COMPUTER AIDED DESIGN FOR PETRI NETS

by

John Kyratzoglou

S.B., City College of New York
(1981)

S.M., Massachusetts Institute of Technology
(1984)

SUBMITTED TO THE DEPARTMENT OF
MECHANICAL ENGINEERING
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF

MECHANICAL ENGINEER

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

August 1987

Signature of Author _____
Department of Mechanical Engineering
August 15, 1987

Certified by _____
Alexander H. Levis
Thesis Supervisor

Accepted by _____
Ain Sonin
Chairman, Departmental Committee on Graduate Students
Department of Mechanical Engineering

Computer Aided Design for Petri Nets

by

John Kyratzoglou

Submitted to the Department of Mechanical Engineering on
August 15, 1987 in partial fulfillment of the requirements
for the degree of Mechanical Engineer

Abstract

The development of an interactive Petri Net Editor, used for computer aided construction and analysis of arbitrary organizational architectures is presented. The Petri Net Editor consists of four major modules. The Graphics Module, which utilizes a set of graphical abstractions, is used to generate or modify structures, create or delete supernodes, form a large structure from smaller ones and select a flow path from a set of possible flow paths. The graphical representation generates a data structure which serves as the input to a number of algorithms for computing structural properties and measures of performance. The Analysis Module contains algorithms for computing the structural properties of an organizational architecture. The Text Module generates a data base to store and retrieve attributes represented by graphical abstractions. The Hardcopy Module produces output on a printer and a plotter. A User's Document is appended.

Thesis Supervisor: Alexander H. Levis
Title: Senior Research Scientist

Acknowledgements

I wish to express my gratitude to my advisor, Dr. Alexander H. Levis, for his guidance and suggestions throughout this work, as well as for presenting me with the opportunity to get exposed to the fascinating areas of C^2 Theory and Perfomance Evaluation. I also thank him for the advice and discussions that I had with him, and I hope I will continue to have, that help me to evolve my thinking and ideas.

I also wish to thank my friend and fellow-student Stamos K. Andreadakis for his encouregment and extremenly useful suggestions for the development of the Petri Net Editor. The late evening walks with him at the North Shore cleared my mind and stimulated my thinking.

I wish to thank my mother Anastasia and my sister Penelope for their continuous encouregement and to whom this thesis is dedicated to.

Finally, I would like to thank my friend Kathy Mullen for proof-reading this thesis. I enjoy her pleasant companionship and fine character.

This research was carried out at the MIT Laboratory for Information and Decision Systems with support provided by the Basic Research Group of the Technical Panel on C^3 of the Joint Directors of Laboratories and the Office of Naval Research under a contract No. $N00014-85-K-0782$.

This thesis is dedicated to my mother and my sister
Anastasia and Penelope

Table of Contents

List of Figures

List of Tables

Chapter 1

INTRODUCTION

1.1 Background and Motivation

Processes requiring a small number of tasks for their completion can be monitored or implemented by a single person. As the size of the process and the number of tasks increases, a single person cannot do the job. Large scale processes are often very complex and, consequently, difficult to control by a single trained person. For example, an electric power plant is a large scale system, and its operation cannot be controlled by a single operator. The operation of an air traffic control center is a large scale process. The information processing and decision making tasks are undertaken by teams of well trained professionals. Hence, there is a need to decompose the overall task into a number of coupled small tasks, which together provide the desired function. These smaller tasks may be equal or unequal, and are divided (distributed) among many operators or decision makers.

Performance and reliability considerations require the coordination of human resources. The act of coordinating the tasks raises the need for the set-up of an organization according to certain rules (doctrine). Those rules may have been established through past experience or belief. For example, the military is organized into ranks, with each subordinate reporting to his superior (hierarchical structure). Students receive their homework at the same time, work independently, and submit their solution to the professor (parallel structured process). Both are examples of different organizational structures. However, since the organizational process or system may be distributed, the structure of an organization cannot always be classified as either strictly parallel or hierarchical. In each organizational structure, there is different information available at different stations at different times. Information processing activities

may take place concurrently or asynchronously. Information processed by the operators must be assessed and evaluated without conflicting with the previous or the next incoming information.

Organizational performance and reliability specifications induce the need for modeling and design. For example, the normative analysis and design of Distributed Decision Making Organization (DDMO) processes is a problem of very large dimensionality, due to the number of tasks involved. The system designer must be familiar with all aspects of the process. Then, through heuristics, assumptions, and experience, must reduce the dimensionality, develop, and evaluate the designed structure. The designer must also be familiar with the existing mathematical tools that will help in modeling and analyzing the organization and carry out the design that meets the specifications.

Few methodologies exist for an efficient organizational design, see [1]. Usually, the system designer relies on a trial and error effort in developing the final design. Thus, the designer should be familiar not only with all the aspects of the system, but also with the analytical and computational tools available that will help him to carry through system design and analysis. Hence, the designer should have at his disposal a software library that contains generic and reliable algorithms and computer aided design tools. How to use productively the software library to develop the system model should depend on the designer's approach and style.

Fig. 1.1 depicts the conceptual sequence of steps undertaken in developing the model of an organization. In this case the designer uses the Petri Net Computer Aided Design (PN/CAD) System, the subject of this thesis, to built the initial model and generate the organizational structure. The organizational structure is then evaluated and analyzed in terms of the performace and reliability specifications. If the results of the analysis are not satisfactory the designer loops back to modify the model. The sequence of iterations may be repeated any number of times until the design is found to be satisfactory. Thus, design flexibility must be an inherent property of the PN/CAD System.

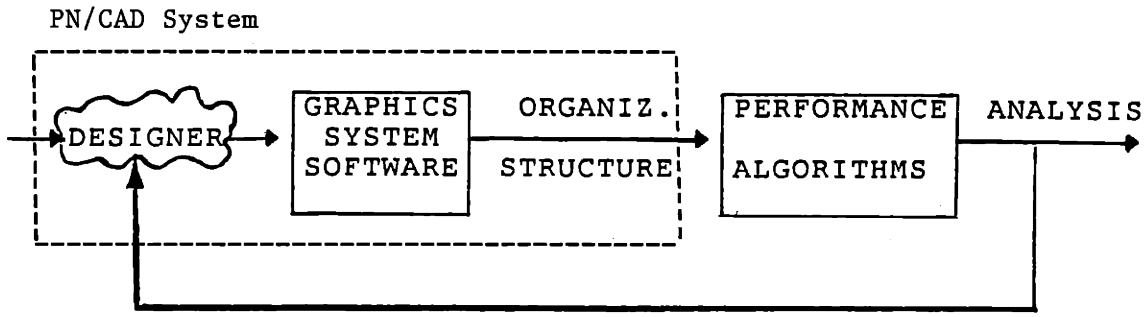


Figure 1.1: The design process and the PN/CAD System block

1.2 Aims of the Thesis

This thesis is concerned with the development of a Petri Net Computer Aided Design System that would allow modeling, analysis and design of Organizations. The class of organizations under consideration is very wide; it is only limited by the analysis and evaluation algorithms that have been implemented.

The use of the PN/CAD System will improve the productivity of the designer by introducing CAD technology to organizational design, and allow the designer to consider many alternative organizational structures. Thus, the systems designer can restructure the organization in an intuitively clear way by using analytical tools, created for assisting him. The PN/CAD System can also be used as a means for broadening the perceptions of a designer, by allowing him to explore new organizational architectures. The availability of the PN/CAD System will shorten the gap between theory and applications and its purpose is to improve the quality of the designs.

1.3 Related Work

The number of commercially available software packages addressing the organizational design problem is very limited. The application package 'Design' developed by the Meta Software Corporation is a design package that helps to generate graphically a

broad class of graphs including Petri Nets. There is also an open architecture, which is commercially available and runs on the Macintosh. The 'GACOT' package, developed by ITT Labs in Madrid on a DEC-VAX and running under both UNIX and VMS operating system, provides also a set of tools for Petri Net modelling. Other design packages are application oriented.

1.4 Thesis Structure

The thesis is structured as follows: Chapter 2 provides an outline of the basic concepts and definitions of Petri Net theory and organizational design. A short description of performance evaluation tools is also given. Chapter 3 describes in detail the structure of the PN/CAD System and its Functional Decomposition. In Chapter 4, two practical examples are used to illustrate the capabilities of the System. In Chapter 5, some empirical observations are stated with some suggestions on future enhancements.

Chapter 2

CONCEPTS AND DEFINITIONS

2.1 Introduction

The Petri Net formalism has been used for modeling asynchronous distributed organizational systems or processes. Petri Nets are bipartite directed multigraphs [2]. They contain two types of nodes: 1) the places, indicated by circles, and denoting signals or conditions; and 2) transitions indicated by rectangles, and denoting activities of processes. Places can only be connected to transitions and vice versa. An arrow denotes the directed relationship between the place and the corresponding transition. Each place, transition, or arrow (connector) may have specific inscriptions associated with it related to the physical component or process it represents. For example, each transition may contain a set of algorithms modeling a specific organizational activity. For an activity to take place, certain conditions must hold true. Then, the places describe the set of conditions (cases or tests).

To represent the decision structure, the concept of the switch is introduced [3]. A switch is considered as a sophisticated transition and represents multiple alternative decision strategies. A switch implies the following action. A particular set of conditions is tested according to a control law based on the transmitted information. The conditions are evaluated, a decision strategy is selected and an action is taken (an algorithm is executed).

2.2 Petri Net Theory

2.2.1 Introduction

A Petri Net [4] is a triple N = (S; T; F), such that S and T are disjoint nonempty sets with a relation F between them, such that every element of S and T occurs in the relation F :

1. $S \cup T \neq 0$
2. $S \cap T = 0$
3. $F \subseteq (S \ x \ T) \cup (T \ x \ S)$
4. $dom(F) \cup range(F) = S \cup T$

where,

S is the set of S-elements, or places

T is the set of T-elements, or transitions

$X = S \cup T$ is the set of all elements of N

F is the flow relation

The S-elements can be represented graphically by circles, the T-elements by rectangular boxes, and the flow relation by directed arcs. The physical meaning of the elements is that the S-elements denote local states, the T-elements denote local transitions and the flow relation denote the extent of change caused by the local transitions of the system. Fig. 2.1 shows a typical Petri Net.

Let x be any node element of the Petri Net, then

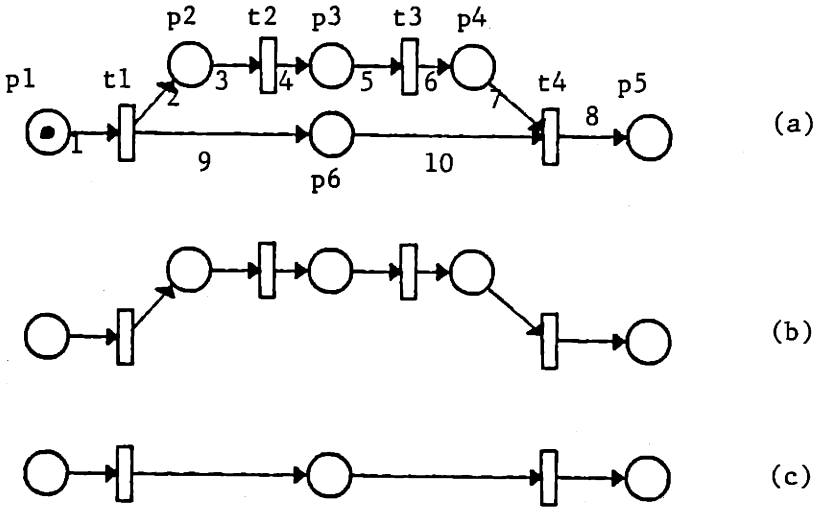


Figure 2.1: A Petri Net (a) and its information flow paths (b, c)

$.x = y \in X | (y, x) \in F$ is the set of all input node elements of the node element x, called the pre-set of x.

$x. = y \in X | (x, y) \in F$ is the set of all output node elements of the node element x, called the post-set of x.

By introducing a number of restrictions on the flow relation F, different classes of Petri Nets can be obtained. The Petri Net $N = (\ S;\ T;\ F)$ is said to be pure, if there are no loops, i.e. directed paths from a node back to the same node. A special case of a loop, called the self-loop, is shown in Fig. 2.2.

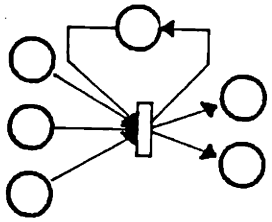


Figure 2.2: Self-loop representation

If there are no two nodes in a Petri Net $N = (S; T; F)$ which have the same post-set and same pre-set, then the Net is called simple.

Definition.A Petri Net is called connected if there is a flow relation
from any node to any other node.

In the present work, all Petri Nets are assumed to be pure, connected and finite (i.e. the set of elements is finite). To characterize the structure of a Petri Net N, linear algebra representations are introduced. One particular representation is the incidence matrix, which provides a description of the interactive behavior among the nodes. For a Petri Net N which has m S-elements, and n T-elements, the indicence matrix C is an m x n matrix, defined by the relation,

$$C_{ij} = \begin{cases} -1 & \text{if s is an input to t} \\ +1 & \text{if s is an output to t} \\ 0 & \text{if there is no flow relation between s and t} \end{cases}$$

Similarly the interconnection matrix M of a Petri Net, is defined as an m x n matrix,

$$M_{ij} = \begin{cases} -(CN) & \text{if s is an input to t} \\ +(CN) & \text{if s is an output to t} \\ 0 & \text{if there is no flow relation between s and t} \end{cases}$$

where,

(CN) is the connector's identification number.

The interconnection and incidence matrices of the Petri Net of Fig. 2.1, are shown in Table 2.1.

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 2 & -3 & 0 & 0 \\ 0 & 4 & -5 & 0 \\ 0 & 0 & 6 & -7 \\ 0 & 0 & 0 & 8 \\ 9 & 0 & 0 & -10 \end{bmatrix} \qquad \begin{bmatrix} -1 & 0 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 1 & -1 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & -1 \end{bmatrix}$$

Table 2.1: INTERCONNECTION AND INCIDENCE MATRICES

2.2.2 Marking and the Firing Rules

Data flow carriers are symbolized by tokens. The distribution of tokens in the Net N = (S; T; F) is called the marking M. A marking M is a function M:S $\mapsto$ (0 1 2 3 $\cdots$). The Petri Net represents the structure of the organizational process, and the marking M represents a state of the organization.

A marked Petri Net is a quadruple $G = (S, T; F, M_o)$. M_o is called is called the initial marking of G. An example of a marked Petri Net is shown in Fig. 2.1. In this case the initial marking vector is $M^T = (1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0)$.

Let M be a marking of the Petri Net N = (S, T; F). Then, a transition node t is enabled at M, if the token content of each input place to transition t, is greater than zero. Mathematically, $\forall \ s \in .t, \ M(s) > 0$.

If t is enabled, it can fire. When a transition fires at M, a new marking M' is obtained. In symbols, $M_o \xrightarrow{t} M'$. Then, a token is withdrawn from each of the input places and a token is deposited in each of the output places. Mathematically, $\forall s \in S$

$$M'(s) = \begin{cases} M(s) - 1 & \text{if s } \epsilon \text{ .t} \\ M(s) + 1 & \text{if s } \epsilon \text{ t.} \\ M(s) & \text{otherwize} \end{cases}$$

Let us denote by$M'\{M/t\}$, the transformation of M into M' through the firing of

transition t. Then, the state space generated by an initial marking M_o of a marked Petri Net $G = (S, T; F, M_o)$, through a sequence of transition firings is called the forward marking class defined by M_o. Lets us denote the forward marking class by $\{M_o \rightarrow M_f/t\}$. The above concepts lead to the following proposition.

Proposition.If M and M' are members of a forward marking class and $t \epsilon T$ and M'' is a marking of the Petri Net N, generated from M' through firing of t, then M'' is also a member of the forward marking class.

Let $\vec{v}$ be a firing sequence from a marking M towards a marking M', where both M, $M' \epsilon \{M_o \rightarrow M_f/t\}$. Then, the following rule can be deduced:

$$M' = M + C \cdot \vec{v} \tag{2.1}$$

This equation is the basis for subsequent application of linear algebra methodologies. It states that the new marking M' is generated from the old marking M by operating on the incidence matrix.

2.2.3 Liveness and Boundedness

Liveness, Boundedness are two of the most important properties of marked Petri Nets. Let $G = (S, T; F, M_o)$ be a marked net. Then,

- G is live if for each $M \epsilon \{M_o \rightarrow M_f/t\}$, and each $t \epsilon T$ there exists a marking $M' \epsilon \{M_o \rightarrow M_f/t\}$, such that t is enabled at M'.
- G is safe iff for each $M \epsilon \{M_o \rightarrow M_f/t\}$, the token content of each S node, where $s \epsilon S$, never exceeds one, $M(s) < 2$.
- G is bounded iff for every reachable $M \epsilon \{M_o \rightarrow M_f/t\}$, the number of tokens in each s node, where $s \epsilon S$, has a finite upper bound K.

Safety is a special case of boundedness with $K = 1$. If for a particular marking M of a Petri Net N, a condition is reached such no transition can be enabled and fired, then the Petri Net has reached a deadlock configuration.

2.2.4 Fundamental Notions

The following notions [5] are fundamental ones in the Petri Net theory and can be brought out with the help of safe marked nets.

1. Causality, where occurence of activity B must be preceded by occurence of activity A, see Fig. 2.3,

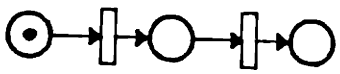


Figure 2.3: Causality

2. Conflict, where either decision strategy A or B, but not both, can be executed, see Fig. 2.4,

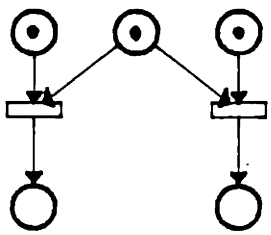


Figure 2.4: Conflict

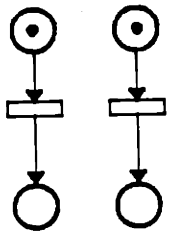


Figure 2.5: Concurrency

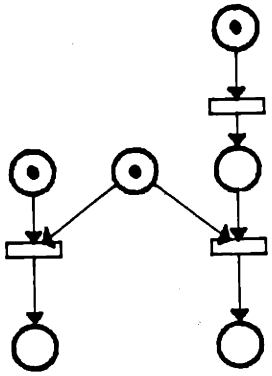


Figure 2.6: Confusion

3. Concurrency of activities, see Fig. 2.5, and

4. Confusion, which is overlap of confict and concurrency, see Fig. 2.6.

The aforementioned notions make the Petri Net model appropriate for modeling organizations.

2.2.5 Graph Theory and Petri Nets

A number of ideas have been borrowed from graph theory and topology to illustrate the Petri Net concept. Since Petri Net theory is concerned with the flow of tokens through the net in a way such that some of the flows are synchronized, some are branched, and

some are merged, the following definitions are introduced.

Definition:A Petri Net is strongly connected if there exists a directed path from every node to every other node.

The above definition leads to the definition of a special class of Petri Net graphs, called event graphs. These graphs possess special properties and are tractable for analysis.

Definition:A connected Petri Net graph in which every place has one input transition and one output transition is called an event graph.

Fig. 2.1 is an event graph.

Definition:Let $\vec{x}$ be a n-dimensional vector, such that $\vec{x} \in N$, where n is the number of s nodes in the Petri Net. Then $\vec{x}$ is called an S-invariant iff $\vec{x}^T \cdot C = 0$

Definition:The set of s nodes whose corresponding components in $\vec{x}$ are strictly positive is called the support of $\vec{x}$ and is denoted by $\| \vec{X} \|$.

Definition:The support $\| \vec{X} \|$ of an S-invariant, $\vec{x}$, is said to be minimal if and only if it does not contain the support of any other invariant, but itself and the empty set.

Definition:An S-component is the subnet whose set of s nodes is $\| \vec{X} \|$ and whose set of t nodes consists of all transitions connected to the s nodes of $\| \vec{X} \|$ by a flow relation F .

Let $\| \vec{X}_1 \|, \| \vec{X}_2 \|, \cdots, \| \vec{X}_i \|$ be the minimal supports of S-invariants of a Petri Net. Then, an information flow path (or simple path) of that Petri Net is the subnet whose set of s nodes is $\| \vec{X}_i \|$ and whose set of t nodes consists of all the transitions connected to s nodes of $\| \vec{X}_i \|$ by a flow relation F. The set of all information flow paths of a Petri Net defines a complete flow path. The lower part of Fig. 2.1 depicts the information flow paths of the Petri Net. The minimal support matrix is shown below,

$$\begin{vmatrix} \| \vec{X}_1 \| \\ \| \vec{X}_2 \| \end{vmatrix} = \begin{vmatrix} 1 & 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 1 & 1 \end{vmatrix}$$

2.3 Review of Organizational Theory

2.3.1 Introduction

Organizational theory addresses organizational processes and the systems that support them. N-dimensional information theory is used as the mathematical framework for modeling the Decision Making Organization (DMO) process and the single interacting decision makers (DM). On a larger scale, the organization which contains many decision makers (DMs) and its decision support systems (DSS) is called a distributed decision making organization (DDMO). The interactions between single decision makers, members of a DDMO, are modeled using Petri Nets [2], [6] and [4], and are described in section 2.3.2 .

The combinatorial design problem of finding the maximally and the minimally connected organizational forms given a set of structural and design constrains was investigated by Remy [7]. Performance evaluation measures, such as timeliness, accuracy, organizational response, and workload have been defined in [8] and [9], and are described in Section 2.4 . Implementation of algorithms quantifying measures of performace can be found in [10], [11] and [12]. Since the decision makers are assumed to have limited memory capabilities, the bounded rationality constraint has been introduced [13].

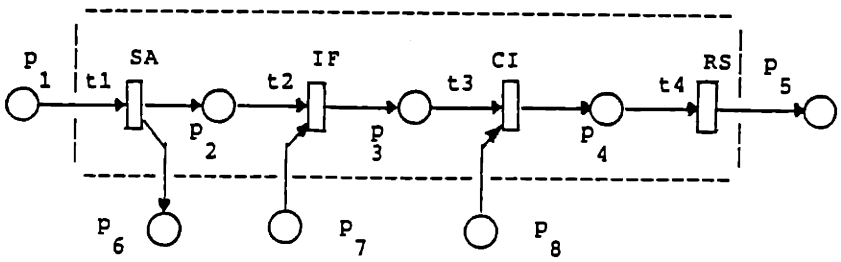


Figure 2.7: Single interacting DM

2.3.2 Decisionmaking Organizations

A DDMO maybe either parallel or hierarchically structured. The DDMO may interact with other DDMOs or with the environment. The information is treated as a resource unit (token), and it is allocated either to a single interacting DM or to the DDMO. Each DDMO consists of smaller processing DMs. The single interacting DM is modeled as a four stage decision process. As the information is received, it is being processed at the first stage for situation assessment (SA). In the second stage there is information fusion (IF) of the situation assessment output with the SA's and responses of other DMs. At the fourth stage, all the data processing has to be translated into a subset of responses (response selection, RS). The range of responses may be restricted by the command interpretation (CI) stage that precedes the RS stage. Each stage (representing activity) is modeled as a transition, while the conditions that must be fulfilled for the activity to take place are modeled by places. The general Petri Net model of single interacting DM is shown in Fig. 2.7.

The way that different stages of the DMs are interlinked defines the structure of the DDMO. As the information is received, it can be processed concurrently by the stages of each DM member of the DDMO. A comprehensive treatment of the allowable ways of interlinking stages of different DMs, interactions of DMs with the enviroment

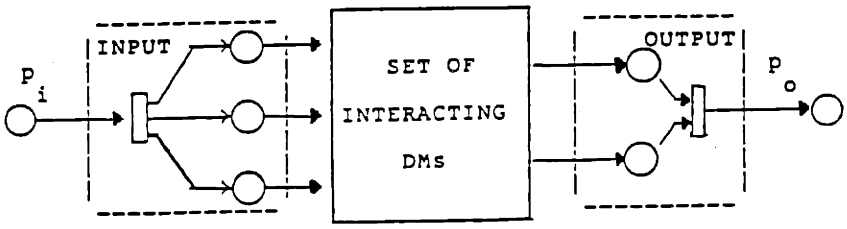


Figure 2.8: Interacting DMs (DDMO)

and the mathematical stuctures used to represent the DDMOs are given in [7]. The model of the DDMO without the limited resources constraint, interacting with the enviroment, is shown in Fig. 2.8. The maximum number of interacting DMs for the present software application is five.

2.4 Supernodes

The existence of supernodes in a Petri Net diagram denotes the existence of hierarchies in the structure. It may be viewed also as a Net reduction technique. Replacing a large and complicated subnet with a single super element simplifies the overall structure, while preserving the original interconnectivity. With hierarchical structuring the designer can deal with each level independently, making it easier to work with. Flow connections are between functions on the same level. Connections between hierarchies must be compatible, that is, the flow connection of the top hierarchy must also be the input flow connection at the lower hierarchy, see Fig. 2.9 . There is no standard way of defining a supernode. In this work, a plausible identification of a supernode is done in terms of its data flow paths.

