

DP 41 RPG PROGRAMMING

3 crs. 3 hrs.

A detailed study of RPG (Report Program Generator) programming. This language enables students to write programs which will produce a wide variety of business reports. Programs written by the students will be run on the remote job entry communication network connected to the University Computer Center.
Prerequisite: DP 12

DP 52 MICROCOMPUTER APPLICATIONS

4 crs. 4 hrs.

An introduction to personal computer hardware and software. Programming will be done in BASIC. Topics covered will include graphics, microcomputer operating systems, electronic spreadsheets, wordprocessing systems, and personal computer selection.
This course is intended primarily for Data Processing and Computer Science majors.
Prerequisite: DP 12 or CS 14 or permission of the Department.

DP 61 SYSTEMS ANALYSIS

3 crs. 3 hrs.

Course introduces the tools and methods used by management to develop systems for computer applications. Topics covered are: Systems investigation, input design, output design, file design, documentation, systems testing, systems implementation, hardware and software.
Prerequisite: DP 12

DP 63 DATABASE PROCESSING

4 crs. 4 hrs.

This course includes: Concepts and structures necessary to design and implement a databased management system (DBMS); Physical file organization and data organization techniques; Network, hierarchical and relational models applied to DBMS; the CODASYL DBTG model; Commercial DBMS systems. Students will use a database management system on the CUNY computer system network.
Prerequisite: DP 12, or, permission of the Department.

DP 81 INDEPENDENT STUDY

1-3 crs. 1-3 hrs.

Independent study of Data Processing is developed individually between student and faculty member and must be approved by the Department.

DP 82

1-3 crs. 1-3 hrs.

This course is of a topical and pilot nature and is designed to meet the immediate needs and interests of various student populations. It is offered for a maximum of two semesters.

A.A.S. Degree

DATA PROCESSING

COMPUTER PROGRAMMING

COMPUTER PROGRAMMING courses are offered as electives to all students

+ CP 29 INTRODUCTION to COMPUTER PROGRAMMING for SECRETARIAL/OFFICE ADMINISTRATION STUDENTS

4 crs. 4 hrs.

An introduction to programming in the BASIC language with more emphasis placed on string and character manipulation and less on mathematical programming. Students receive hands-on experience using one or more of the following microcomputers: the Apple, the Pet, the North Star, or the TRS 80 Radio Shack Computer. Disk operations and proper use of a printer for producing hard copy will also be emphasized.
Open to Program Majors only.
Prerequisite: Elementary Algebra or MAT R2, or score of 25 or higher on Math Placement Exam.

+ CP 30 INTRODUCTION to BASIC

4 crs. 4 hrs.

BASIC is a general-purpose computer language used in both conversational mode and time-sharing computer centers. It is a simple and natural language requiring a minimum of programming skills, yet bringing about an appreciation of the power of a computer. In this "hands-on" course, with direct interaction between students and the computer, problems in mathematics, the sciences and the social sciences are explored to awaken and augment student interest in the problem areas of today's civilization.
Prerequisite: MAT 03 or Intermediate Algebra.

+ CP 35 INTRODUCTION to APL

4 crs. 4 hrs.

APL is a powerful computer language developed in the early 1960's for application to math and science. Since then its capability was broadened to include applications in Business, Computer Assisted Instruction, text analysis and statistical analysis.
Students will have "hands-on" experience at APL terminals to do their programming practice and assignments.
Prerequisite: 11th Year Math through Intermediate Algebra or MAT 03.

+ CP 51 PROGRAMMING in FORTRAN IV

4 crs. 4 hrs.

An introduction to programming including flow-charting using the FORTRAN IV program language. Examples are from business and mathematical applications. Programs written by the students will be run on the remote job entry communication network connected to the University Computer Center.
Computer Science Majors will NOT receive credit for this course.
Prerequisite: Three years of high school mathematics including Intermediate Algebra or Eleventh Year Math, or MAT 03.

+ CP 52 INTRODUCTION to COBOL PROGRAMMING

4 crs. 4 hrs.