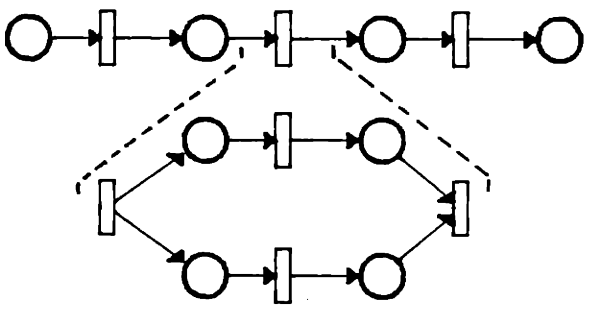


Figure 2.9: Representation of a supertransition

As a second example of a supernode structure, a refined version of the basic 4-stage DM shown in Fig. 2.7 is presented in Fig. 2.10. In the refined version both SA and RS stages contain substructures, such as a 2 and a 3-decision switch, respectively. Each switch setting represents a decision strategy. The implementation of that strategy depends on the tested conditions and the probability of occurence associated with that decision strategy. This model is discussed in Boettcher and Levis [12]. In this example, p_1 is the input node and p_2, p_3 are the output nodes of the supernode t_1 . Similarly, p_4 and p_5 are the input and output nodes, respectively, of the supernode t_4.

Two more supernode structures are shown in Fig. 2.11 . In Fig. 2.11à the simple paths dp_1, dp_2 and dp_3 coming out of the p_1 node are branching to the same type of node p_2. There should be no functional (data flow) relation from any node member of the supernode structure to any node outside the supernode (except to the input and output nodes p_1 and p_2). Similarly Fig. 2.11b shows a superplace structure, where t_1 and t_2 are the input and output nodes, respectively, of the supernode.

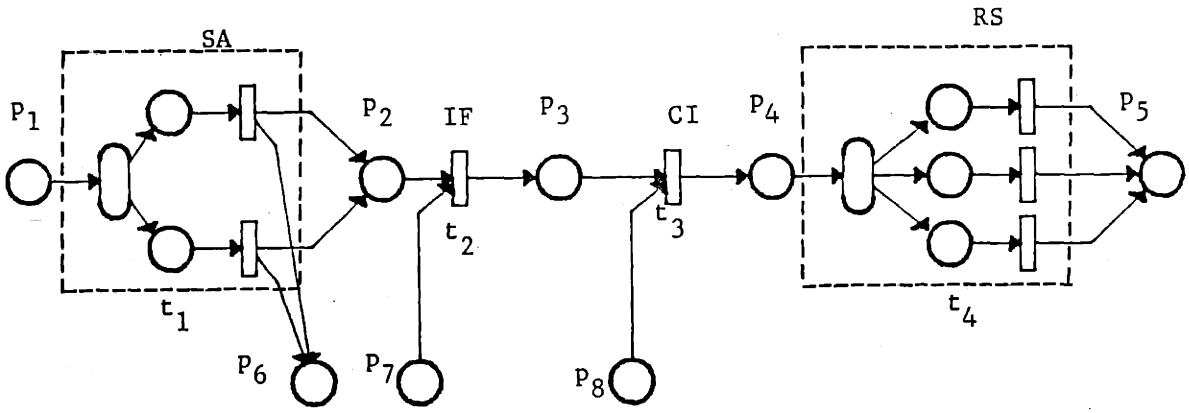


Figure 2.10: A refined version of the DM

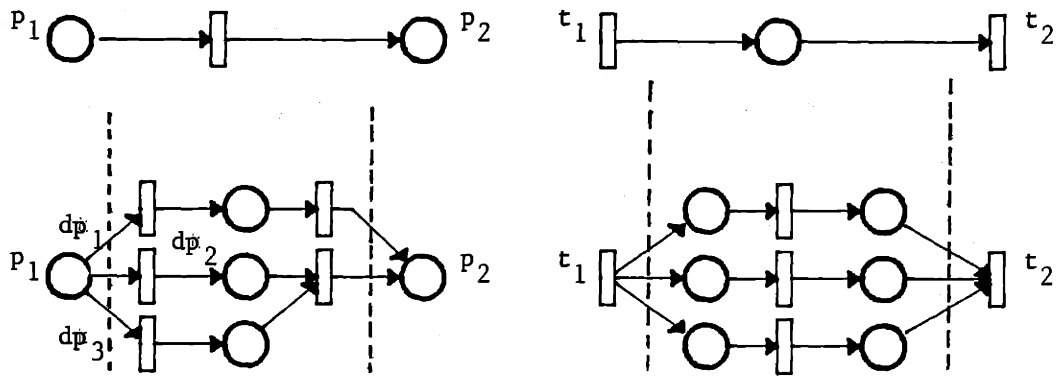


Figure 2.11: Supertransition and superplace representations

2.5 Evaluation Tools

The behavior of the Petri Net model of a DDMO can be viewed by the set of basic structural properties. These properties can be expressed algebraically. As a result, linear algebra techniques can be applied in evaluating the performance attributes of the net. The basic structural properties used in this thesis are:

1. The incidence matrix C_{ij}, where the rows of C are the places, and the columns of C are the transitions. Each entry of C indicates if there is a $s_i t_j$ connection and,
2. S and T-invariants of the Petri Net.

The structural properties of a system may be used to calculate its attributes. One can later on consider the calculation of Measures of Performance such as organizational delay, accuracy of the response, and workload of the DM.

Organizational delay is the time taken for an organization to process a single input.

Accuracy of the response is the degree to which the actual DMO response matches the ideal response.

The Workload of a DM is the amount of mental effort that each decision maker applies to perform a task. The N-dimensional information theory is used for workload computation [14] and [15]. Total activity is the measure of information processing workload of each DM. The bounded rationality constraint is introduced as an upper bound on the information processing capability of a single interacting decisionmaker.

The DDMO designer could be greatly assisted if he had at his disposal a software system containing graphical algorithms. The graphical software design system could greatly facilitate the building and modification of complex organizational structures. The use of the software system will increase the design quality and reduce design time. The architecture of such a Software System is described in Chapter 3.

Chapter 3

SOFTWARE SYSTEM ARCHITECTURE

3.1 Introduction

This chapter is devoted to the description of the PN/CAD System architecture. The software system structure can be analyzed in terms of the following levels:

- the Top Command Level
- the Functional Level
- the File System
- the Grammar-Rule Checker
- the Software-User Graphics Interface

The purpose of the Top Command Level is to provide an interactive interface between the user and the program's functions. The heart of the software system is the Functional level. It implements all the function calls. The File Manager keeps track of the existing files and stores them in the file library, if necessary. The Grammar-Rule algorithm checks for syntactical rule violations. The aforementioned levels provide a framework for the description of the major features of the software.

3.2 Software Principles

The approach taken for the development of the Software System is to construct a set of simple modules to implement low level commands and then to program the higher

level commands in terms of the low level ones. Thus, the execution of commands by the system involves levels of implementation. Each user command is evaluated in terms of several lower level commands. The motivation for such an approach is due to design and implementation considerations. A wide range of Petri Net designs can be implemented using the notion of flexible design by combining low level commands.

The IBM PC/AT supports two graphics software packages for software development. This includes the Graphics Development Toolkit (GDT) and the Graphics Kernel System (GKS). GDT and GKS provide a uniform programming interface between an application program and the conventional graphics hardware. Both packages implement the Virtual Device Interface (VDI) Controller as a DOS device driver. The main feature of the packages is that they provide the applications programmer with a set of low-level graphic functions, common to all applications, thus allowing the programmer to concentrate on the application program development. These functions enable the programmer to construct a two-dimensional picture on a graphics output device (display, printer or plotter).

GDT and GKS graphic packages have a lot of common features and to decide which one to use depends on a number of factors such as:

- Nature of application and program development
- Language bindings
- Program portability and standarization
- Software support and future enhancements

The goal of the program was discussed in Chapter 1 . To achieve the graphical construction of the Petri Net model, a set of building blocks (primitives) is needed. These graphics primitives are user-constructed and represent the simplest entities with which the overall graphics application is concerned. The graphics software package used must provide the following mechanisms for accomplishing the task:

- Each graphic primitive must be dealt as an individual part (segment) in the overall construction of the Petri Net.
- A way to combine the graphic primitives into a compound graphic image (segment). The graphic image must have a name associated with it and be dealt as a unit.
- Support a wide range of graphic manipulation procedures including transformation, screen detectability, visibility, and highlighting of the segments.
- Supply a rich set of inquiry functions. Those functions will allow the programmer to have a better control of the application program and exploit fully the workstation's capability. It will also allow better user-application program interchange of information and command implementation.

A careful study of the packages reveals that the GKS possesses these additional features needed for the present application. The application software under development must also address the following Software Engineering issues:

- User friendliness
- Structural programming
- Graphical abstraction
- Ability to maintain and modify the code
- Future enhancement
- Error detection
- Documentation

Lattice C, Pascal and the IBM Professional Fortran are the language bindinds supported by GKS and GDT. For engineering purposes and ability to exploit the use of data stuctures, in both the graphical and numerical stage of the program, Pascal and Fortran languages are suitable. For the development of the PN/CAD System the Fortran language was chosen.

3.3 Analysis of the Software System Structure

A brief description of each level is given in the next sections. A schematic view of the PN/CAD System architecture is shown in Fig. 3.1 . The PN/CAD Functional modules, such as the Graphics Editor, the Text Editor, the Structural Analysis and Hardcopy Module, are all interfacing with the Top Command Level. All the activity takes place at the Functional Level. In addition, a File Manager is implemented for loading, manipulating, or storing files containing data structures. Also a Grammar-Rule Checker algorithm verifies the syntactical correctness of the Petri Net structure.

3.4 Top Command Level

The Top Command Level consists of a set of algorithms that allows the PN/CAD System initialization, termination and command interpretation. When the System is invoked, diagnostic and directory files are loaded by the File System. At the Top Command environment, each designer command is identified and processed by the algorithms. The interpreter algorithm decodes the command and executes it. The execution determines what sequence of operations are to be performed next. Control is passed to the appropriate modules in the Functional Level and execution of the modules that carry out these commands takes place. Before terminating the program session, the File Manager is called to perform certain managerial duties. Additional commands can be incorporated into the system to enhance its options. The Top Command Level has a standard I/O interface with a) the user through the keyboard and/or mouse and the various display monitors, b) auxiliary devices for output spooling such as printers and plotters and a standard programming interface with the Functional Level and File System.

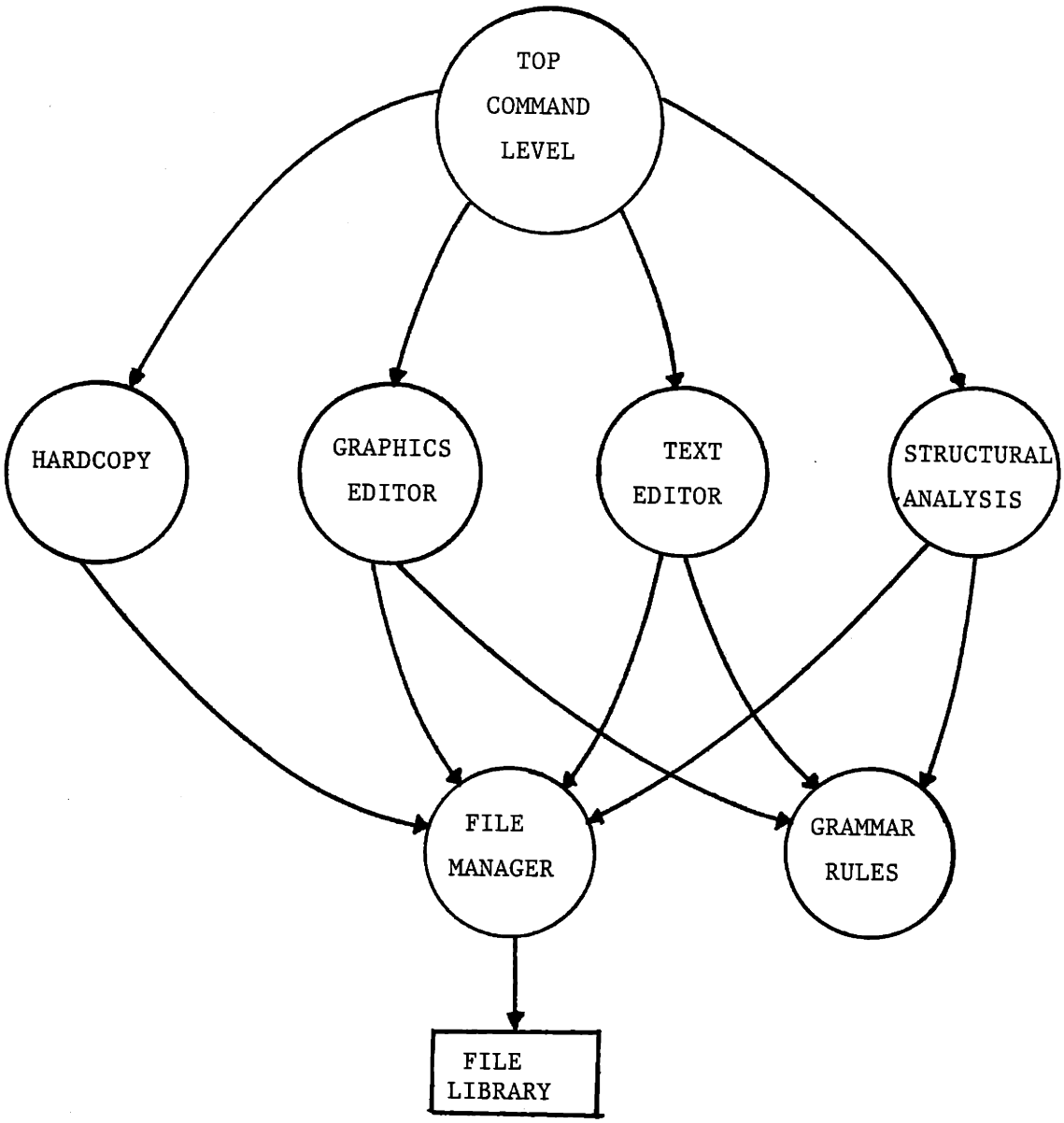


Figure 3.1: The PN/CAD System Chart

3.5 Functional Level

The user communicates with the functional level through the command level. At the functional level sets of modules are aggregated for command evaluation. The Functional organization may be classified into the Graphics Editor, Text Editor function, Stuctural Analysis and Hardcopy Module. Once a function is invoked, the designer is in the local function's enviroment (mode). The next sections describe each Functional Module.

3.6 Petri Net Graphics Editor

3.6.1 Introduction

This section describes the purpose of the Petri Net Graphics editor, its main features, its functional structure, the syntactical rules of construction and its capabilities. It also covers the basic graphics concepts used, a short description of the graphics commands available, and a technical discussion of the data structures used for graphical information storage and retrieval. A more detailed documentation of the graphics commands available, as well as a description of a sample graphics editing session, is provided in Appendix A.

3.6.2 Graphics Editor

It was stated in Chapter 2, that the Petri Net representation is used for system modeling, analysis and synthesis. The objective is to come up with an organizational design that satisfies certain performance and reliability specifications.

The PN graphics editor is an interactive set of graphical programs, which allow the system designer to develop the graphical Petri Net representation of an arbitrary organizational structure. The purpose of the Graphics editor is to furnish the designer a set of graphics tools for developing and modifying typical Petri Net structures.

The graphical representation of the organizational architecture provides better means of visualizing the complex system structure, and better means of teaching and communicating ideas to others. The system designer must be familiar with the technical constraints of the editor. The analytical description task is undertaken by the Structural Analysis Function of the Software System and is described in section 3.8.

3.6.3 Graphics Editor Features

The Petri Net Graphics editor offers specific features that make graphical construction and modification easy. The features that make the editor unique are:

- interactive development of the Petri Net model
- interactive modification of an existing Petri Net model
- interactive synthesis of a Petri Net from simpler structures
- decomposition of a large structure into simpler designer defined structures
- hierarchical ordering of organizational architectures

The whole software package is classified as a Computer Aided Design (CAD) System for Petri Net modeling of organizations. In fact, the use of the package is more general than its initial intent.

For a graphics operation to take place, the user has to be in the graphics mode. When the Graphics editor is invoked, the designer is placed at Level One Mode. While at this level, the designer can recall from the file library and display the contents of any graphics file, or use the HELP facility to get help. At Level Two, a set of diverse functions is provided that allow creation and modification of the nets.

3.6.4 Screen Organization and Feature Description

The screen area where all the information is presented is called window. The window is divided into the Workarea, where the graphical development of the Petri Net structure takes place, and the window frame. The window frame surrounds the Workarea and has a rectangular shape. The graphics window frame contains the Information Menu, indicating the working level and the file name and status, and the Graphics Command Menu providing a list of graphics edit commands. At the lower left side, a Dialog Area is reserved. In this area, the system prompts the user to enter additional information or responds to a user request. Fig. 3.2 depicts the screen organization. For the graphical creation of a Petri Net structure, a menu of the standard elements appears in symbolic form, on the Element Menu area. There are four symbols in the Element Menu which represent:

1. place
2. transition
3. switch
4. connector

The Element Menu is located on the upper right side of the screen. The graphical construction takes place on the Workarea. To select an element from the list, the user must place the keyboard controlled cursor on the element's symbol and press the enter key. If the operation is succesful, the symbol will be highlighted. The user can select repetitively elements and place them on the Workarea. In such a way, a Petri Net structure is developed.

Using the Graphics editor you can interactively create on the screen the Petri Net model of a DM. In parallel with the graphical design, the underlying data structure is formed. With every graphic element placed on the screen, a set of descriptive information attributes is created and saved with a special formated record. The final data structure is stored in a data file. Subsequent re-editing and modification of the existing

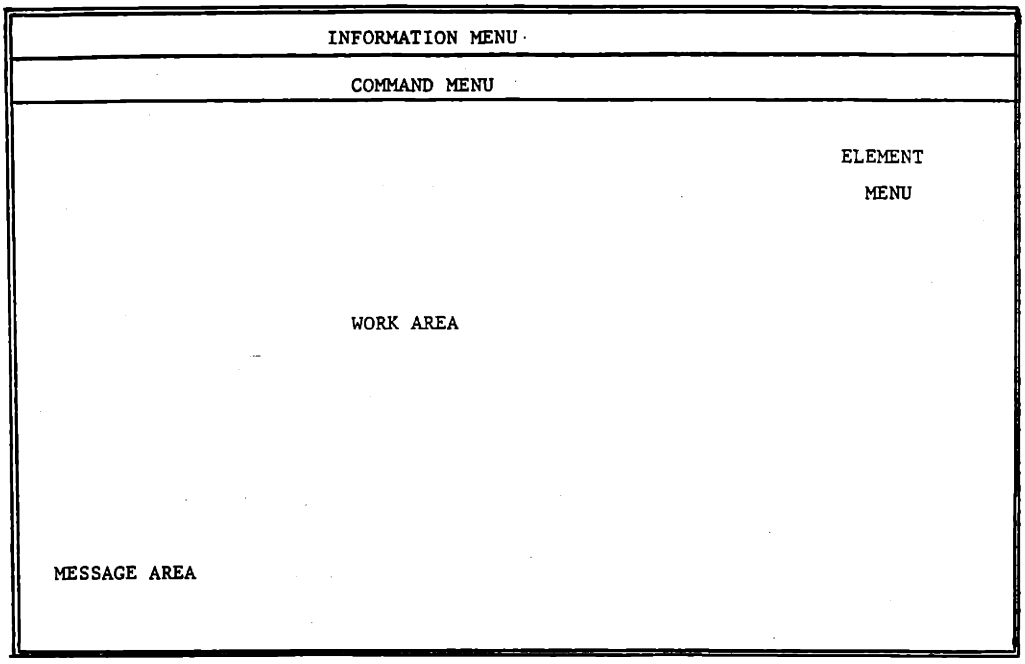


Figure 3.2: Screen Organization

structure means to access the file from the file library, extract, manipulate and restore the contents of records.

Using the ideas discussed above, the designer can define a set of generic structures, used as building blocks of a much larger structure. The building blocks can be developed and analyzed separately. The final structure is constructed by inserting all individual generic structures and synthesizing them. The composite Petri Net model must obey the design rules and may be defined as a single unit.

The reverse operation can also be done. The designer can decompose a complex structure and evaluate the resulting parts separately. The decomposition procedure takes place according to stringent decomposition rules. The assembly/disassembly of complex PN structures is a unique feature of the PN/CAD System.

But the Software's features do not stop here. The unique structure of the data records, along with the Graphics and Text editors features, provide another extra tool to the designer. It gives the designer the ability to define supernodes and work at various hierarchical levels. The designer can bound a region of a Petri Net structure

and define it as a supernode. The bounded region collapses into a standard element (superplace or supertransition) with special features. This method permits creation of different hierarchical levels. Each hierarchical level reveals the degree of nesting. Aggregation may be used recursively, until the overall structure collaspes into a single supernode. Thereafter, the use of the supernode implies the use of the aggregated network. All supernodes of a net are related to their underlying level through inter-net relations, recorded in the corresponding data record and text file. However, the available disk space poses a constraint on the aggregation level. In this implementation, a nesting of degree two (two levels) is allowed. The reverse operation of unfolding the network can be performed.

3.6.5 Graphics Editor Structure

Fig. 3.3 shows the functional structure of the Graphics editor in a Petri Net representation. Table 3.1 provides a description of the module actions.

The actions of the Graphics editor program are associated with transitions 4 through 17. At the Top Command Level the user invokes the Graphics editor. Once in graphic mode the user requests an action and selects a command. The command is evaluated by the command interpreter. If a valid command is entered the user is placed at that command's mode. There are five modes of operation at the first level. When the show mode is on, the system prompts the user for a file name. If the file exists in the file library, it is displayed. Each time a command is interpreted, the system responds with a message.

3.6.6 File Organization and Graphics Data Structure

The GKS provides a metafile or graphical information storage system. However, because the data structure is used for processing, and due to current implementation constraints, this specialized form of data storage is inadequate. Therefore, there was a need to develop a special data storage file. The File System assigns a symbolic name

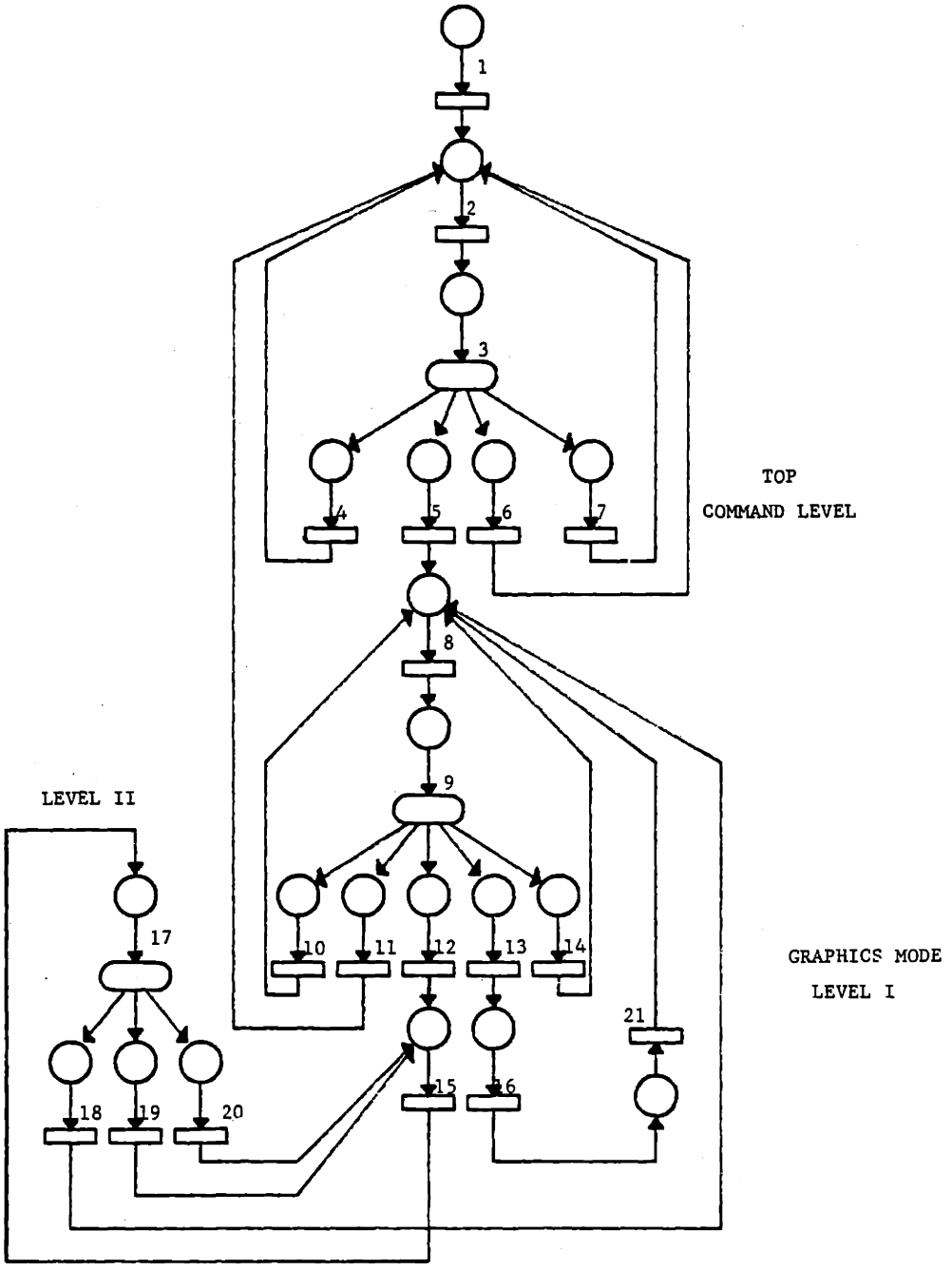


Figure 3.3: Functional Structure of the Graphics Editor

Action (Transition No.)	*Interpretation*
1	Start and setup system
2	Request user's choice
3 (switch)	Evaluate choice and enter graphic mode
4	Text Editor Mode
5	Graphics Editor Setup Mode
6	Analysis Mode
7	Hardcopy Mode
8	Request command
9 (switch)	Evaluate Command and take action
10	Help mode
11	Exit mode
12	Edit File mode
13	Show mode
14	Directory mode
15	Request file name and Command in edit mode
16	Request file name
17 (switch)	Evaluate command and take action
18	Exit to higher level
19	Level II graphics commands
20	Pick Element commands
21	File system

Table 3.1: GRAPHICS EDITOR MODULAR DESCRIPTION

to that file, with a file extension .GRF

A data file contains fixed length records. A record has a specific structure, and is divided into fields. Each field contains either geometric or data attributes of the graphics element. The geometric attribute fields contain coordinate position information. The data attribute fields contain information about the element identification number, its number in the element list, angle of rotation, supernode code, and error code. A short description of all the attributes follows:

- Coordinate position. It is a single precision floating point number (4 bytes) indicating the coordinate position of the standard element on the screen.
- Identification number. It is a two byte integer value, denoting the identification

number of the element. Identification numbers are:

  - 1, for place
  - 2, for transitions
  - 3, for switches
  - 4, for connectors

- Element list number. A two byte integer value, denoting the list number of the element on that Element's Table.

- Angle. A single precision floating point value, denoting in degrees the relative angle position of the element with respect to horizontal. It is used in switches and transitions elements.

- Supernode code. A two byte integer code value indicating if the node is a supernode or not. It is used in place and transition elements.

- Error code. A two byte integer value, indicating the error status of the element.

All the attributes are used to organize the Graphics structure on the screen. There is one sequential access data file, used to store all the information about the Petri Net structure. If some of the nodes of the Petri Net are supernodes, there is a sequential file associated with each supernode present. Each such file contains the underlying Petri Net structure. The supernode code indicates if a node is a supernode.

There are four temporary direct access files created during each graphics editing session. Each of these files contains attribute information corresponding to each standard element of the net. Using direct access mode, retrieving and storing of records is easier during modifications. At the end of an editing session, the direct access files are merged, read into the sequential access file, and deleted from the system.

3.6.7 Graphics Commands

The set of graphics editing commands included in the first level is:

- show
- exit
- dir(ectory)
- help
- edit_file

and at the second level

- help
- refine
- coarsen
- copy
- insert
- switch
- exit
- delete
- quit

Each command may have parameter options. Command parameters modify a command. They provide additional information and allow the designer to perform the following operations:

- add/delete elements
- insert/delete nets
- copy parts of the net
- display nets

Some of the editor limitations were described in the previous sections. The number of hierarchical levels allowed is two, with a maximum of 20 supernodes at each level.

3.7 Text Editor

3.7.1 Introduction

The following sections describe the purpose of the Petri Net Text editor, its features, its functional structure, and an overview of its capabilities. A description of the data structure of the edited information and Information Retrieving Facility is also given. Documentation that shows how to access the text editor and a complete description of all text commands available is provided in the User's Document, Appendix A.

3.7.2 Petri Net Text Editor

The text editor is an interactive utility program which allows the user to create or recall from the file library a text file, to enter or to modify its contents, and to save or delete the work the user has done during the text editing session. The text file contains information for all the elements of the Petri Net structure. The Text editor works only with text files that contain specific text editing and formatting capabilities.

The purpose of the Text editor is to allow the user to store information represented by places, transitions, switches or connectors. In some cases the information may be used for syntactical or semantic purposes. In other cases, for simulation or performance evaluation of systems. In general, the stored information is retrieved and used for analysis and design of complex organizational architectures.

3.7.3 Text Editor Features

The Petri Net Text editor offers specific features that make text editing easier and the information stored more accessible. These features include:

- An on-line HELP facility the user may access at any time.

- Menu driven actions.
- Interactive information access in the form of queries.
- Display of stored information.
- Node record editing.
- Internal command parser.

For the operation to take place, the user has to be in Text Mode. When in Text Mode, the user can recall from the library the file containing the textual material.

A short description of the features of the Text Edit Mode follows:

The HELP facility is available at each possible Menu level. It provides descriptive messages about menu choices, ways to implement them and how to construct text edit records of the elements. The Text Menu allows the user to examine the choices he has available, without choosing any. If the user requests so, the application program calls the functions implementing the command and performs the action. The text menu consists of a number of menu items listed vertically in the middle of the screen. The Menu items (or list of text mode commands) are:

- modify a text file
- display a text file
- edit node record
- help
- exit

The Modify command allows to enter new information or modify the old one. The user modifies the text records sequentially, according to the record number assigned by the numbering routine during the graphical creation of the Net. The insertion of

information is assisted through the use of a dialog. An *alert* mechanism reports to the user any wrong entries, whenever possible, i.e. probability of occurence is 1.2, the delay of transition No.3 is -14 seconds or token content of place No. 2 is -16. The Display command displays on the screen the stored information for visual examination. The Node Number line editing mode is useful for those that prefer random access of the records. Each record can be accessed by specifying the element type (place, switch, transition or connector) and its corresponding assigned logical number. The Exit command terminates the session and exits to higher level.

3.7.4 Text Editor Structure

Using the Petri Net representation, the functional structure of Petri Net Text editor is shown in Fig. 3.4. Fig. 3.4 depicts also the Top Command Level and Text editor interface. Table 3.2 depicts the interpretation of the module actions, laid out in pseudocode manner.

The actions of the Text editor program are associated with transitions 4 through 14. At the Top Command Level, the user enters a command, invoking either the Graphics or Text editor, or entering Petri Net analysis or hardcopy generation mode, or exiting into DOS command enviroment. The command is evaluated and a selection is made. As stated earlier, the process of selection of an algorithm from a set of algorithms that perform different functions is modeled by a switch. Node 3 is a switch, and its setting determines what selection is active. Once the user has selected the text edit mode, he enters the text mode command enviroment. The text mode command interpreter prompts the user to enter the name of the file he wants to edit. The File Manager searches the file library for the file, and returns an error code. If the error code is 0 (file exists), the interpreter opens the necessary files, otherwise it returns control to the higher level (main level). At the last stage, the user enters text mode commands for editing an information file.

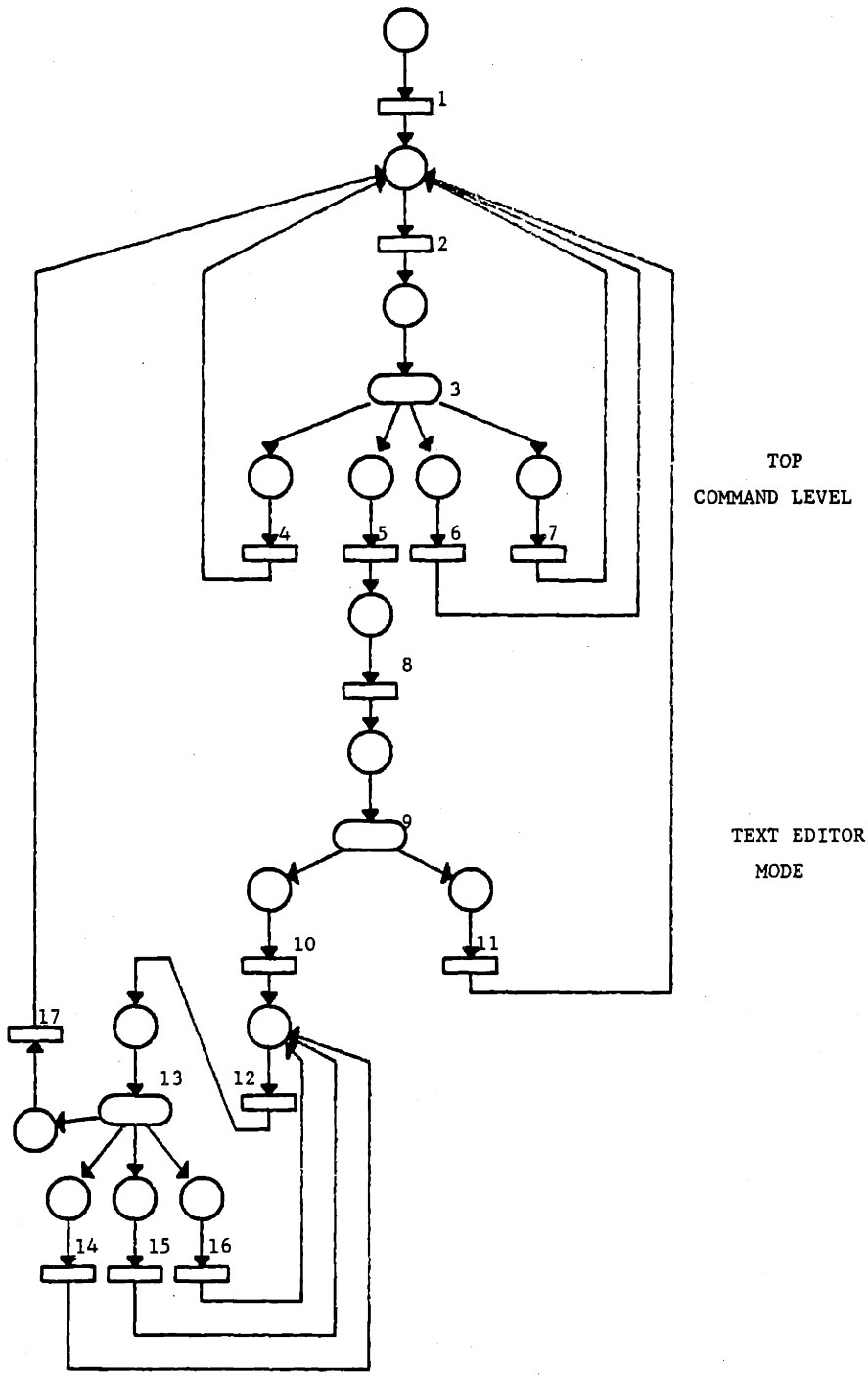


Figure 3.4: Petri Net representation of the Text Editor Modules

Action (Transition No.)	*Interpretation*
1	Start and setup system
2	Request User's Mode Choice
3 (switch)	Evaluate User's Choice and Setup mode
4	Graphics Editor
5	Text Editor
6	Analysis Mode
7	Hardcopy Mode
8	Request file name and search file library
9 (switch)	Evaluate error code from File Manager
10	Open File
11	If file does not exist, return
12	Request user's action
13 (switch)	Evaluate user's action
14	Display information algorithm
15	Modify/create information algorithms
16	Node record editing algorithms
17	Exit/close files/return algorithms

Table 3.2: TEXT EDITOR MODULAR DESCRIPTION

3.7.5 File Organization and Text Data Structure

There are four direct access text files, each used to store information about places, transitions, switches and connectors of a Petri Net structure. Each record of a file may contain information relative to capacity, algorithms/procedures/functions, probabilities, logical or mathematical expressions, comments, existence of subnets or any other information associated with each of the above four elements. As the name of the file indicates, the file organization is of the direct access type as opposed to sequential.

The program uses the direct access mode to retrieve and store text records in the file. A text record consists of data fields, where information is stored, and a record number indicating the record's position in the file. Consequently, a record in the file can be accessed either at random by specifying its record number, or sequentially by going through all previous records. All text records stored in the file are variable length records with a pre-specified upper bound.

Each particular information entry is called a data object and occupies a data field. Each data object is specified by its type code and data. It has the format:

```
tc:data
```

where

```
tc      type code
data    alphanumeric, real, integer or logical data
```

Each data object is preceded by its type code followed by a colon (:), followed by its data. A text record may contain any number of data objects. These data objects must be separated by a semicolon (;). As an example,

```
tc1:dt1;tc2:dt2;tc3:dt3;
```

Type codes may be entered using either upper or lower case lettering. A semicolon after the last data object is optional. A short list of valid type codes is given in Table 3.3.

A complete list is given in the User's Document.

The information stored in the record can be parsed by a command parser. The command parser exploits the structural breakdown of the data object and interprets

Data code-Meaning	*Example*
C - capacity	C:3
P - probability	P:0.7
F - function	F:g2
L - boolean expression	L:'a=b'
T - text	T:comments
ST - Supertransition	ST:4

Table 3.3: VALID DATA CODES

the contents of the fields. The algorithmic implementation of the parser targets on the semicolon delimiter and extracts the alphanumeric characters between the semicolons. It then passes control to the command interpreter. The interpreter checks the parsed information against valid type codes and data and subsequently interprets.

3.7.6 Text Editor Capabilities and Limitations

The capabilities of the Text editor are application oriented. The editor cannot be used for general purpose text editing. Therefore the discussion of the pros and cons of the editor is within the context of its restricted use. This is an inherent limitation of the editor. Most of editor's main features were discussed in section 3.5.3 . An additional feature is that Table 3.3 can be expanded easily, to incorporate more type codes. The command interpreter algorithm has to be adjusted accordingly.

Other limitations:

- maximum record length is 120 characters,
- to change the data part of a data object within a record, one has to erase, change and retype all following data objects,
- insertion or deletion of data objects are to be done according to the rules described in section 3.5.6

3.8 Structural Analysis

3.8.1 Introduction

The objective of this section is to state the purpose and the features of the Structural Analysis Function. An outline of the set of algorithms used for analysis of system architectures is also given, as well as algorithm classification. A short discussion of the

syntactical rules used for consistent and effective application of the algorithms is also provided.

3.8.2 Purpose

It was stated in Chapter 1, that a system engineering study must include modeling, analysis and synthesis. After the Petri Net model has been developed, the performance of the system must be evaluated. If the system specifications are not satisfied, the model is altered, and re-evaluated. This recursive methodology of Petri Net model modification, analysis and performance evaluation, forms the basis of designing the overall system architecture. This is the purpose for Computer Aided Evaluation of System ARchitectures (CEASAR) package. The PN/CAD System is a component of the CEASAR .

The purpose of the Structural Analysis Function is to provide the system designer with a set of descriptive tools for analysis of the Petri Net structure. The combination of the Graphics editor, Text editor, and the Analysis Function, furnishes the system designer a powerful tool.

3.8.3 Features

The Structural Analysis Function offers only two algorithms, which are essential for subsequent evaluation. Both algorithms are based on linear algebra. They are:

- Algorithm for interconnection and incidence matrices construction.
- Algorithm for determination of the minimal supports of S-invariants.

Both algorithms perform the operations in a numerically stable manner provided a number of requirements, discussed in section 3.7, are satisfied. The two algorithms play a dual role:

- they represent the starting point for many analysis, performance evaluation and design procedures,

- they are used for evaluating the final system architecture.

The interconnection matrix is stored in a file with an extension .MTR and can also be displayed in textual form graphically. The textual form of the minimal supports matrix is displayed graphically, while the information flow paths are stored in a file with an extension .SIG, and can be displayed upon request.

Interconnection Matrix Algorithm. In the context of this work, the interconnection matrix can be thought of as a mapping from the graphical Petri Net space to mathematical space. It is an algebraic representation of interrelated nodes, which correspond to the topology of the Petri Net structure. A Petri Net with M places, N transitions, and L switches has an M x (N+L) matrix representation. The nonzero elements of the matrix indicate the connector's number. This number is assigned by the numbering routine of the system. For example, if connector number k, connects the i'th place with the j'th transition, then element (i,j) has the value -k. The construction of the algorithm is simple. The algorithm checks to find to what nodes the two ends of the connector are connected, provided that valid connections exist. Valid connections are:

- place to transition/switch

- transition/switch to place

From the interconnection matrix the incidence matrix is generated and stored in a file with .INC extension. The Incidence Matrix algorithm changes every positive element of the interconnection matrix to $+1$ and every negative one to -1.

Minimal Supports of S-invariants Matrix Algorithm. This is the Alaiwan and Tudic [16] algorithm which generates the minimal supports of S-invariants by operating on the incidence matrix. It is useful in the analysis of a type of Petri Net called event graph. The minimal supports matrix is used to determine the information flow paths of a Petri Net.

3.8.4 Analysis Function Structure

Fig.3.5 shows the Petri Net representation of the Structural Analysis Function. Table 3.4 provides a short description of modular actions.

The modular actions are modeled with transitions 4 through 10. Once in this mode, the designer has a set of commands to choose from. Transitions 8 and 9 may be considered as supernodes and their underlying structure is depicted in Fig. 3.6 and Table 3.5, and in Fig. 3.7 and Table 3.6 respectively.

In the S-invariant mode the program checks the graphical representation, to see if the Petri Net syntax is correct for the algorithm to be applied. If it is not, the system displays a message and exits to the Analysis mode. Otherwise, it determines the minimal supports.

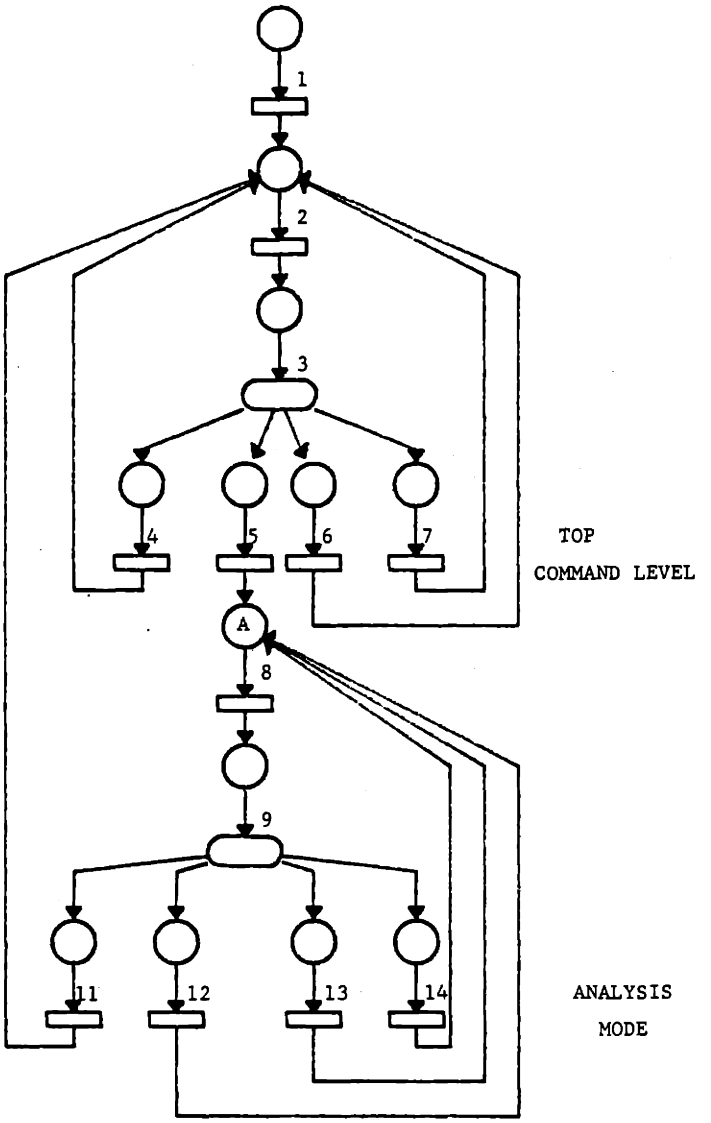


Figure 3.5: Petri Net representation of the Analysis Mode

Action (Transition No.)	*Description*
1	Start and setup system
2	Request command
3 (switch)	Evaluate command code and enter mode
4	Graphics Editor
5	Setup Analysis mode
6	Text Editor
7	Hardcopy Mode
8	Request Command
9 (switch)	Evaluate Command code and take action
10	Exit Mode
11	Incidence matrix mode
12	S-invariant mode
13	Directory mode

Table 3.4: ANALYSIS MODE MODULAR DESCRIPTION

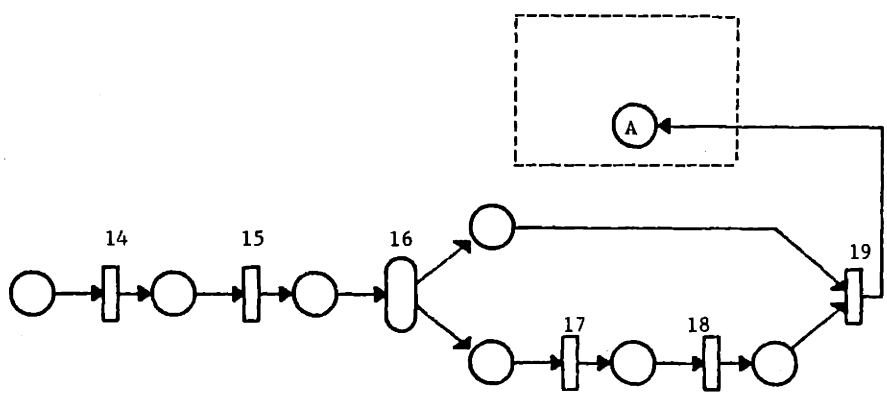


Figure 3.6: Petri Net representation of the Flow Matrix Mode

Action (Transition No.)	*Description*
14	Request File
15	File System
16 (switch)	Evaluate Code
17	Read file, find interconnection matrix
18	Store/display results
19	Return to Analysis Mode

Table 3.5: FLOW MATRIX MODE DESCRIPTION

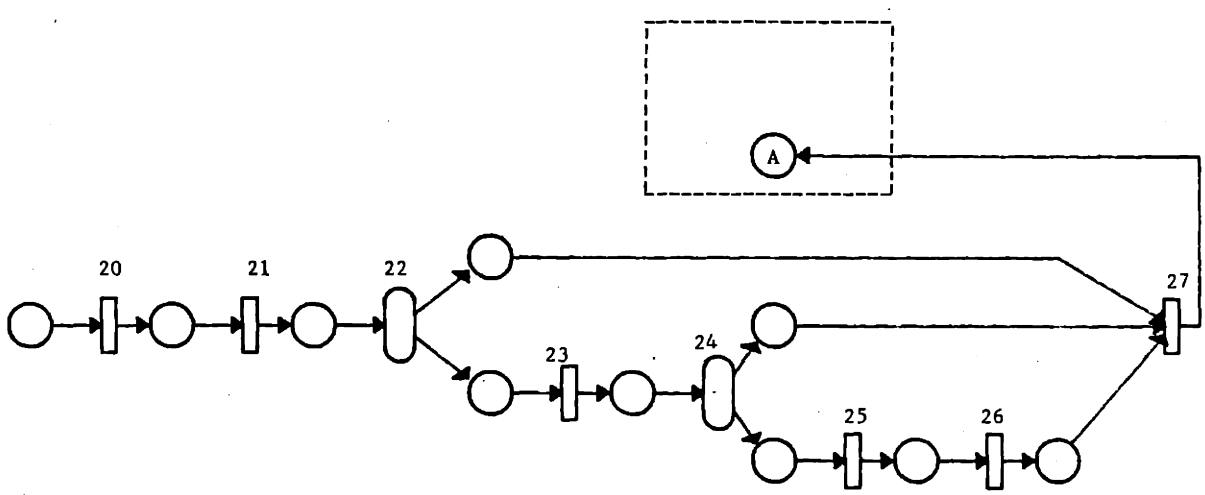


Figure 3.7: Petri Net reprentation of the Minimal Support Mode.

Action (Transition No.	*Description*
20	Request file
21	File system
22 (switch)	Evaluate error code from file manager
23	Read file, check grammar rules
24	Evaluate Grammar code
25	Find minimal supports of S-invariants
26	Store/display results
27	Return to Analysis mode

Table 3.6: MINIMAL SUPPORTS OF S-INVARIANT MODE DESCRIPTION

3.9 Hardcopy Function

3.9.1 Introduction

This section provides an overview of the capabilities and the uses of the Hardcopy Function. Then hardware and software considerations that the system designer should be aware of, before implementing the Function, are discussed.