An introduction to programming using Common Business Oriented Language. Typical business applications will be considered. Programs written by the students will be run on the remote job entry communication network connected to the University Computer Center.
This course is recommended as an elective for students who wish to learn how to program a computer using a language which is oriented to the solution of business problems.
Data Processing Majors will NOT receive credit for this course.
Prerequisite: MAT 01, or Elementary Algebra, or MAT R2.

+ CP 53 PL/1 PROGRAMMING

4 crs. 4 hrs.

A detailed study of PL/1 programming. Both business and scientific applications will be considered. Programs written by the students will be run on the remote job entry communication network connected to the University Computer Center.
Recommended as an elective for Data Processing Majors.
Computer Science Majors will NOT receive credit for this course.
Prerequisites: Familiarity with at least one programming language and Intermediate Algebra or its equivalent.

+ CP 54 INTRODUCTION to PASCAL

4 crs. 4 hrs.

In this introduction to programming in Pascal, students will write and run several scientific and commercial programs. This course also serves as a preparation for other high-level languages.
Recommended as an elective for Data Processing and Computer Science Majors.
Prerequistes: Familiarity with at least one programming language and Intermediate Algebra or its equivalent.

BIBLIOGRAPHY

1 Baer, J.L. and Ellis, C.S., "Model, Design, and Evaluation of a Compiler for a Parallel Processing Environment", IEEE Trans. on Software Engineering, Vol. SE-3, No. 6, Nov. 1977.

2 Baer, J.L., Gardarin, G., Girault, C. and Roucairol, G., "The Two-step Commitment Protocol: Modeling, Specification and Proof Methodology", Proc. of 5th Int'l Conf. on Software Engineering, San Diego, Calif., March 1981.

3 Bard, Y., "PERFEX- A prototype performance modeling expert system",IBM Cambridge Scientific Center, G320-2153, May 1986.

4 Chevasin, K. and Priemer, R., "Robot Activity Simulation and Plan Testing", ISA Transactions Vol. 23 No. 4, pp. 51-54 (1984).

5 Cohen, B.L., "A Powerful and Efficient Structural Pattern Recognition System", Artificial Intelligence 9 (1978), pp. 223-255, North-Holland Publ. Co.

6 deAntonellis, V., et al , "Use of Bipartite Graphs as a Notation for Data Bases", Information Systems, Vol. 4, 1979, pp. 137-141.

7 Douglass, R.J., "A Qualitative Assessment of Parallelism in Expert Systems", IEEE Software Vol. 2, pp. 70-81, May 1985.

8 Forgy, C.L., "OPS5 User's Manual", Technical Report CMU-CS-81-135, Carnegie-Mellon Univ., 1981.

9 Forgy, C.L., "Rete: A Fast Algorithm For the Many Pattern/ Many Object Pattern Match Problem", Artificial Intelligence 19 (1), pp.17-37, September 1982.

10 Genrich, H.J. and Lautenbach, K., "The Analysis of Distributed Systems by means of Predicate/Transition Nets", in Semantics of Concurrent Computation, G. Kahn, Ed. New York : Springer-Verlag, 1979, pp.123-146.

11 Gold, E.M., "Deadlock Prediction: Easy and Difficult Cases", SIAM Journal of Computing, Vol. 7, No. 3, Aug. 1978.

12 Gupta, A., "Implementing OPS5 Production Systems on DADO", Technical Report CMU-CS-84-115, Carnegie-Mellon Univ., Mar. 1984.

13 Hack, M., "Decision problems for Petri nets and vector addition systems", Computation Structures Group Memo 95-1, Project MAC, M.I.T., August 1974.