3.9.2 Overview

The Hardcopy Mode is used to provide the designer with a hardcopy output of the graphic image on a standard output device. It is capable of providing to the application programmer:

- two standard output methods, printing and plotting
- the ability to display the file before it is spooled.

The available commands at this functional level are:

- exit
- show
- print
- plot
- help
- dir(ectory)

These commands are the needed tools in generating a graphic output. The user should connect a printer or a plotter to the IBM PC AT (or compatible) before he turns the power on. Because of the variety of printers and plotters existing in the market, each printer or plotter has its own device driver. A device driver is file a containing the protocol needed to establish a communications channel between the application program and the output device. The user must insert special commands in the CONFIG.SYS file, in order for DOS to load into the memory the file containing the device driver. The designer must also assign a logical name to that device driver

in the AUTOEXEC.BAT file. The logical name is the one the GKS uses to open the desired workstation. The user should be careful in loading the right device driver and selecting the correct configuration mode when he first enters the Software's System Top Command Level.

3.9.3 Structure

Table 3.7 provides a short description of each modular activity while Fig. 3.8 depicts the structure of the Hardcopy Function.

The system checks the file specification of the requested file. It only prints or plots files with the .GRF extension. All other files can be printed when the user is at the DOS command enviroment.

Activity (Transition No.)	*Description*
1	Setup and configuration
2	Request command
3 (switch)	Evaluate command code
4	Setup Hardcopy Mode
5	Request command
6 (switch)	Evaluate Command code
7	Exit Mode
8	Directory Mode
9	Request File
10	Invoke File System
11	Evaluate Code
12	Show Mode
13	Help Mode
14	Print/Plot mode, request file
15	Invoke File System
16	Evaluate File System code
17	Check file specification
18 (switch)	Evaluate file spec code
19	Return to Hardcopy mode
20	Print/Plot

Table 3.7: HARDCOPY MODE MODULAR DESCRIPTION

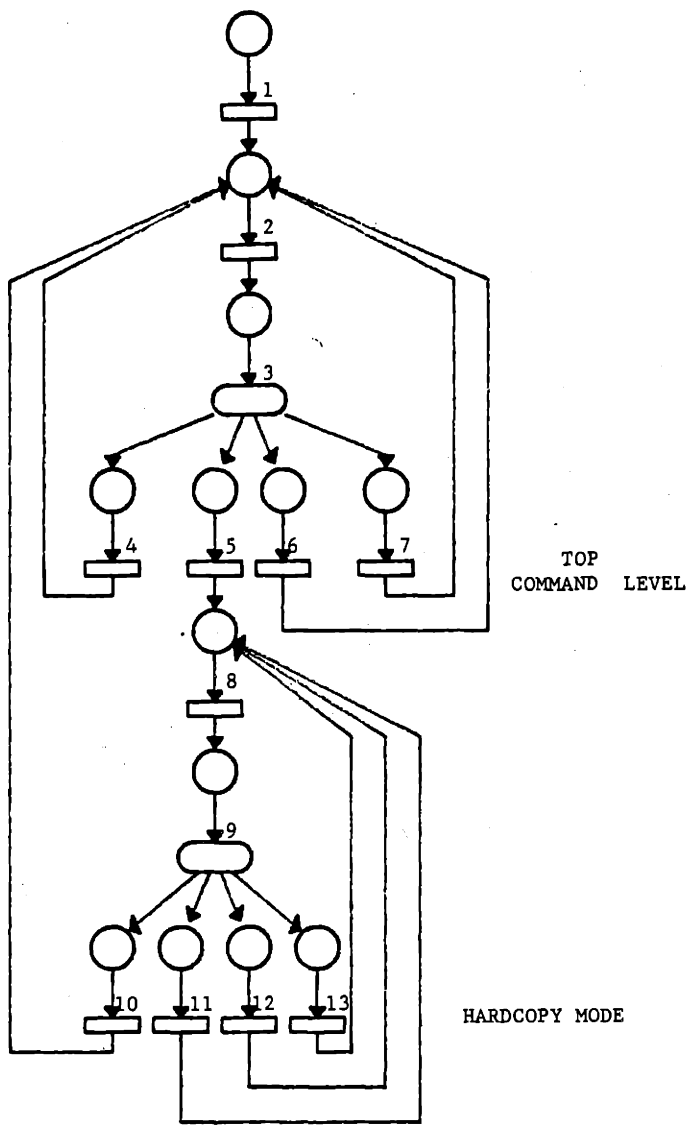


Figure 3.8: Petri Net representation of the Hardcopy Mode.

3.10 File System

3.10.1 Introduction

This section describes the File System Services and the basic concepts behind them. This includes the functional structure, file organization, directory structure, and how the whole concept fits into the application program.

3.10.2 Purpose

The File System is part of the overall Software System. It consists of a set of routines that allow communication paths to be established between the application program and the information files created by the PN/CAD System. The information files are stored on the system disk and contain records with special data structure. The data structure differs for different type of files.

The File System may be considered as a two level hierarchical structured System. At the higher level, there is a set of procedures that provides the File Request Services (FRS). The FRS establishes the communication lines between program and files. It also controls the designer's entered commands and displays diagnostic messages for incompatible requests. At a lower level, the File Manipulation and Managment Services (FMS) are provided. A set of file handling procedures operates on requests from the FMS and manages the file structure.

Because of the number and the special type of files that should be accessed during a graphics or text editing session, there was need to write the Software's own File System Service. A set of diagnostic and auxiliary files are loaded into memory during PN/CAD System startup.

3.10.3 File Request

At the top level of the File System there are a set of routines that handle file requests. For example, upon entering the main graphics editing session, the File's System Request Service prompts the designer to enter the name of the file to be edited. The designer's reply to System's request starts a process. The Request Service receives the response and evaluates it. That is, there is a condition handling routine that is passed a list of arguments from the Graphics editor. The condition handling routine first checks for file request-application mode compatability. That is, while in graphics edit mode, the designer can request only a graphics file, a file with extension .GRF . If there is an incompatability mode fault, control is returned back to the main edit level at the point of call, and a message is displayed on the screen, indicating to the designer to take corrective action.

A second task of condition handling routine is to check, if the file exists in the file directory. For example, while in Hardcopy Mode, the designer may request a file to be printed or plotted. A file existence is indicated by the fact that its file name is included in the directory listing. It has to be mentioned that the condition handling routine does not correct any problems, but acts as a filter indicating if the designer's request is valid or not.

At the lower level of the File Request Service, there are routines evaluating auxiliary designer's requests, such as supernode manipulation, file listing, etc. At the succesful completion of the evaluation phase, a status code is generated. The status code is used to transfer control to the routines that establish the communication lines between the Software System Modules and the files.

3.10.4 File Manipulation

This subsection describes the basic concepts of File Manipulation and Managment procedures and how they fit into the framework of the Software System.

The File manipulation provides five common operations on files:

- create
- delete
- merge
- close
- open

A short descriprion of each operation is given in the following paragraphs.

To create a new file, control is passed from the File Request low level routines to the add new file routine. The routine creates the file name with an appropriate file type. When a new graphics file is created during a graphics editing session, there are also four information files, one for each type of element. In these files, the designer may insert, using the text editor, information attributes of each element. A logical unit number is assigned to the file and control is passed to the Open routine.

The Open routine receives the logical unit number assigned to the file and creates a communication channel. From then on, Read and Write record operations can be performed by the Software System's Functions, such as the graphics editor, text editor or analysis function. The logical unit number serves as a reference to that file. A file status code is received from the File Request Service, indicating whether the file status is new or old.

The Merge operation allows the merging of the contents of two files into one file. This calls for absolute renumbering of all elements of the new Petri Net. This operation takes place when the designer is in the Insert Mode during graphics editing. During such an operation the attribute information files are also merged.

Once the Read or Write record operations have been completed, the system automatically disconnects the communication channels between Software System and files. This operation is performed by the Close All (ClosAll) routine. This procedure offers two options. The default option 1 saves the contents of the file buffer and the file is written on the system disk. Option 2 deletes the contents of the file and erases the file from the system disk. After a file is closed, it cannot be reaccessed unless a call to Open is made.

3.10.5 File Managment

During the Create and Delete operations, the File Managment Service is accessed. The File Manager (FM) updates the file listing and keeps track of the number of files listed. During the Create operation, the FM creates a record with the file name, a file number and date of file creation. The file number, created by the Allocate procedure, is used as a pointer to indicate the entry number in the directory file, where the record is stored.

Similarly, during a Delete record operation on the directory files, the record entry with the specified file name is found and deleted from the directory files. The insertion and deletion of records is facilitated by the link-list operation. The link-list implementation associates with each record entry of the directory file a pointer indicating the position of the entry. Subsequent operations are not performed on the records but on the pointers.

3.10.6 File and Data Record Organization

The data structure of each record in a file is defined by the Function that uses the file. For example, the data structure of a graphics file record is different from the one of an attribute file record. The record format and file contents issues are discussed at the corresponding sections describing each Functional Level.

File organization has to do with the way that records are arranged, such that during Read or Write operations records can be inserted or deleted. There are two types of file organizations selected during file-creation, sequential and direct access ones. In the sequential file the records are organized one after the other. A new record is added at the end of the file. A direct access file consists of a series of memory cells numbered sequentially. Each number serves as a pointer, indicating the relative record position with respect to the first file record. Record insertion or deletion is done by specifying the record number. The maximum record length in a direct access file is 120 and is specified during file creation. The distinct difference between the two organizations is illustrated in the following paragraph, by discussing the record access modes.

The record access mode has to do with how each record is accessed from the file. All files produced by the Structural Analysis Function are sequentially organized. The files produced and accessed during Petri Net text editing are direct access files. The final data structure, generated during Petri Net graphic editing is stored in a sequential type file.

When a graphics file is edited, during a starting Petri Net graphics editing session, the File System generates four direct access files. These files are temporary and are used for record manipulation. Each file is used to store geometrical attributes of the same Petri Net elements. The read operation uses the sequential access mode to read the records sequentially from the old sequential file. The sequential access mode is also used to write the records in each of the temporary direct access files. All entries are arranged in ascending order by record number. An interface routine does the record conversion. During Petri Net modification, the identification number of the picked element is used as the a record number. In that way, the random access by record number mode is used for accessing the records of the direct access file. The last type of access mode makes record accessing fast and easier. The conversion back to a sequential file is done in order to give the designer the ability to display the graphics data structure at the DOS command level.

3.10.7 File Specification

The File Specification attributes furnish to the File System and the designer means of identifying the data stuctures stored in that file. The file specification must be less than 15 characters long, and is composed of two parts: the file name and the file type. It has the following format:

filename.typ

The file name provides a meaningful name to the contents of the file. The file type is three characters long and is used to classify the data structures stored in the file. Table 3.8 lists the file types used:

File Type	*Contents*
.GRF	Numeric file containing the graphics data structure.
.PLC	Text file containing attribute information about places.
.TRN	Text file containing attribute information about transitions.
.SWT	Text file containing attribute information about switches.
.CNT	Text file containing attribute information about connectors.
.MTR	Numeric file containing the interconnection matrix.
.INC	Numeric file containing the incidence matrix.
.SIV	Numeric file containing the minimal supports matrix.
.SIG	Numeric file containing the ids of the elements in an information flow path.
.T*xx* or .P*xx*	Numeric file containing the graphics data structure of the supernode (*xx* is a number).
.TMP	Scratch file.

Table 3.8: FILE TYPES

When a new graphics file is created, the File System creates only the graphics file. The text files are created by invoking the Text editor or when a supernode is defined by the Coarsen command.

When in graphics edit mode, and the designer requests the display of a supernode structure, control is passed to File Request Services. The FRS scans the corresponding text file. Using that supernode's id as a record number, the command parser tries to target **SP** for superplace, or **ST** for supertransition. If it finds it, it constructs the file and then scans the directory to find the file. For example, a Petri Net structure is stored in the file **ORGZ.GRF** . If transition number 15 is a supertransition, then its data structure is stored in a file with a name **ORGZ.T15**. That is the original name concatenated with the character string 'T15', where 'T' stands for transition and '15' is

the transition's identification number. For supernodes the file type has the form shown in Table 3.8. The maximum allowable number of supernodes is 20.

3.10.8 Directories

The PN/CAD Software has its own directory system. All directories are managed by the File Manager. A directory is a file which contains a list of files created by the Software System. The System has four directories, where each directory file is used to store files with the same file type. Table 3.9 lists the directory names.

All directory files are of the sequential type, but during program execution the directory file contents are transferred to a direct access file for better and easier file accessing. The designer should be careful not to delete the directory files.

3.10.9 Conclusions

In this section a detailed description of all the services provided by the File System has been presented. At the top level of the File System reside the File Request Services. At a lower level, the File Manipulation and Manager Services handle all the user's requests and manage the file organization and record structure.

Directory Name	*Contents*
dir.grf	All .GRF file types
dir.txt	All text files
dir.sup	All supernode files
dir.scr	All other files

Table 3.9: DIRECTORY NAMES

3.11 Grammar-Rule Checker

The grammar-rule checker is a set of algorithms used to check the syntax of the generated architectures and the constraints that have to be satisfied for the implementation of the available analytical tools. This section describes the rules for:

- Entering commands
- Entering options
- Entering file specifications
- Graphical construction
- Structural Analysis

3.11.1 Rules for Entering Commands and Options

In an interactive editing session all commands are displayed in the Command Menu Area. Certain commands may have their own command mode menu. In that case, the command interpreter may request from the system designer to enter an option. To enter a particular command mode, the designer has to move the cursor on top of the command and press the Enter key on the keyboard. If the command is picked it will be highlighted. The procedure is similar in picking graphical symbols from the list provided when in graphics edit mode. Certain commands provide a set of options. In such a case, a message is displayed in the Message Area, listing the options, their correct input syntax and requesting from the designer to take action. All command options must be entered in lower case.

In text edit mode all requests should be entered in lower case lettering and the data structure of the information entries should be done according to specifications set and described in section 3.7.5 .

3.11.2 Rules for Entering File Specification

The format of file specifications was described in section 3.10.7. In an interactive editing session the designer is asked to enter the proper file name. For example, during a graphics editing session all files edited must have .GRF, Txx or Pxx as a file type (xx is the supernode's identification. Similarly, in order for the interconnection matrix to be generated the requested file must have .GRF as file type. To generate the minimal support matrix the input file must have .MTR as a file type. In a text editing session all input files should have .TXT as a file extension.

3.11.3 Rules for Graphical Construction

During graphical construction of a Petri Net, the following rules must be observed:

- Graphics elements should not overlap.
- Graphics elements should be placed within bounds of the screen.
- No connectors should connect two nodes of the same type.
- Existence of self loops should be noted.
- Space must be available before inserting a Petri Net.
- Each supernode must be a subnet, all input and output nodes should be specified.

3.11.4 Rules for Structural Analysis

There are only two rules that should be observed during the analysis of the minimal support of S-invariants:

- The Petri Net graph should be an event graph.
- If the Petri Net has switches, a condition must be implemented.

3.12 Software-User Graphics Interface

All the design programs and files are stored in the disk of the system. Upon user's request, each information should be readily available and easily accessible. At the command level, a list of commands is provided to facilitate information interchange. Hence, requirements arise for good user interface and efficient programming.

The graphical user-software interface employs a keyboard to provide an easy access to commands. The keyboard position is indicated by a cross hair on the monitor screen. To pick a command, the cross hair is placed at the correct spot, on top of the command. Pressing the Enter key causes the command to be highlighted. If any action is required from the user, a guiding comment or a query sequence is displayed in the Message Area. The user cannot proceed further without heeding the instruction.

3.13 Conclusions

In this section a detailed description of the ideas involved in the development, implementation and interconnection of all the modules of the PN/CAD System has been presented. First, the Top Command level is described. The Functional Level is the heart of the integrated system. There are four major functions available in the current version, i) the Graphics Editor, ii) the Text Editor, iii) the Analysis Mode and, iv) the Hardcopy Mode. The file exchange service is handled by the File System. The Grammar-Rule Module checks for syntactical errors in the generation of organizational architectures and enforces the rules for structural analysis.

The PN/CAD System is a powerful interactive program for the generation of the Petri Nets of arbitrary organizational architectures. The PN/CAD System can also be used to create or delete supernodes, determine specific structural properties and provide means for storing information attributes for each element of the Petri Net. Two examples, demostrating the capabilities of the PN/CAD System, are presented in the next chapter.

Chapter 4

APPLICATIONS

4.1 Introduction

In this chapter, two examples are presented to illustrate the ideas described in chapter 3. The examples are used to demonstrate both the graphical and analytical capabilities of the PN/CAD System. The designer may

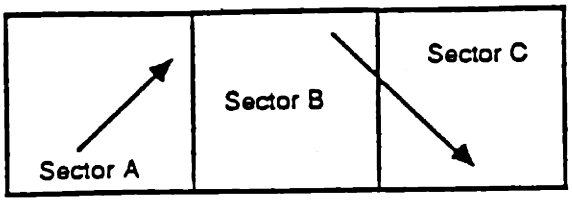
- have already developed the Petri Net representation of an organization on paper, and use the interactive PN/CAD System for further graphical and analytical processing, or initiate modifications on the basis of organizational performance;
- develop interactively the Petri Net model of arbitrary organizational architectures.

In the following sections, the first method is used in order to illustrate the capabilities of the PN/CAD System.

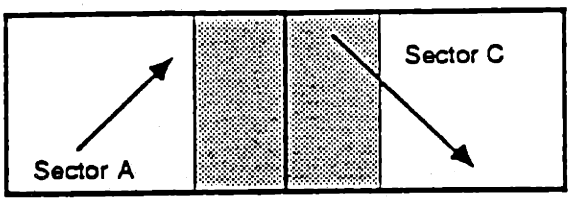
4.2 Examples

Two Petri Nets, each representing three decision makers of a decision making organization, one with hierarchical structure and the other with parallel structure, will be used to illustrate the graphical and analysis capabilities of the software package. Specifically the method used for the graphics construction is stated briefly, and then the construction of the interconnection and incidence matrices is shown. The graphic display of all the information flow paths for the different decision strategies is also generated.

Both decision-making organizations implement an air defense task [11]. In the parallel organization the airspace has been divided into three sectors, with each decisionmaker assigned to one sector, see Fig. 4.1(a) .Each decision-maker can observe and engage threats in his sector. However, threats may move between sectors, therefore there is a need for communications and information sharing between decision-makers in adjacent sections. In the hierarchical organization, the airspace is divided into two sectors and each sector is assigned to one decision-maker, see Fig. 4.1(b).



(a) Three Sectors (Parallel)



(b) Two Sectors (Hierarchical)

Figure 4.1: Division of Airspace

A center region is defined that straddles the two sectors and a supervisor is introduced, which does not observe the airspace directly, but receives information about threats in the central region from the other two decision-makers. He then processes the data and allocates the threats in the central region to either one of the decision-makers, depending on the trajectory of the threat.

4.3 A Sample PN/CAD Session

This section illustrates the use of selected PN/CAD commands through a sample PN/CAD session. In the first session, the hierarchical organization, Fig. 4.2, is graphically constructed in a step by step manner. Then, iff possible, all supernodes are identified such that the overall structure is simplified. Next, the Analysis Function is invoked to determine the interconnection matrix, generate and display graphically the information flow paths of the organization. Finally all graphs are plotted (or printed) using the Hardcopy Function. In the following section, the parallel organization, Fig. 4.3, is constructed.

4.3.1 Hierarchical Organization

When the session is started, the PN/CAD Choice Menu appears on the screen. The System requests from the user to choose from a list of devices. The System evaluates the user's choice and it will activate the devices, Fig. 4.4. The PN/CAD opening screen is shown in Fig. 4.5. The next screen is the Top Command Level screen, see Fig. 4.6. To initiate the graphical construction of any organizational architecture, the user has to invoke the Graphics Editor. To invoke the Graphics Editor the cross hair is placed on top of the GRAPHICS EDITOR command. By pressing the *Enter* key the PN/CAD System highlights the text. If the user invokes succesfully the command the screen is updated and the user is in Level One Mode of the Graphics Editor, see Fig. 4.7 . Invoking the EDIT FILE command the user is placed on Level Two of the Graphics Editor. At this Level one can create a Petri Net representation, initiate modifications and save or delete in a file the work done in the editing session, see Fig. 4.8.

The hierarchical architecture is created using two generic architectures. The two architectures are used as building blocks to compose a complex one. The first generic architecture, whose structure is stored in the 'SNET1.GRF' file, is the Petri Net representation of the four stage DM. The Responce Selection (RS) stage is modeled by a 2-decision switch, see Fig. 4.9. The second generic architecture, whose structure is

stored in the file 'SNET2.GRF', is the Petri Net representation of a two stage DM (SA and RS), see Fig. 4.10 . The file 'HIER.GRF' is created to store the hierarchical structure. The two generic substructures, stored in files 'SNET1.GRF' and 'SNET.GRF' respectively, are inserted into 'HIER.GRF' file by invoking the INSERT command, see Fig. 4.11. The user completes the representation by selecting the Element Menu commands, see Fig 4.12. The major ideas illustrated in this example are the :

- Identification of generic architectures.
- Graphic development and storage of generic architectures.
- Combination of generic architectures to form a complex organizational structures

At this point, the first stage of graphical construction has been completed. The user has the option to exit the editor and proceed to the analytical description of the Petri Net or try to simplify the structure. Supernodes were discussed in section 2.4. The RS stage of each of the three DMs is a switch with two switch settings, and may be modeled as a supernode. To perform the operation the user must invoke the COARSEN command. In COARSEN mode the user has to identify the input nodes, the output nodes, and the boundaries of the candidate supernode, see Fig. 4.12. The simplified Petri Net structure, shown in Fig. 4.13, is considered to be the first hierarchical level, while the refined one is considered to be the second one. The user may save his work, exit from the Graphics Editor and return to Top Command Level.

The next step is to invoke the Analysis Function and generate the analytical description of the architecture. To create the interconnection matrix, the user must invoke the MATRIX command and enter the file name 'HIER.GRF' . The Analysis Mode opening screen is shown in Fig. 4.14. The interconnection matrix is stored in a file 'HIER.MTR', see Fig. 4.15. The incidence matrix is also generated and is stored in the file 'HIER.INC'; it can be viewed when the user is in the DOS enviroment. The incidence matrix serves as input to a number of algorithms for performance evaluation. To create the minimal support matrix of S-invariants, the user must invoke the S_INV command and enter the file name 'HIER.MTR'. The Petri Net representation of the

hierarchical organization, Fig. 4.13, is an event graph, and the Alaiwan and Toudic [16] algorithm can be used to find the minimal supports of S-invariants for Petri Nets without switches. Thus in order to be applied to a Net containing switches, the switch and the corresponding transitions have to be combined into a supertransition. The minimal support matrix is displayed on the screen and it is stored in the file 'HIER.SIV'. The numeric file containing the data structure of all the information flow paths is stored in the file 'HIER.SIG', and can be displayed on the screen upon user's request.

The information flow paths thus identified contain the supertransitions that have replaced the selection switches and their transitions. In order to identify the paths that result when the switches select specific transitions, the following procedure is employed [17]:

If a supertransition contains K settings and this transition is on the path, the path has K different instantiations with probabilities $P_1, P_2, \cdots, P_k$ assigned to each distinct path, corresponding to the relative frequency of use of transitions $T_1, T_2, \cdots, T_k$ where

$$P_1 + P_2 + \cdots + P_k = 1 \tag{4.1}$$

If an information flow path has two supertransitions that contain switches, the first with K settings and the second with L settings, that information flow path has $N = K \times L$ different instantiations each one implemented with probability $P_{ij} = P_i \times P_j$ where,

P_i is the probability of selection by the first switch and,

P_j is the probability of selection by the second switch.

In this example, some flow paths with a single supertransition have two different instantiations, while some information flow paths with two supenodes have four different instantiations.

Finally, all graphs may be plotted or printed using the Hardcopy Function. Fig.

4.16 shows the Hardcopy screen after the PLOT command has been invoked. All the information flow paths are shown in Fig. 4.17-26.

4.3.2 Parallel Organization

The Petri Net representation of the parallel organization is constructed in a similar fashion. The designer invokes the Graphics Editor from the Top Command Level Enviroment. To design graphically the parallel organizational structure, he uses the same generic architectures he used in the construction of the hierarchical one. Let 'PARL.GRF' be the name of the file containing the structure of the parallel organization, see Fig. 4.27 . The Petri Net shown in Fig. 4.3 has decision switches. It has three switches with two switch settings each. Therefore there exist eight different combinations that instantiate distinct Petri Nets. For each such Petri Net the incidence matrix and information flow paths need to be identified.

In the hierarchical case, the decision switch and its possible paths were modeled by a supernode. In the parallel case, a switch setting is selected and a particular decision strategy is implemented. To achieve this the designer must invoke the SWITCH command. When the SWITCH command is implemented, the System searches for the attributes of each switch setting. By default, it searches for a connector attribute file named 'PARL.CNT'. If the file does not exist, the PN/CAD System requests from the user to activate a switch path, see Fig. 4.28. The user must reply to the System's request by typing his choice for each switch. The Petri Nets for all possible combinations of switch settings are shown in Fig. 4.29-4.36. The respective graphical structure is stored in a separate file. The name of the file is furnished by the user.

If the file does exist, the System parses the information stored in the record corresponding to that node (connector 8). Specifically, it searches to find the probability assigned to the specific connector. If the probability of instantiating path A (associated with output connector 8) is greater than the probability of instantiating path B (associated with connector 9), then path A is selected.

To create the file 'PARL.CNT' the user must exit the Graphics Editor and invoke

the Text Editor from Top Command Enviroment. The graphics screen is switched off and the user is placed in the Text Editor mode. The Text Editor's Opening Screen and the Editor's Choice Menu are shown in Fig. 4.37. The user enters the connector attributes and exits the Text Editor. The information flow paths for the case shown in Fig. 4.29 are depicted in Fig. 4.38-4.44.

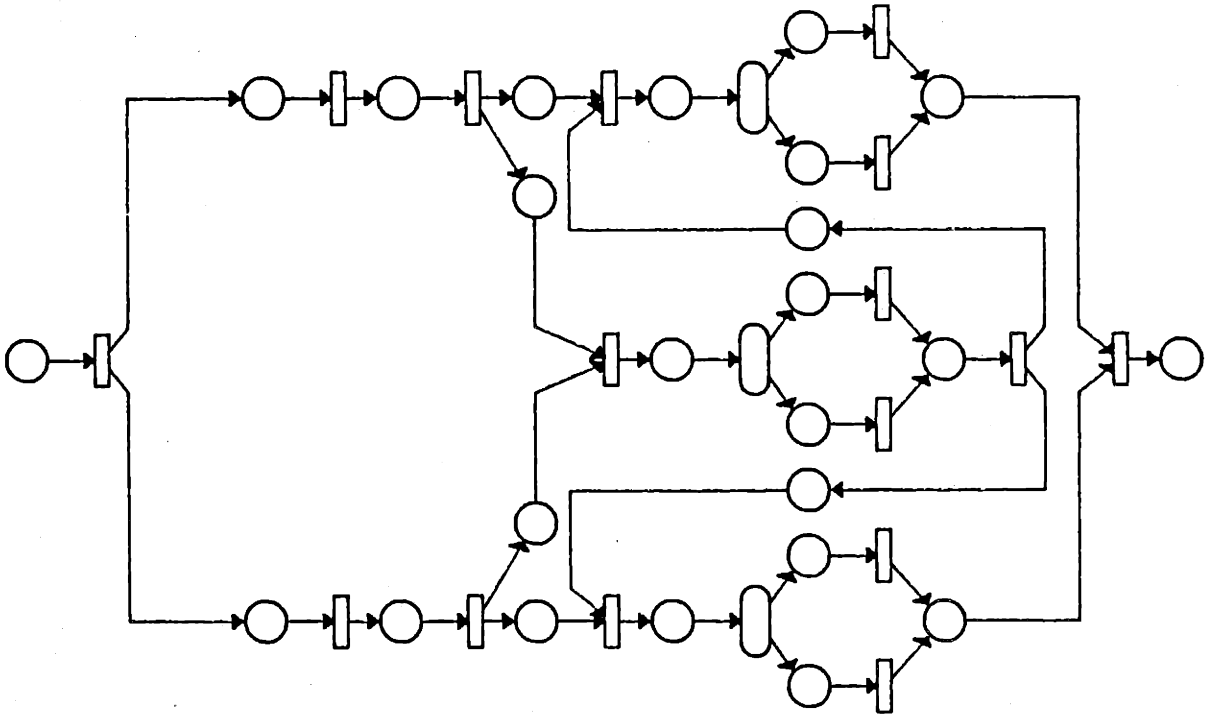


Figure 4.2: Hierarchical Organization.

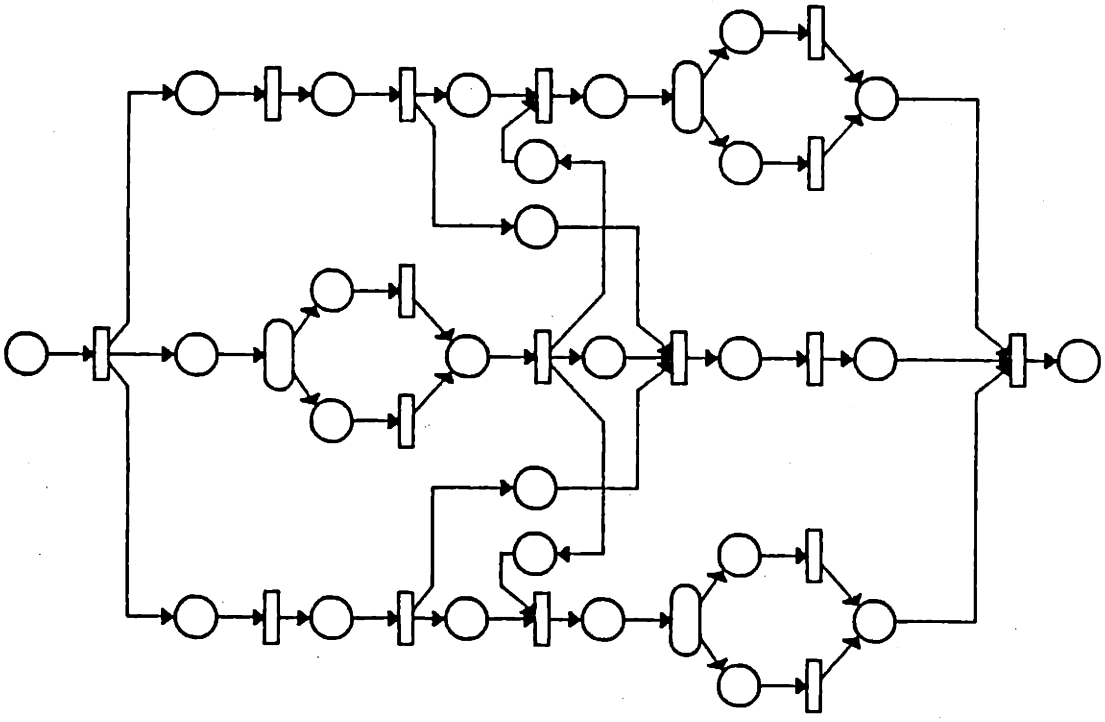


Figure 4.3: Parallel Organization.

```
*** PETRI NET EDITOR CONFIGURATION MENU ***

- Please choose configuration -

1. DISPLAY
2. DISPLAY AND PRINTER
3. DISPLAY AND PLOTTER
4. DISPLAY, PRINTER AND PLOTTER
5. EXIT

ENTER CHOICE (1, 2, 3, 4, 5):
```

Figure 4.4: PN/CAD Choice Menu.

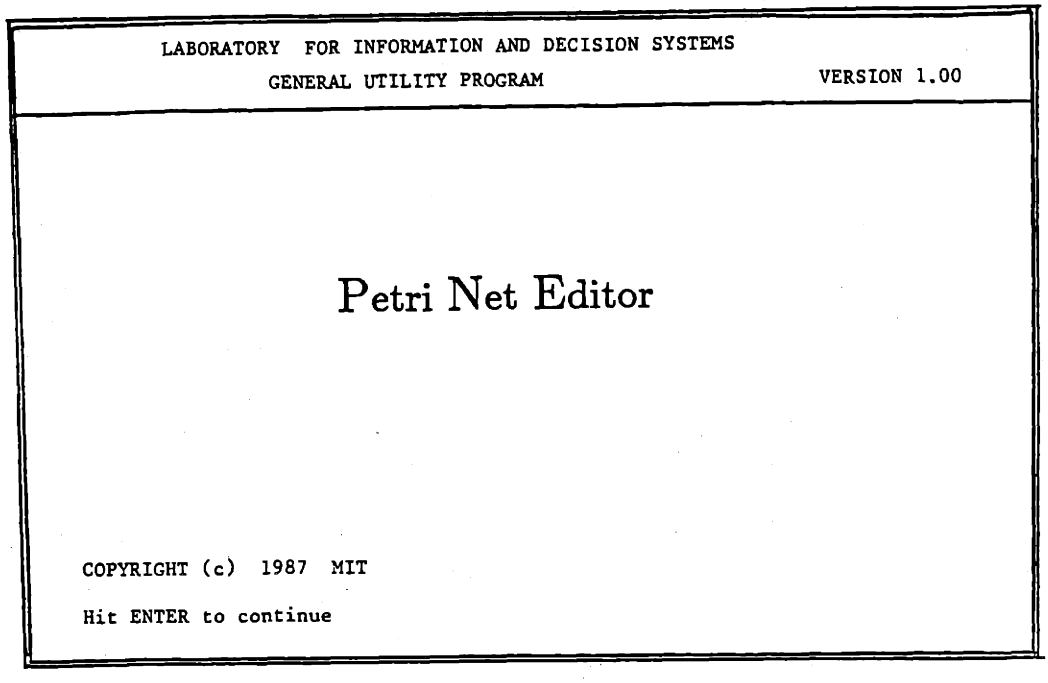


Figure 4.5: PN/CAD Opening Screen.

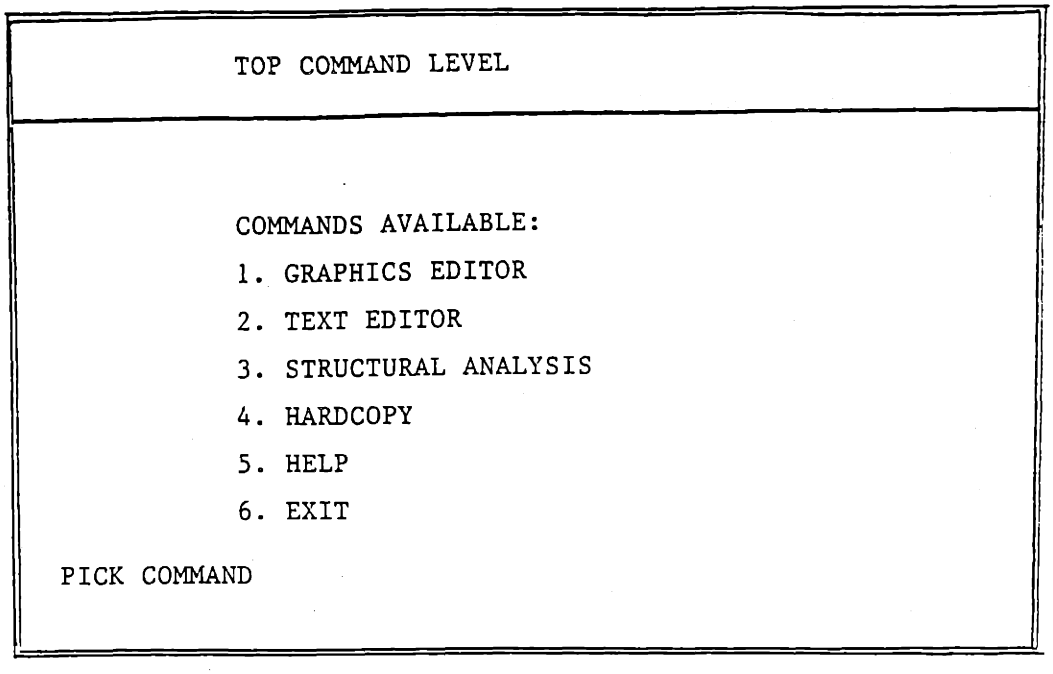


Figure 4.6: Top Command Level Opening Screen

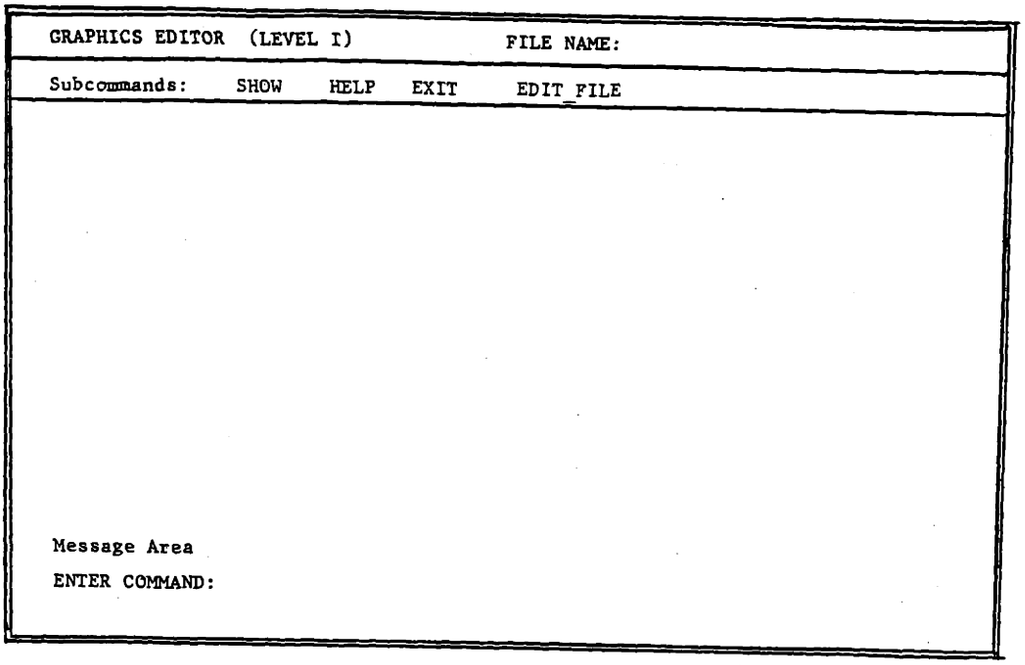


Figure 4.7: Graphics Editor Level One Screen.

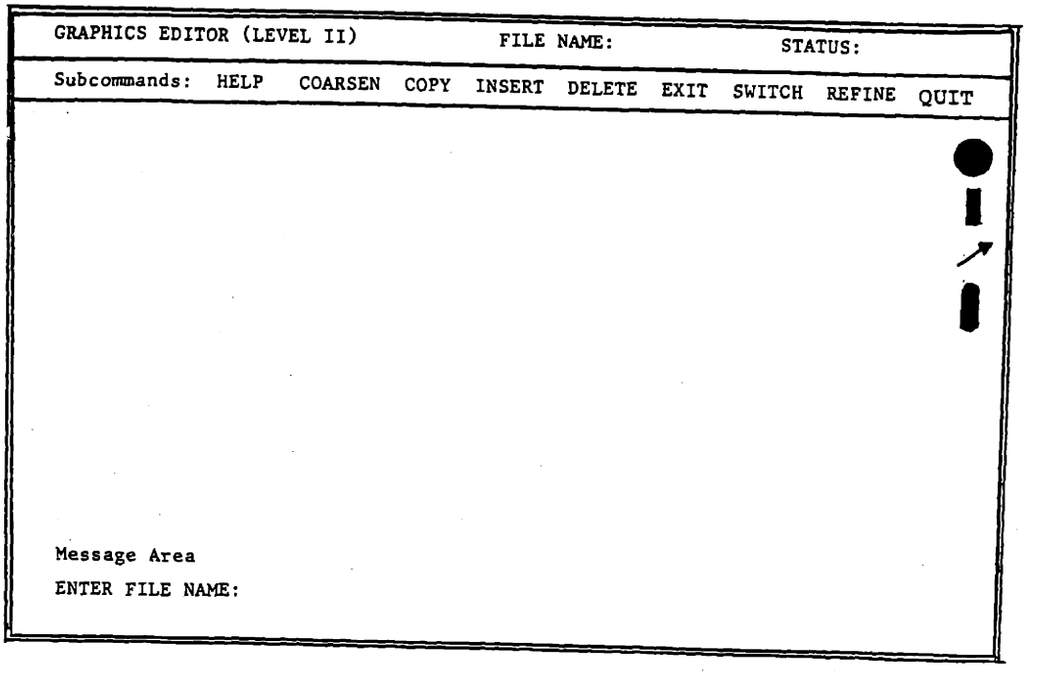


Figure 4.8: Graphics Editor Level Two Screen.

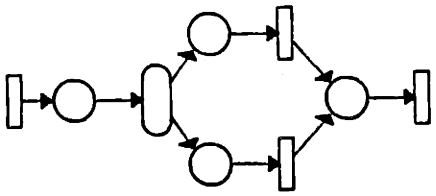


Figure 4.9: First Generic Architecture.

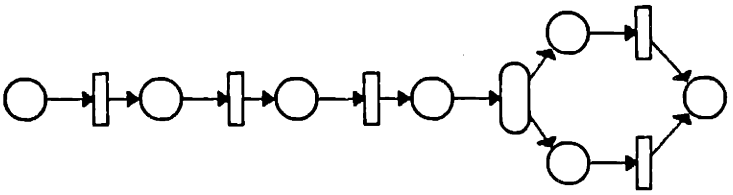


Figure 4.10: Second Generic Architecture.

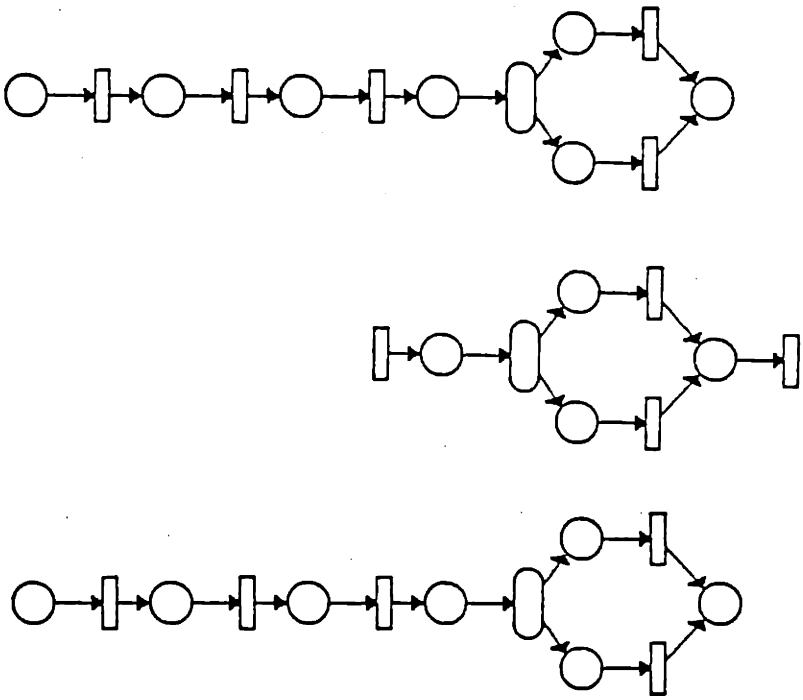


Figure 4.11: Contents of file 'HIER.GRF'.

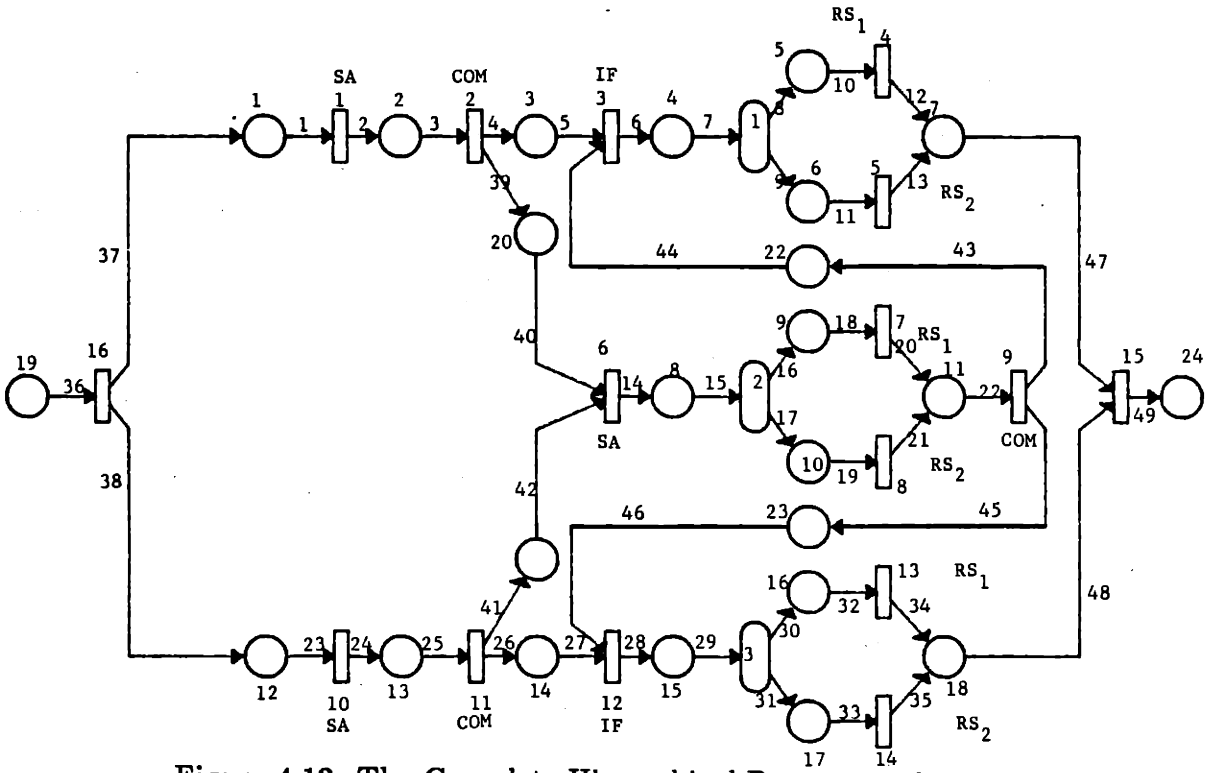


Figure 4.12: The Complete Hierarchical Representation.

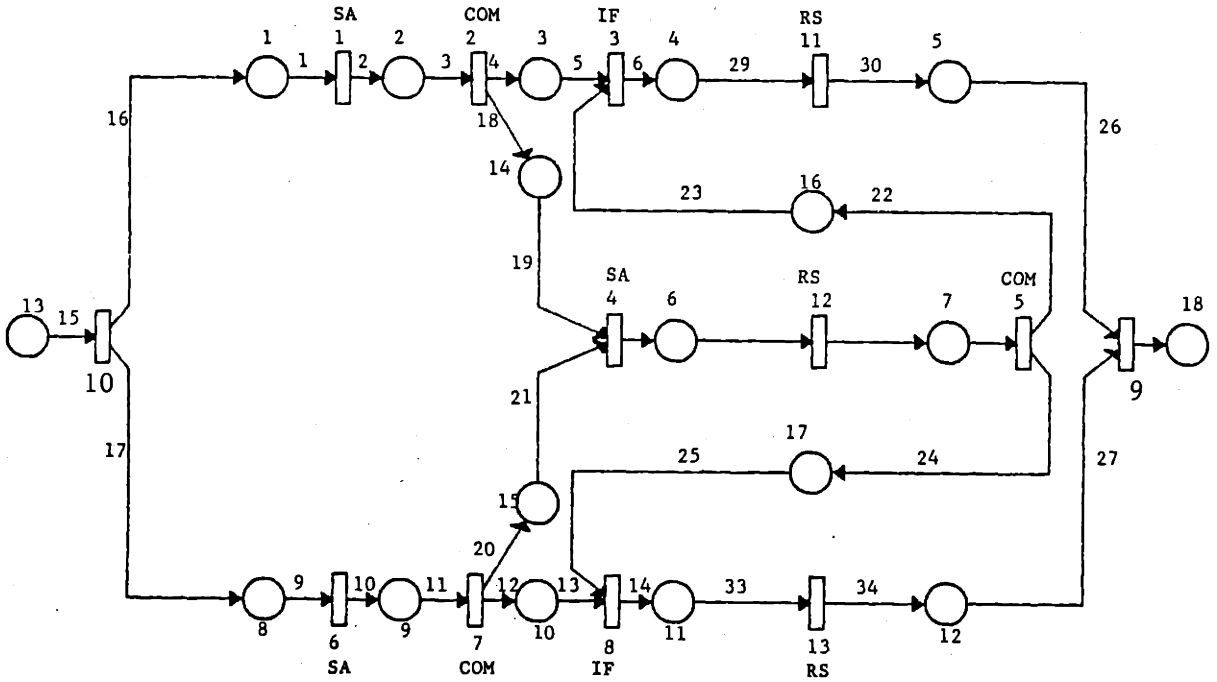


Figure 4.13: The Simplified Hierarchical Representation.