14 Hayes-Roth, F., et al Eds. "Building Expert Systems", Addison-Wesley Publ. Co., Reading, Mass.,1983.

15 Jensen, K. "Coloured Petri-Nets and the Invariant Method ", DAI MI-PB-104, Aarhus Univ. , Oct. 1979, pp.1-27.

16 Karp, R.M. and Miller, R.E., "Parallel Program Schemata", IBM Research Report RC-2053, April 1968.

17 Keller, R.M., "Vector Replacement Systems: A Formalism For Modeling Asynchronous Systems", TR 117, Computer Science Laboratory, Princeton Univ., Princeton N.J., Dec. 1972.

18 McAloon, K., "Petri Nets and Large Finite Sets", Theoretical Computer Science, Vol 32, (1984), pp.173-183.

19 McDermott, J., "R1: A Rule-based Configurer of Computer Systems", TR CMU-CS-80-119, Carnegie-Mellon Univ., April 1980.

20 McDermott, J. and Forgy, C., "Production System Conflict Resolution Strategies", in *PDIS*, Waterman et al Eds., Academic Press, N.Y.1978.

21 Mekly, L.J. and Yau, S.S., "Software Design Representation UsingAbstract Process Networks", IEEE Trans. on S.E. Vol. SE-6 No. 5, (1980) pp.420-435.

22 MIT Laboratory for Computer Science, "CSG Memo 168", Computer Structures Group Progress Report, 1974-1975.

23 Murata, T., "Synthesis of Decision-Free Concurrent Systems for Prescribed Resources and Performance", IEEE Trans. on Software Eng. Vol. SE-6, pp. 525-530, Nov. 1980.

24 Nutt, G.J., " The Formulation and Application of Evaluation Nets", Ph.D. Diss., Univ. Wash., Seattle, 1974.

25 Ozsu, M.T., "A Net Simulator as a Performance Evaluation Tool for Distributed Data Base Systems", in Proc. Convention Informatique Vol. A , 1984, pp. 154-159.

26 Ozsu, M.T., "Modeling and Analysis of Distributed Database Concurrency Control Algorithms Using an Extended Petri Net Formalism", IEEE Trans. on Soft. Eng. Vol. SE-11 No. 10, Oct. 1985, pp. 1225-1239.

27 Pasik, A. and Schor, M., "Table-driven Rules in Expert Systems", SIGART Newsletter No. 87, Jan. 1984, pp. 31-33.

28 Peterson, J.L., "Petri Net Theory and the Modeling of Systems", Prentice-Hall, Englewood Cliffs, N.J. (1981)

29 Petri, C.A., "Communication with Automata", Final report, Vol. 1, Supplement 1 to RADC-TR-65-377, Vol. 1, Griffiss AFB, New York (1966), orig. in German; 'Kommunikation mit Automaten', Univ. of Bonn (1962).

30 Priese, L., "Automata and Concurrency", Theoretical Computer Science, Vol 25, (1983), pp.221-265.

31 Ramamoorthy, C.V. and Ho, G.S., "Performance Evaluation of Asynchronous Concurrent Systems Using Petri-Nets", IEEE Trans.on Soft. Eng. Vol. SE-6 No. 5, (1980) pp. 440-449.

32 Ramchandani, C., "Analysis of Asynchronous Concurrent Systems By Timed Petri-nets", MIT Project MAC TR-120, February 1974.

33 Reisig, W., "Petri Nets: An Introduction", EATCS Monographs on Theoretical Computer Science, Vol. 4, Springer-Verlag, 1985.

34 Voss, K., "Using predicate/transition nets to model and analyze distributed database systems", IEEE Trans. on Software Engineering, Vol. SE-6, pp.539-544, Nov. 1980.

35 Waterman, D.A. et al, "Design of a Rule-Oriented System for Implementing Expertise", The Rand Corp., N-1158-1-ARPA, May1979, Santa Monica, Calif.

36 Weiss, S. and Kulikowski, C., "A Practical Guide to Designing Expert Systems", Rowan and Allanheld Publ., Totowa, N.J., 1984.

37 Zisman, M.D., "Use of Production Systems For Modeling Asynchronous, Concurrent Processes", in 'Pattern-Directed Inference Systems', Waterman and Hayes-Roth Eds., Academic Press, New York 1978.