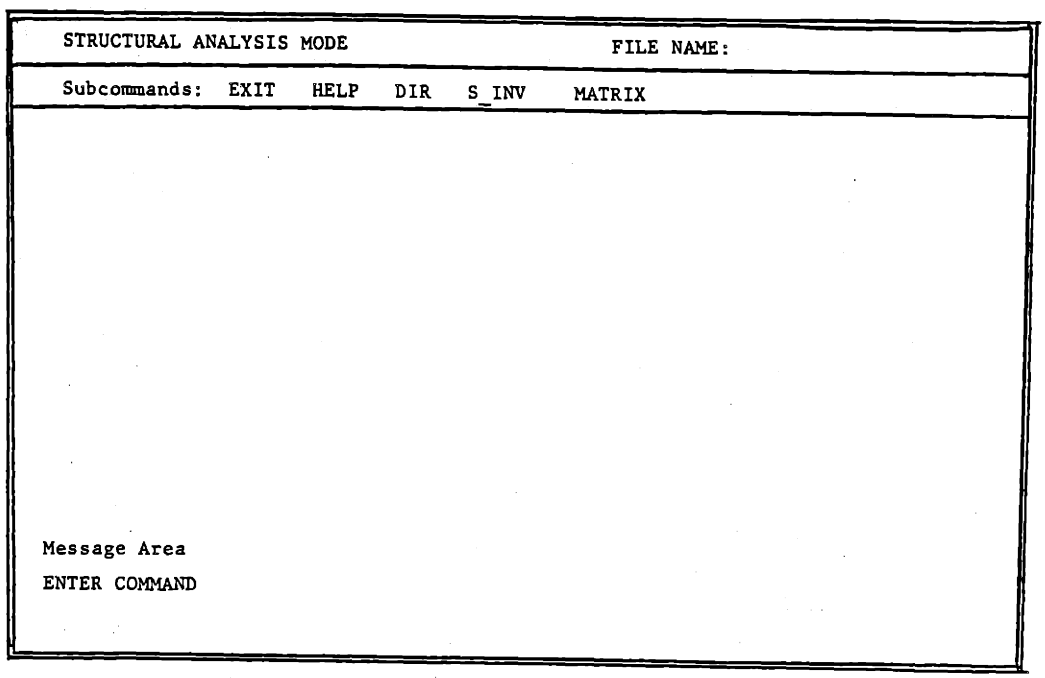


Figure 4.14: Analysis Mode Screen.

-1	0	0	0	0	0	0	0	0	16	0	0	0
2	-3	0	0	0	0	0	0	0	0	0	0	0
0	4	-5	0	0	0	0	0	0	0	0	0	0
0	0	6	0	0	0	0	0	0	0	-29	0	0
0	0	0	0	0	0	0	0	-26	0	30	0	0
0	0	0	7	0	0	0	0	0	0	0	-31	0
0	0	0	0	-8	0	0	0	0	0	0	32	0
0	0	0	0	0	-9	0	0	0	17	0	0	0
0	0	0	0	0	10	-11	0	0	0	0	0	0
0	0	0	0	0	0	12	-13	0	0	0	0	0
0	0	0	0	0	0	0	14	0	0	0	0	-33
0	0	0	0	0	0	0	0	-27	0	0	0	34
0	0	0	0	0	0	0	0	0	-15	0	0	0
0	18	0	-19	0	0	0	0	0	0	0	0	0
0	0	0	-21	0	0	20	0	0	0	0	0	0
0	0	-23	0	22	0	0	0	0	0	0	0	0
0	0	0	0	24	0	0	-25	0	0	0	0	0
0	0	0	0	0	0	0	0	28	0	0	0	0

Figure 4.15: Interconnection Matrix of the Simplified Hierarchical Model.

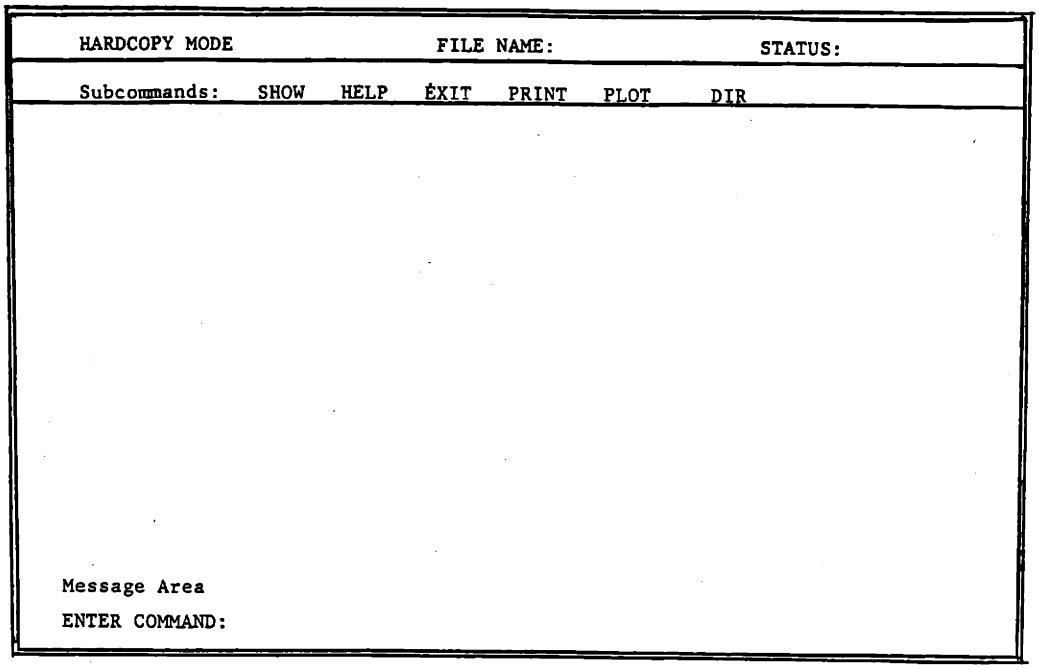


Figure 4.16: Hardcopy Mode Screen.

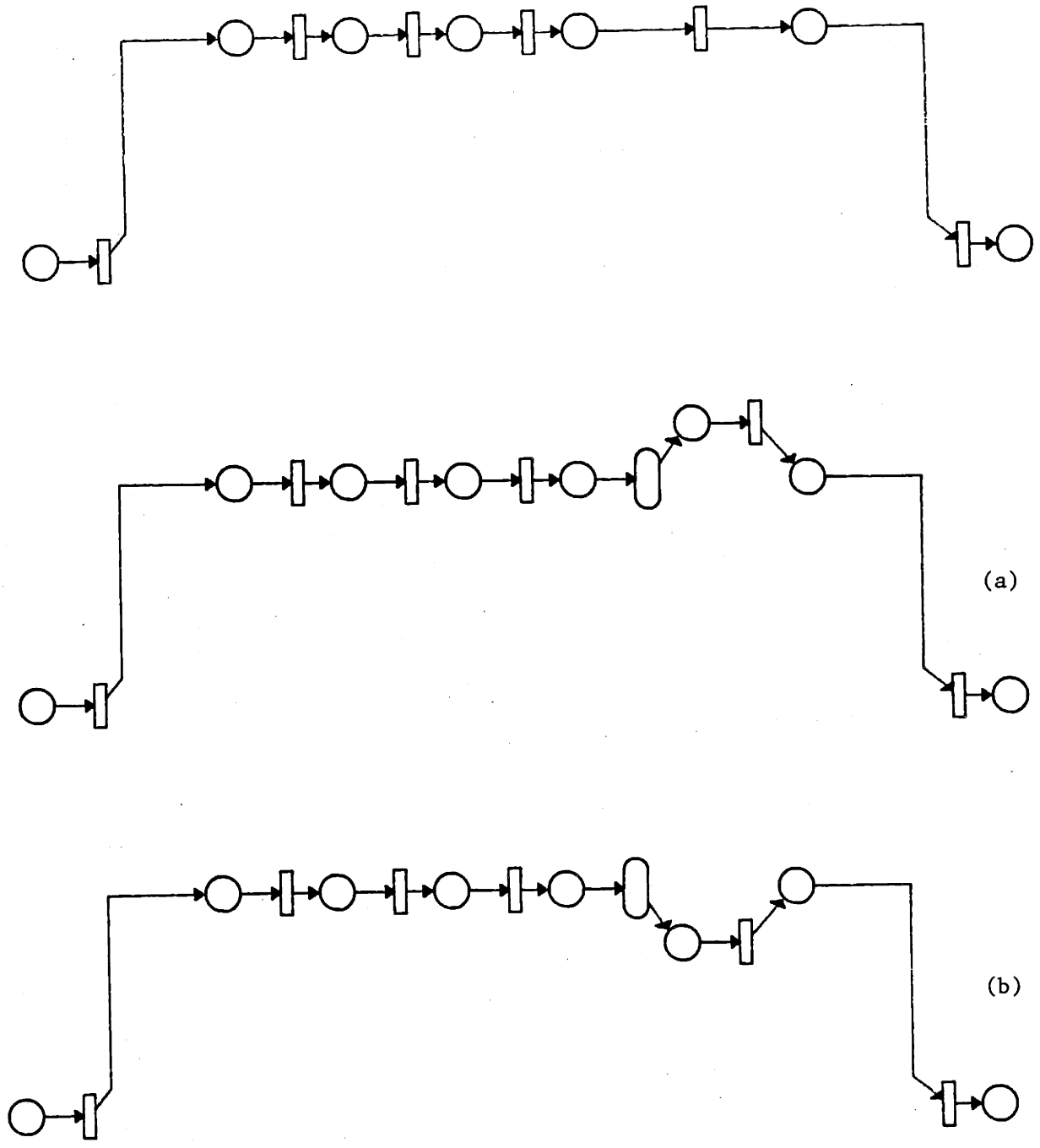


Figure 4.17: Hierarchical Model: Flow Path No. 1 with instantiations (a), (b)

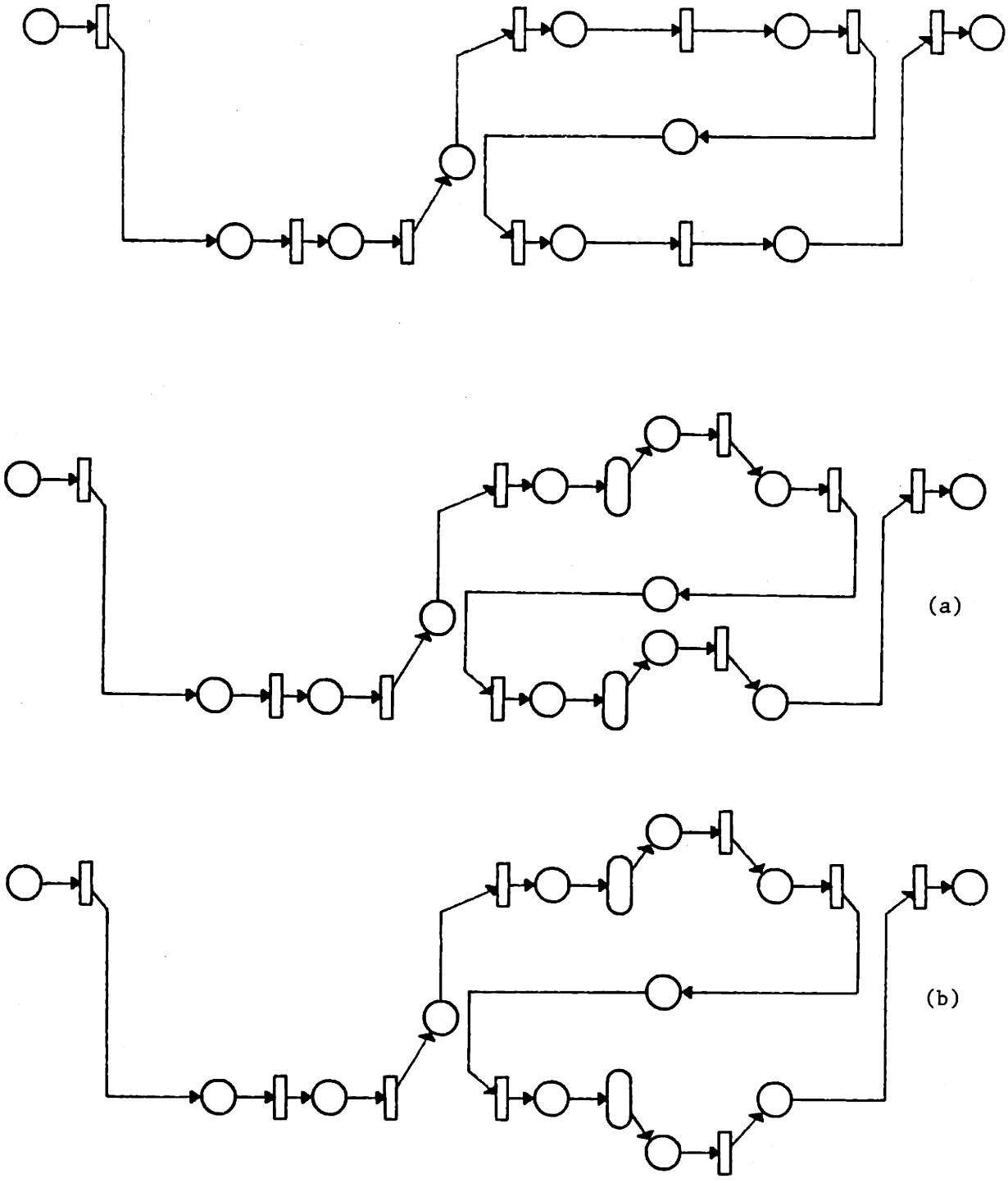


Figure 4.18: Hierarchical Model: Flow Path No. 2 with instantiations (a), (b)

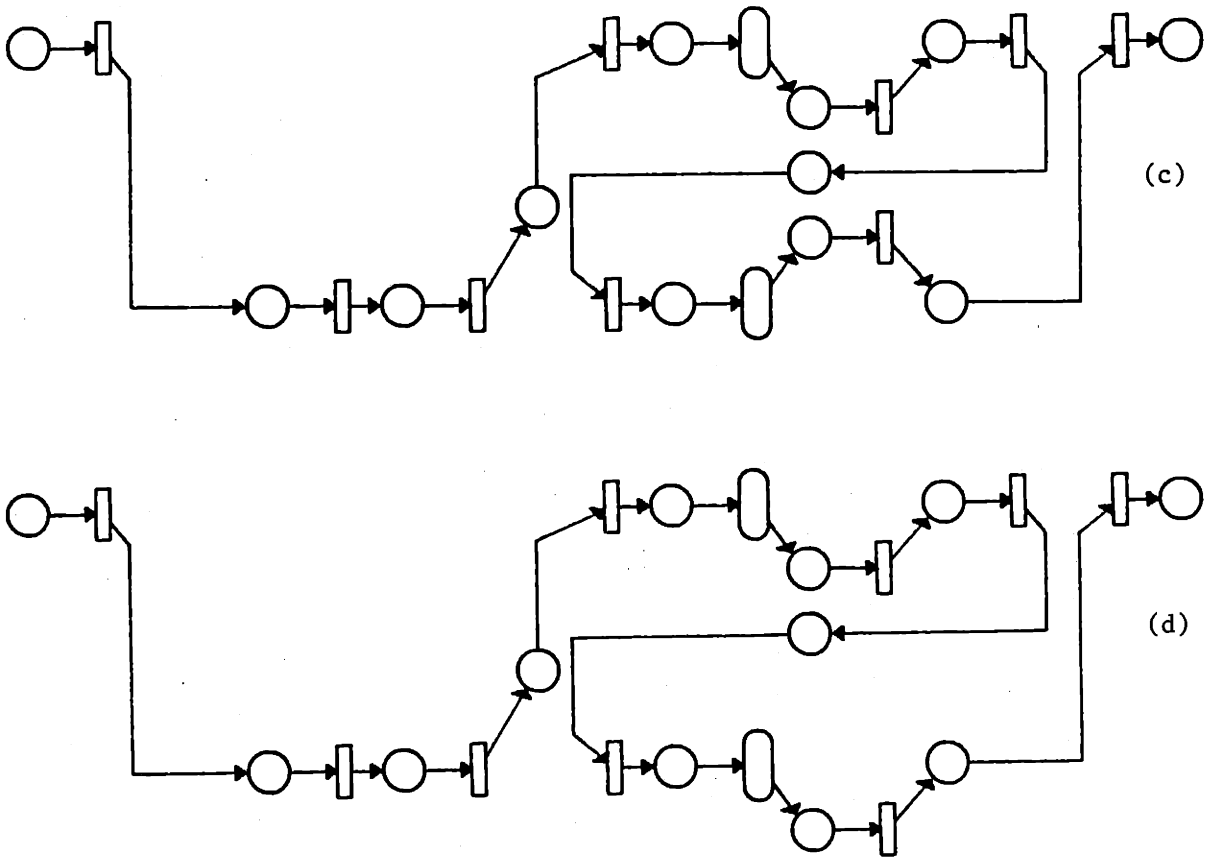


Figure 4.19: Hierarchical Model: Flow Path No. 2 with instantiations (c), (d)

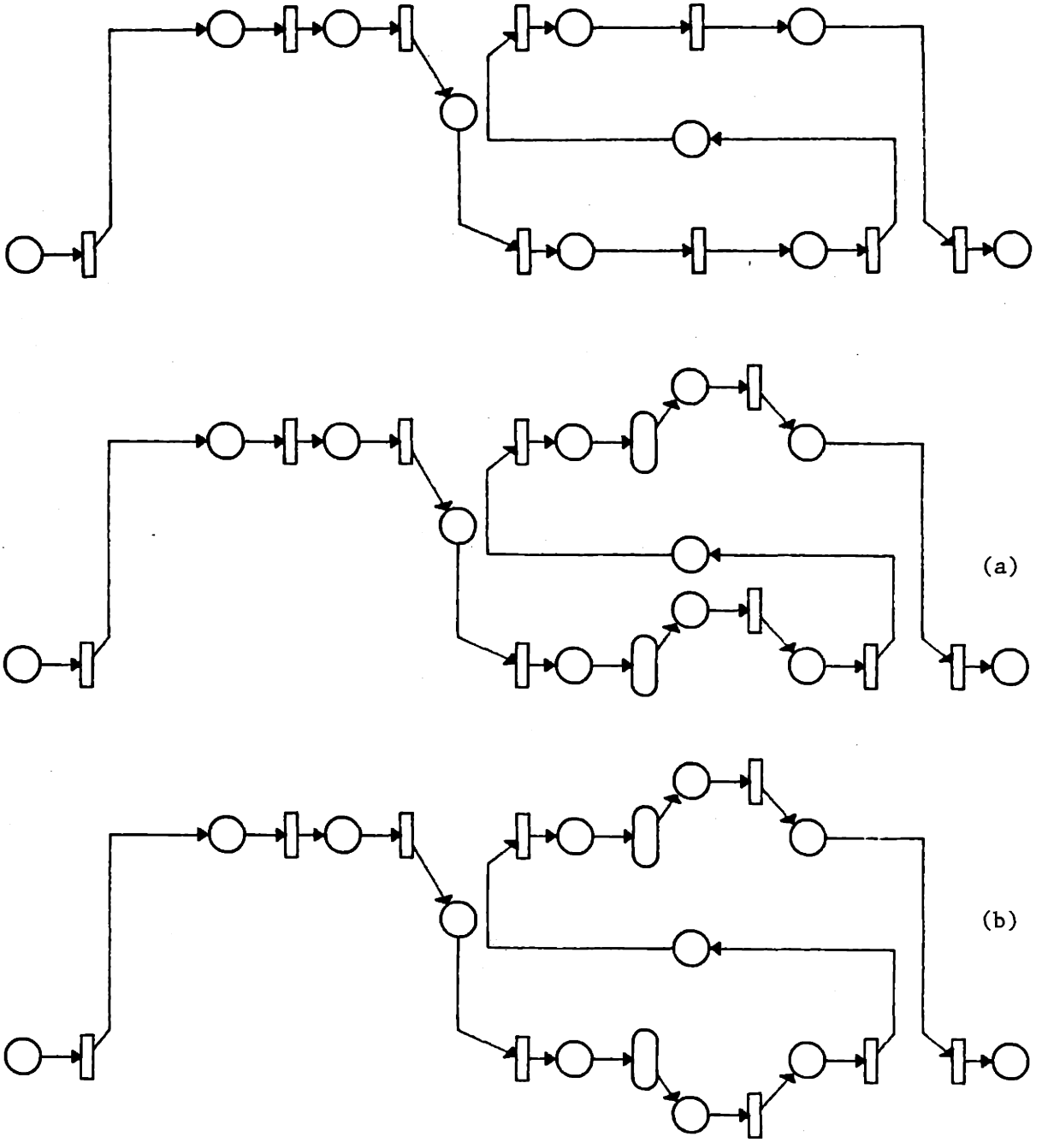


Figure 4.20: Hierarchical Model: Flow Path No. 3 with instantiations (a), (b)

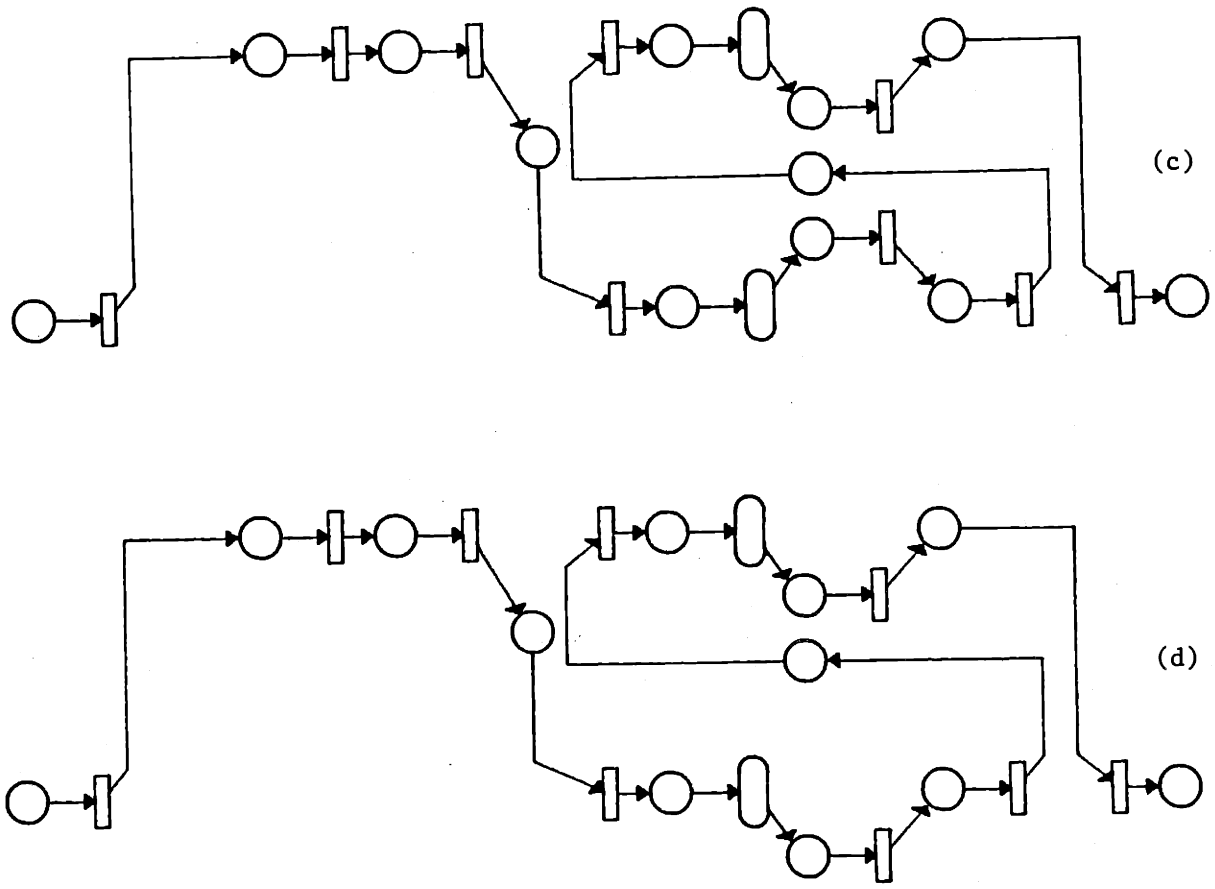


Figure 4.21: Hierarchical Model: Flow Path No. 3 with instantiations (c), (d)

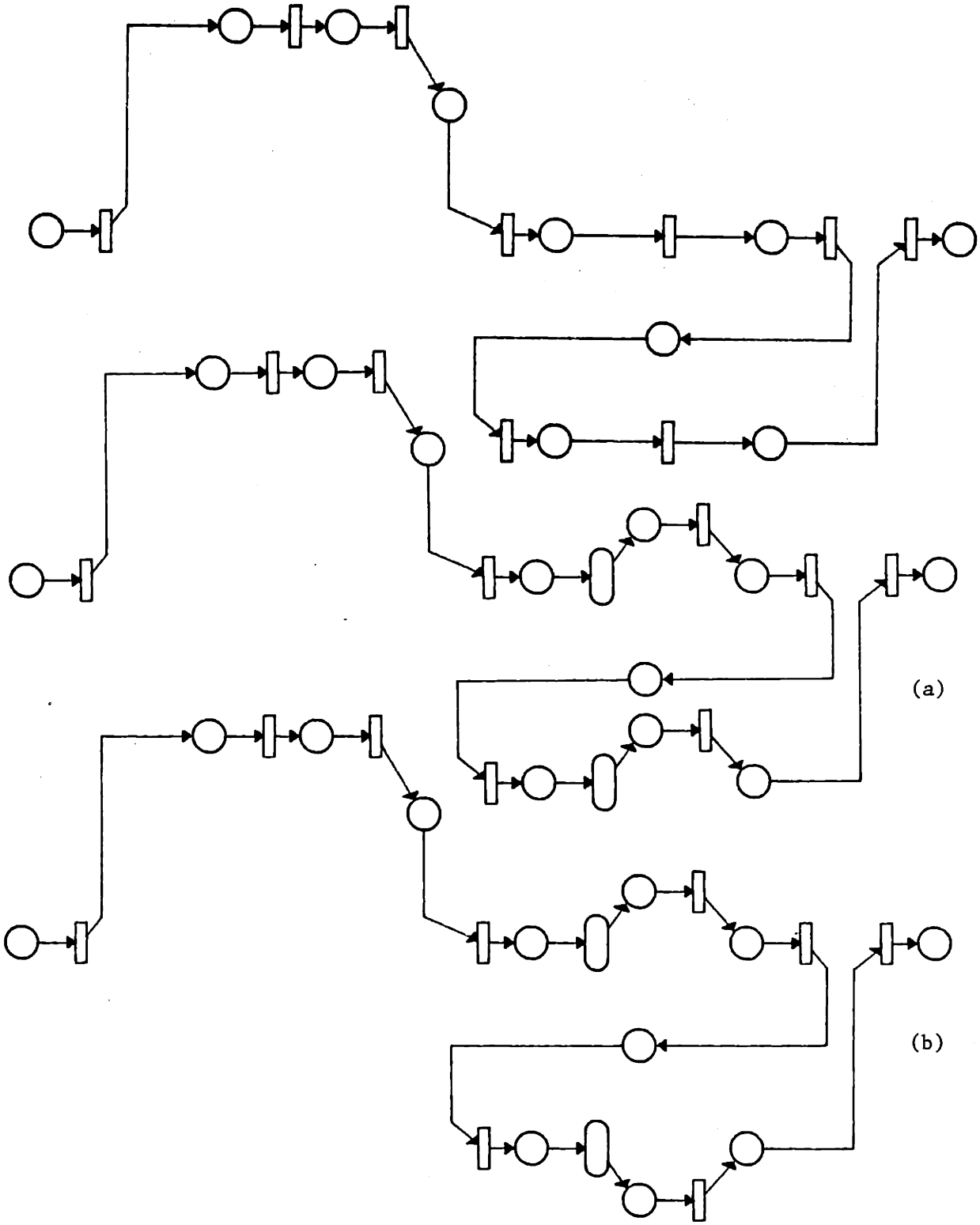


Figure 4.22: Hierarchical Model: Flow Path No. 4 with instantiations (a), (b)

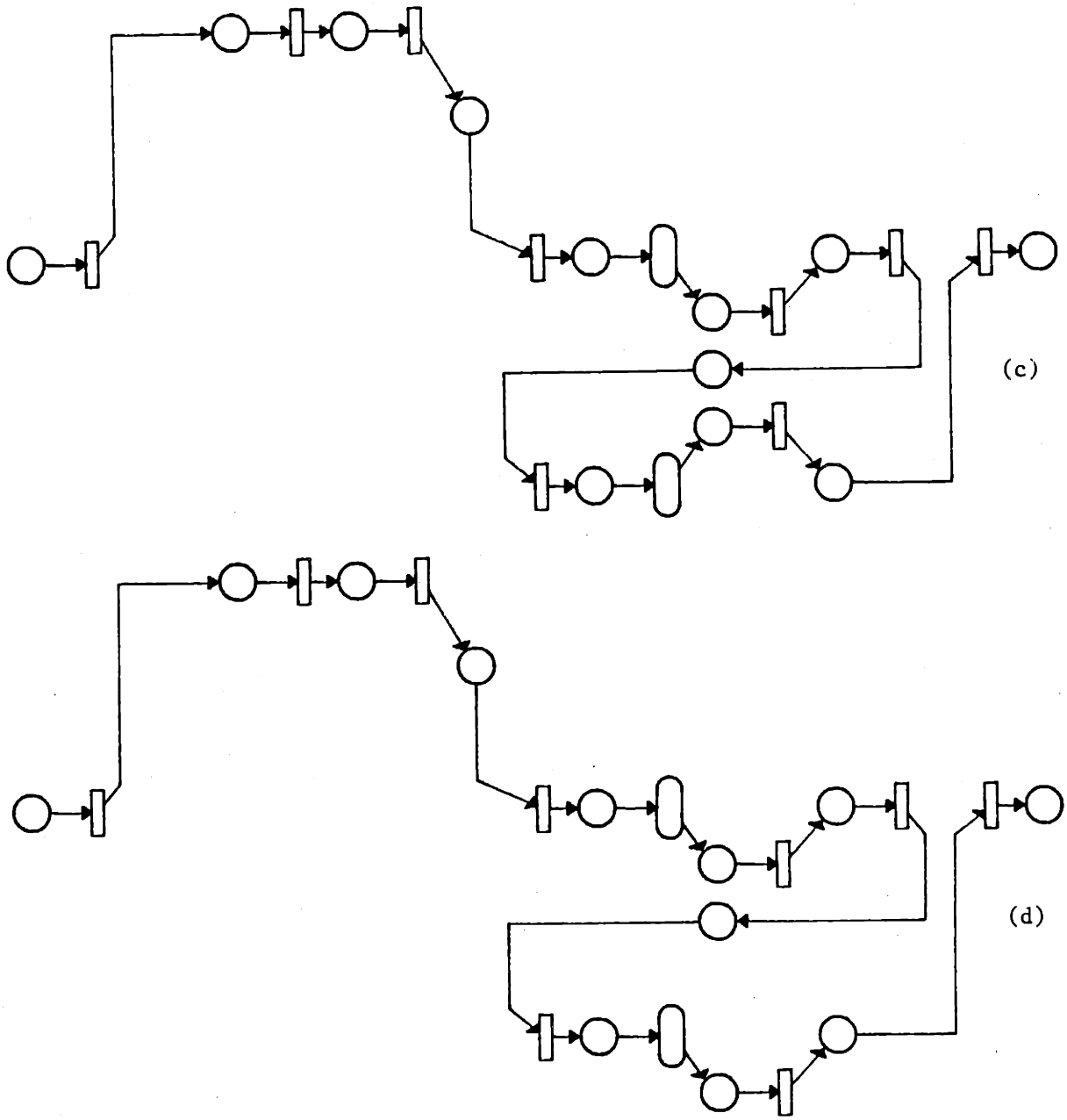


Figure 4.23: Hierarchical Model: Flow Path No. 4 with instantiations (c), (d)

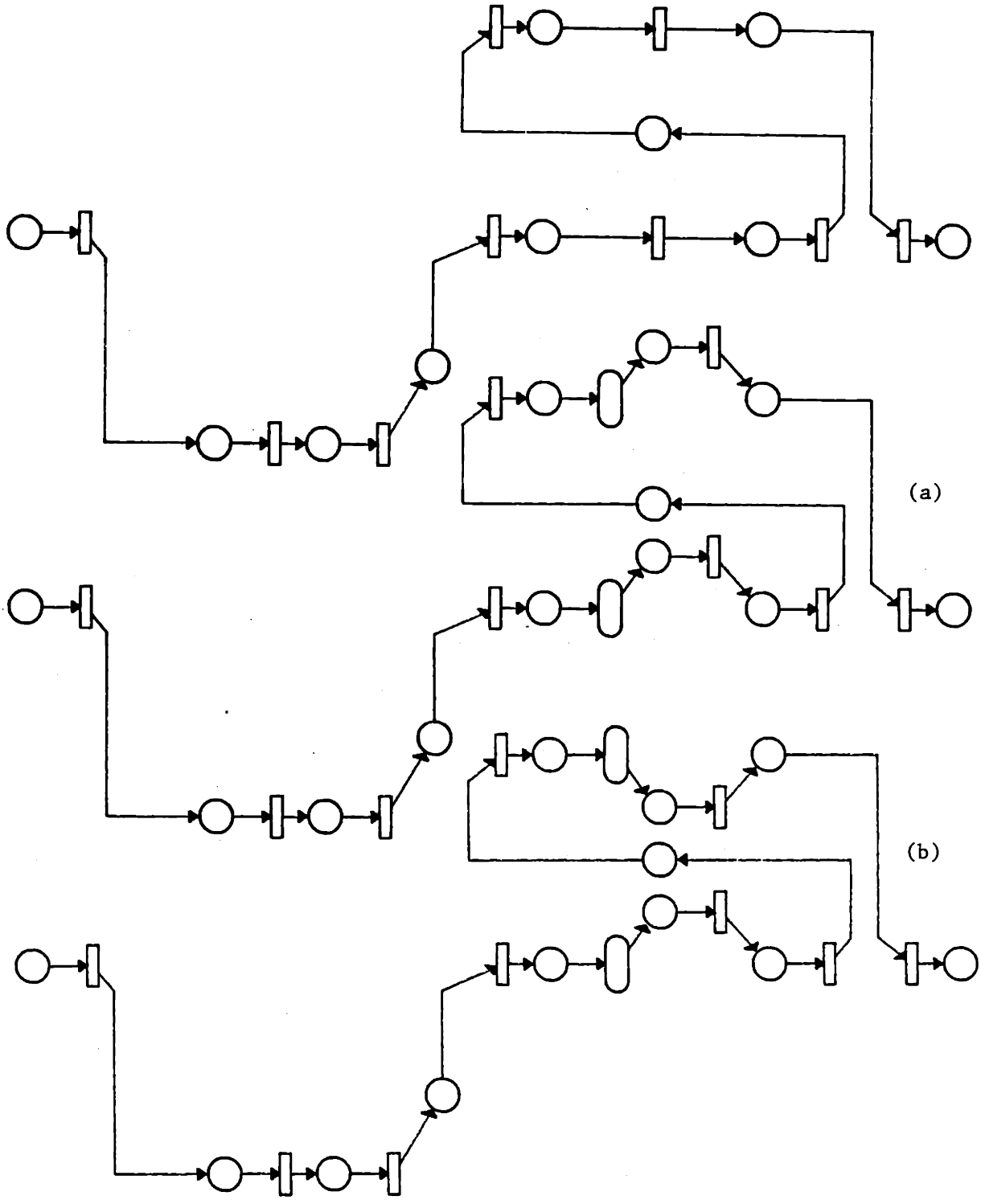


Figure 4.24: Hierarchical Model: Flow Path No. 5 with instantiations (a), (b)

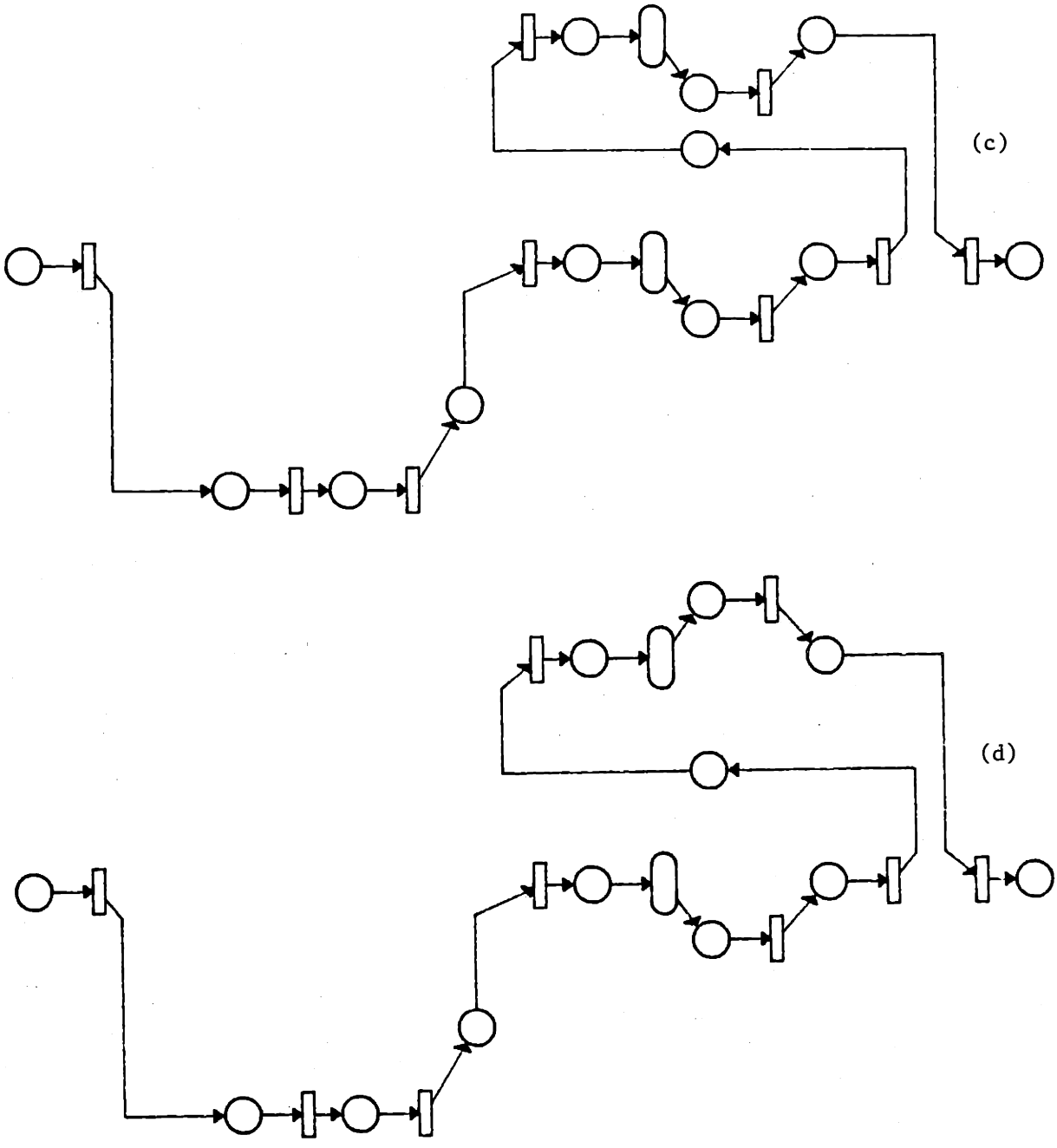


Figure 4.25: Hierarchical Model: Flow Path No. 5 with instantiations (c), (d)

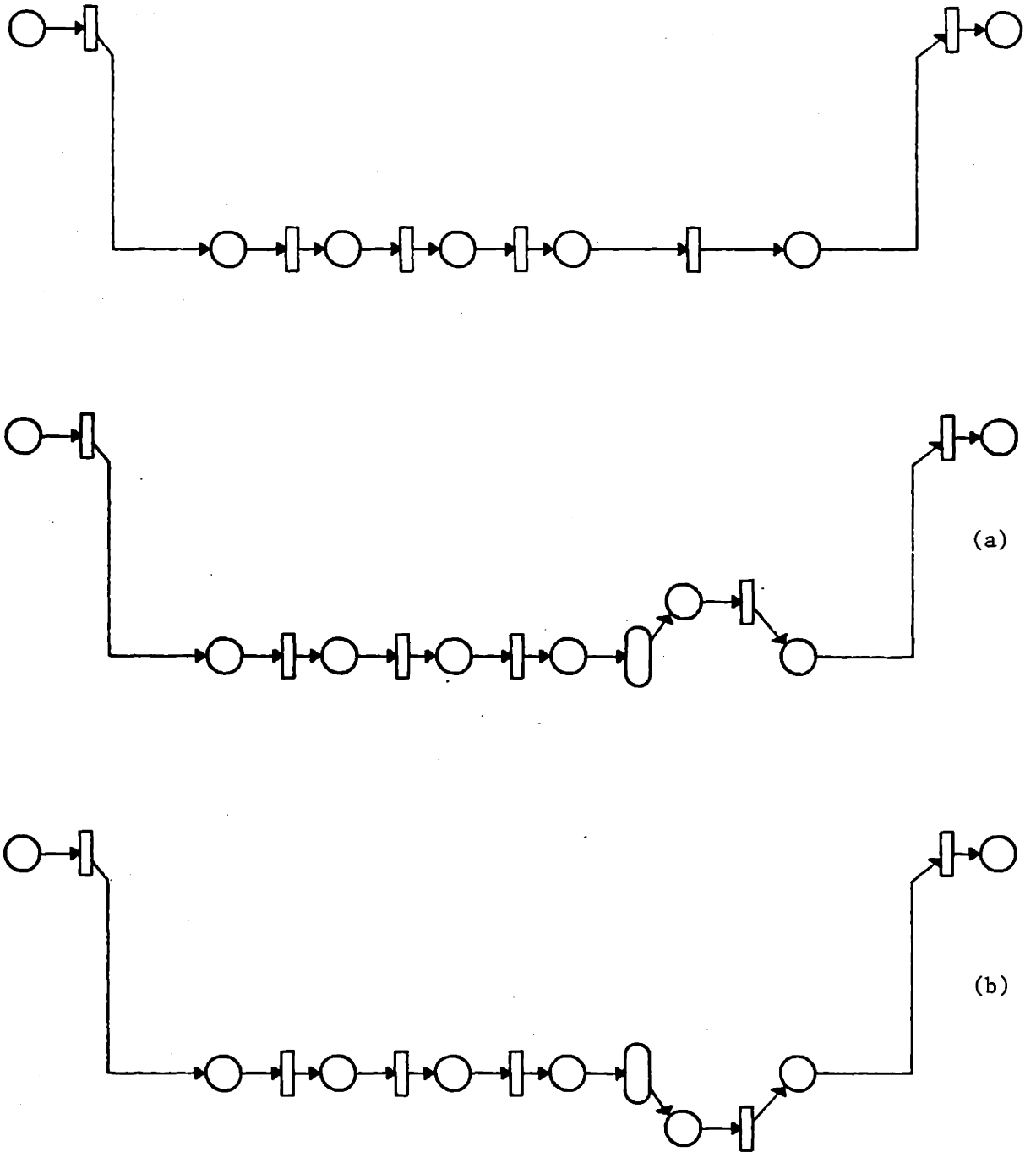


Figure 4.26: Hierarchical Model: Flow Path No. 6 with instantiations (a), (b)

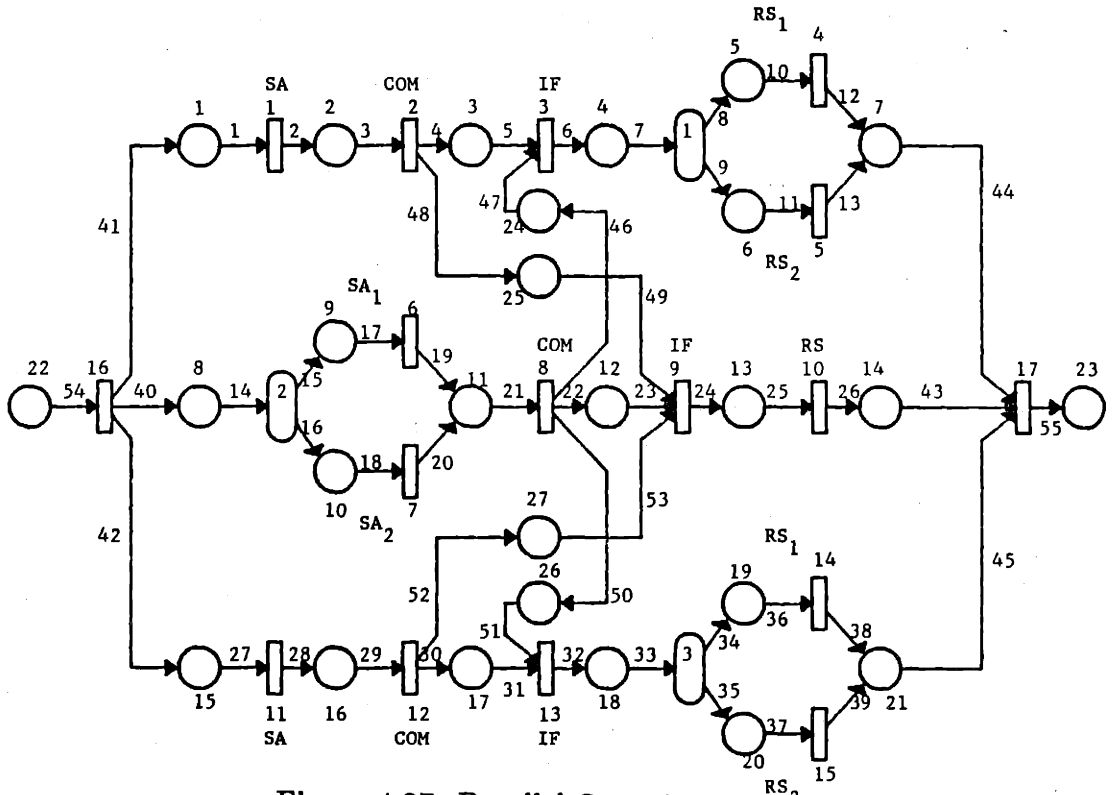


Figure 4.27: Parallel Organization

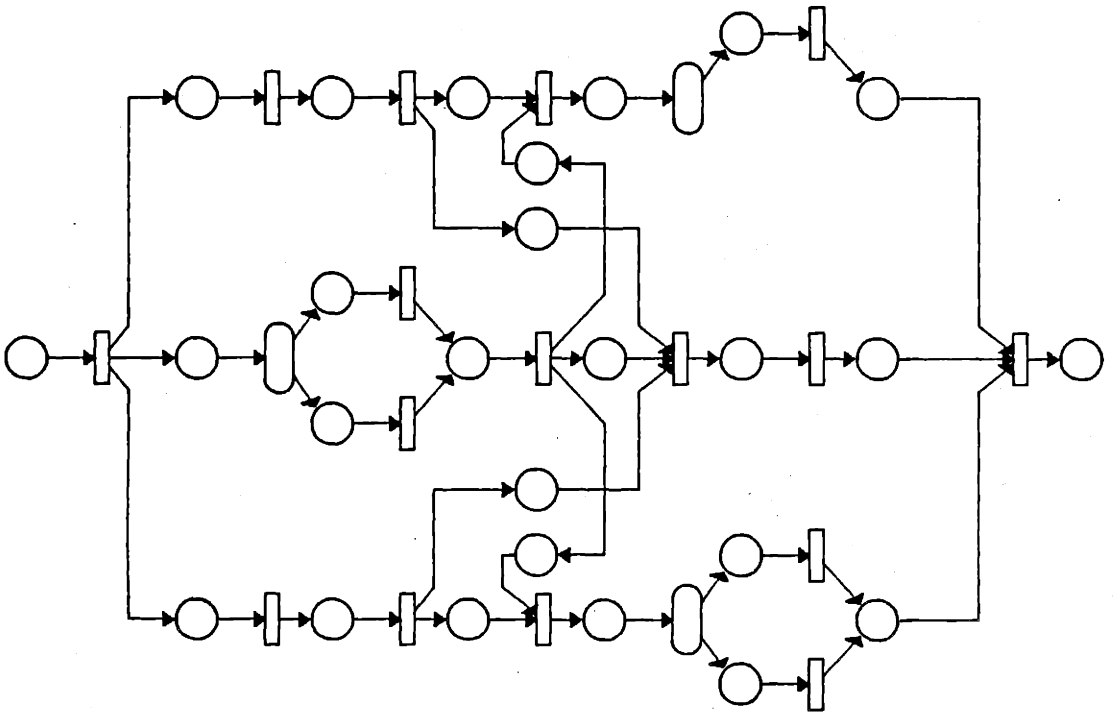


Figure 4.28: Implementation of the SWITCH command

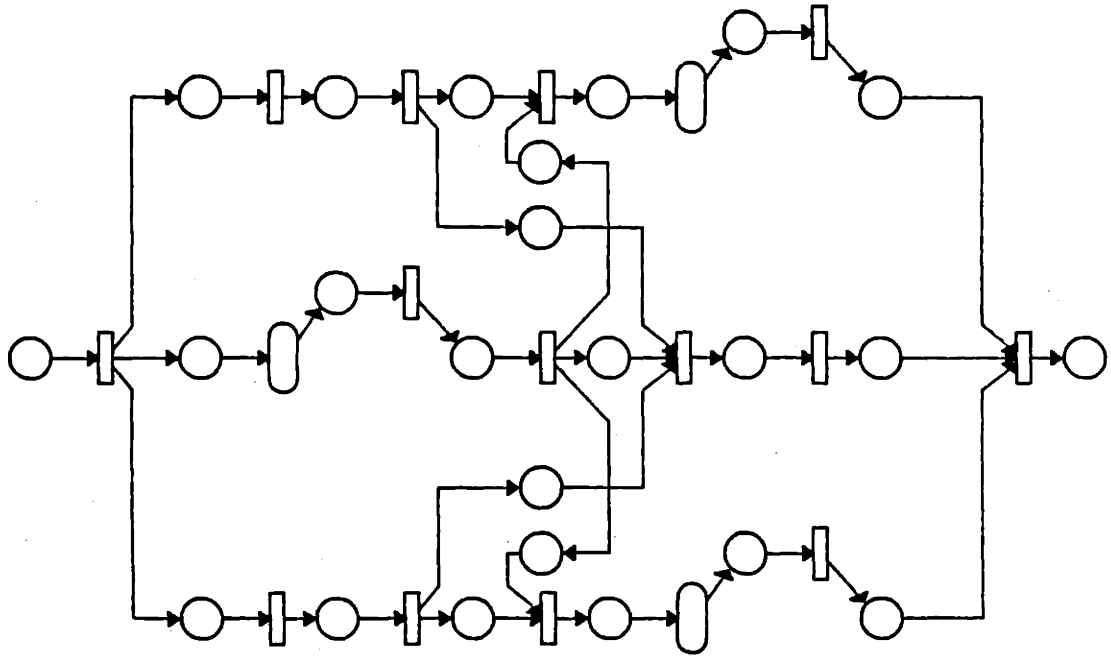


Figure 4.29: Parallel Organization:Instantiation 1

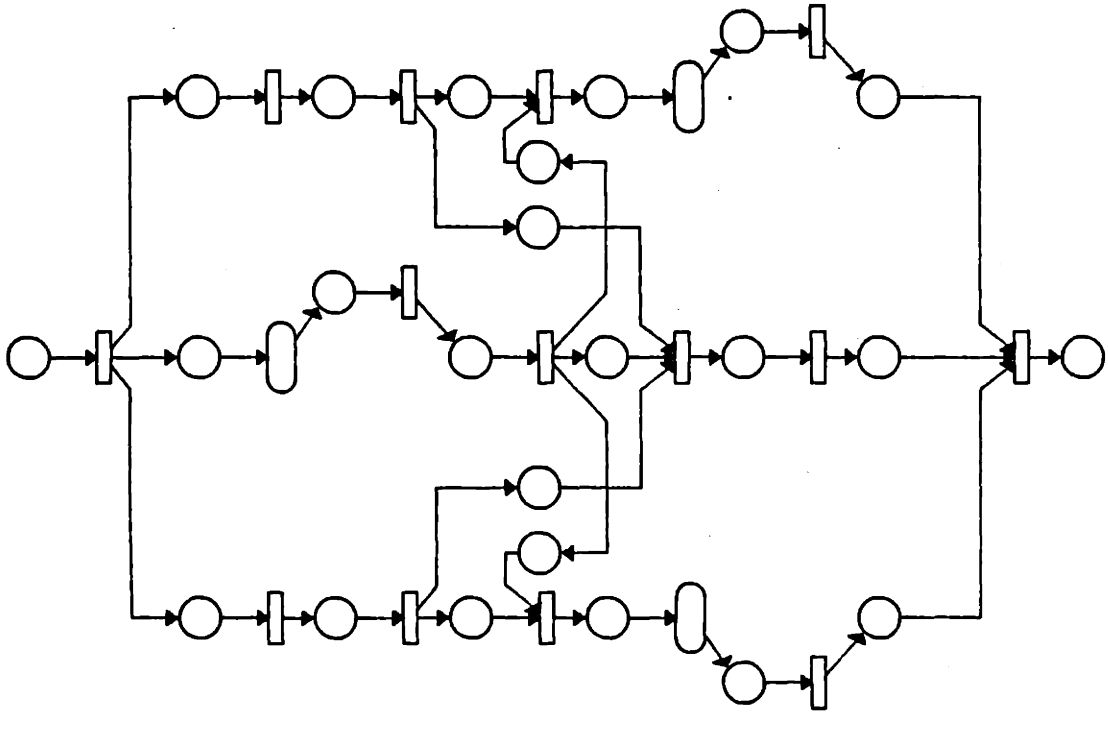


Figure 4.30: Parallel Organization:Instantiation 2

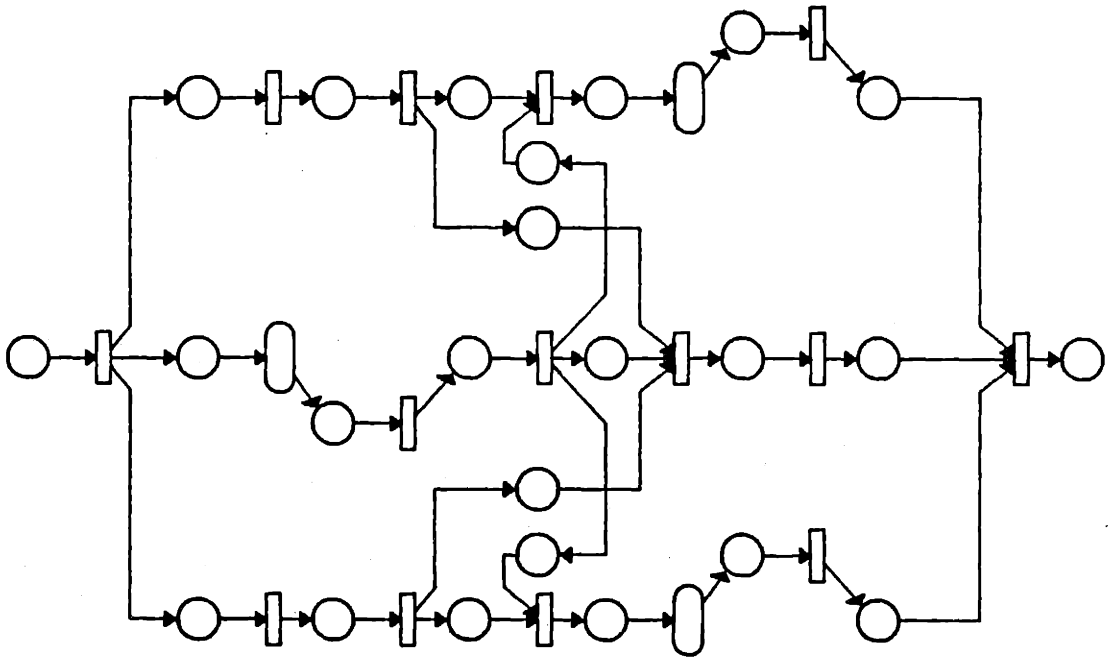


Figure 4.31: Parallel Organization:Instantiation 3

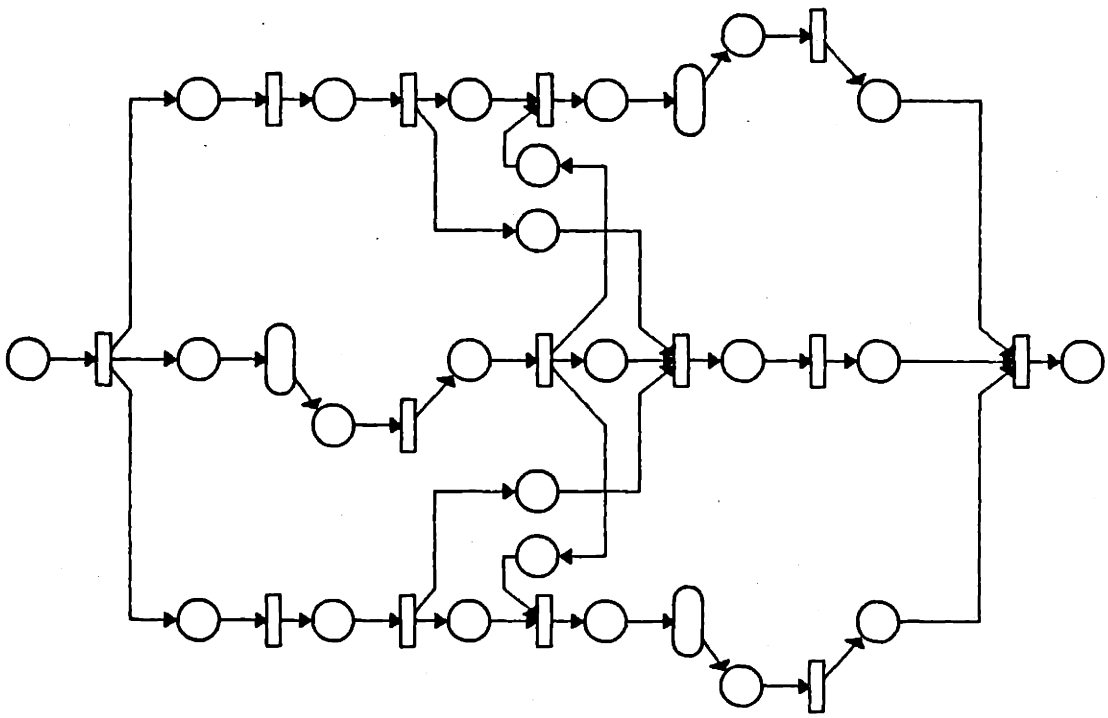


Figure 4.32: Parallel Organization:Instantiation 4

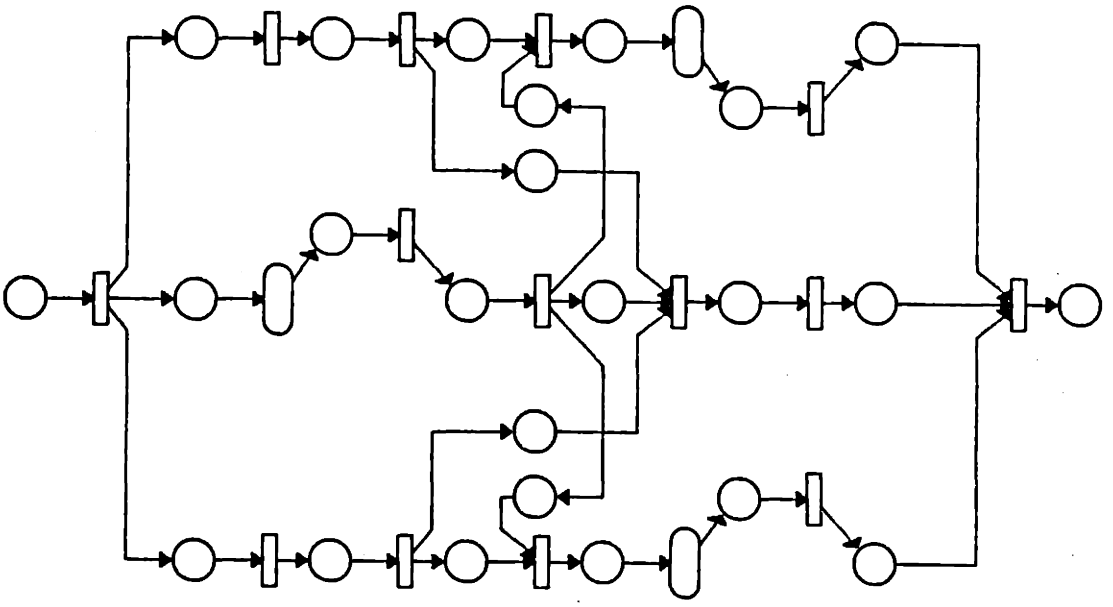


Figure 4.33: Parallel Organization:Instantiation 5

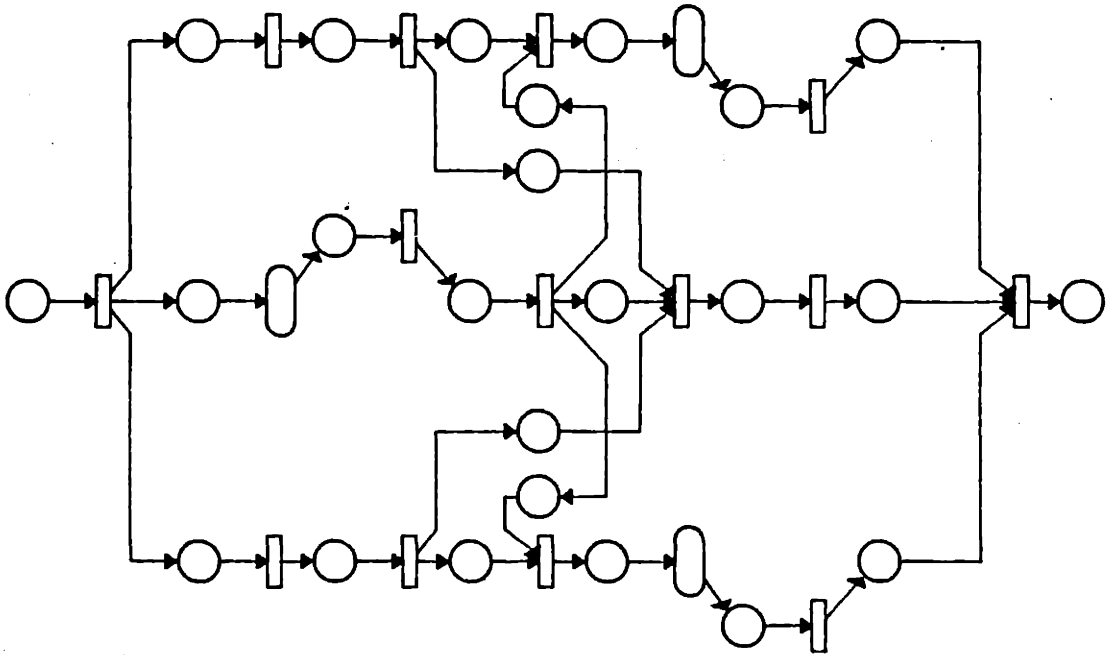


Figure 4.34: Parallel Organization:Instantiation 6

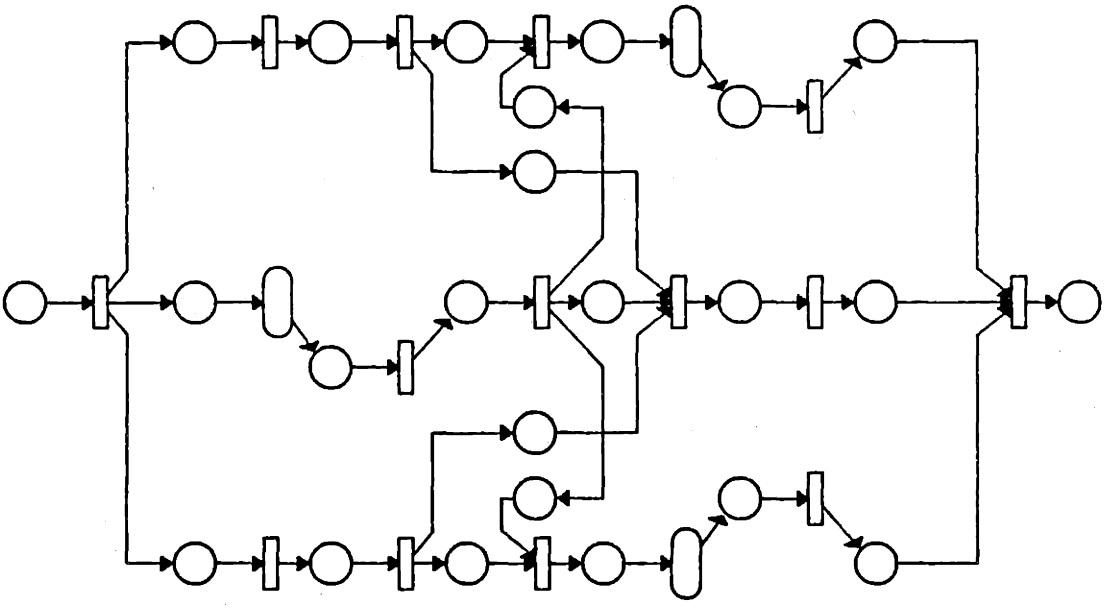


Figure 4.35: Parallel Organization:Instantiation 7

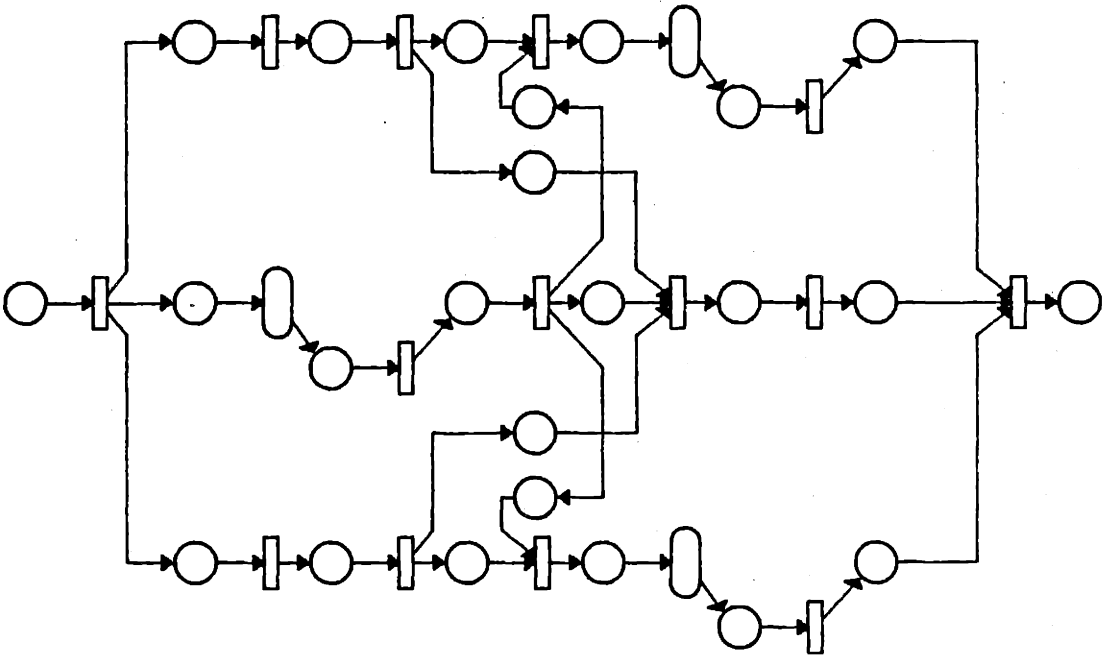


Figure 4.36: Parallel Organization:Instantiation 8

```
*****  PETRI NET TEXT EDITOR  *****


     ENTER FILE NAME [.GRF] :

     CHOICES:
       1. PLACE
       2. TRANSITION
       3. SWITCH
       4. CONNECTOR
       5. EXIT

     ENTER CHOICE (1, 2, 3, 4, 5):
```

Figure 4.37: Text Editor Screen

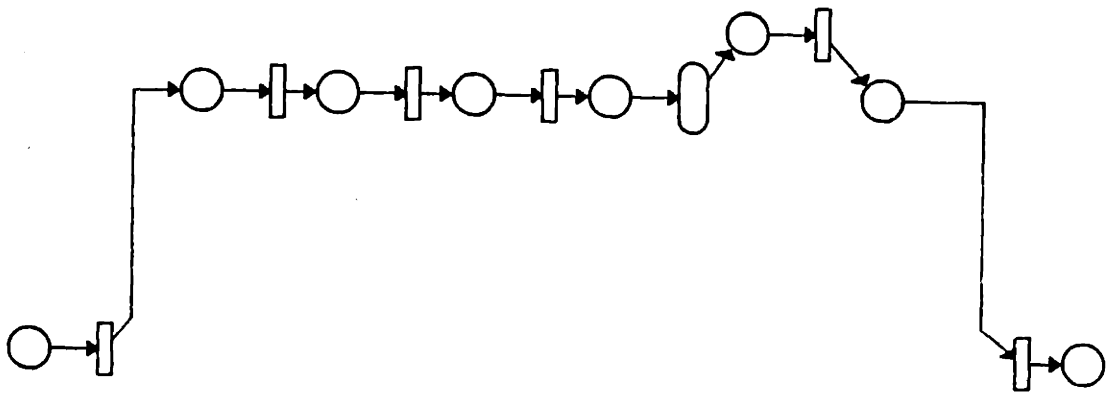


Figure 4.38: Information Flow Path No. 1 for the Parallel Case

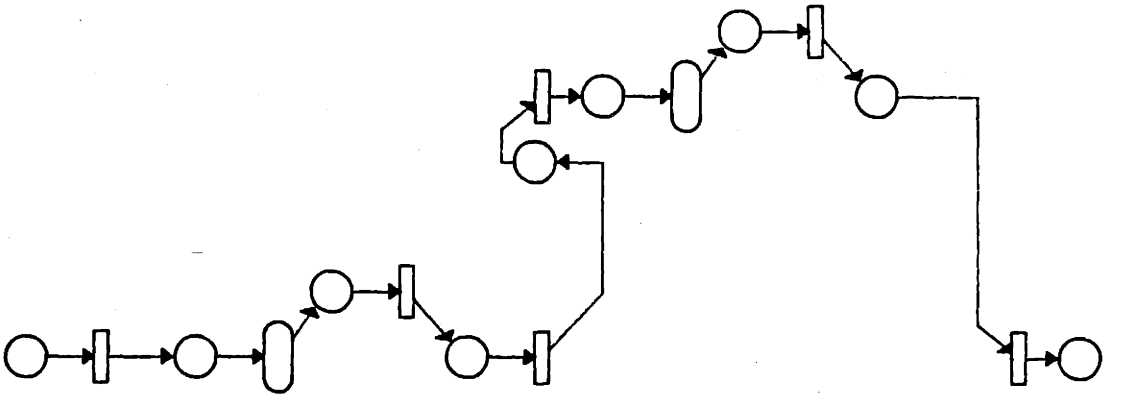


Figure 4.39: Information Flow Path No. 2 for the Parallel Case

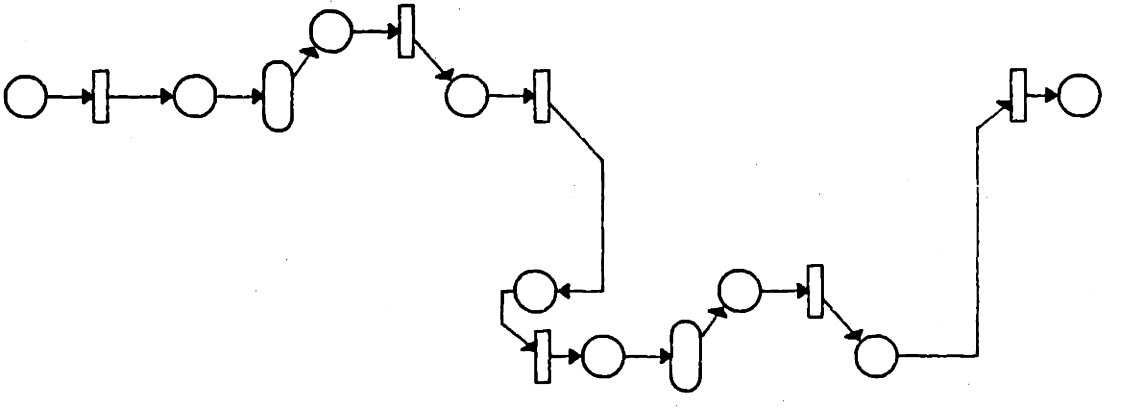


Figure 4.40: Information Flow Path No. 3 for the Parallel Case

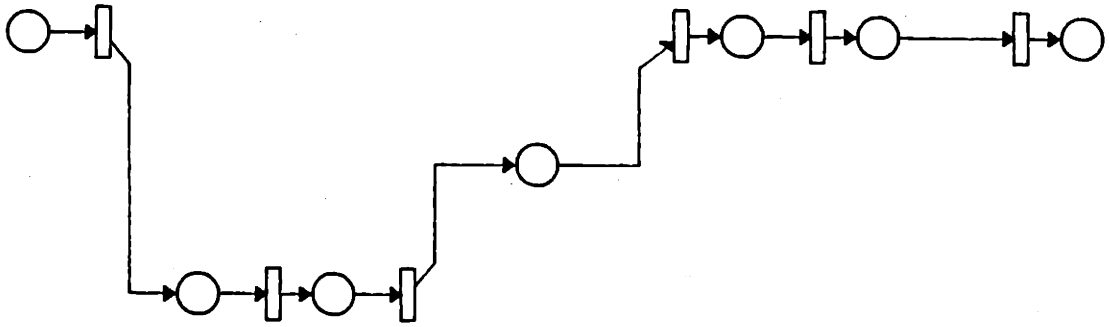


Figure 4.41: Information Flow Path No. 4 for the Parallel Case

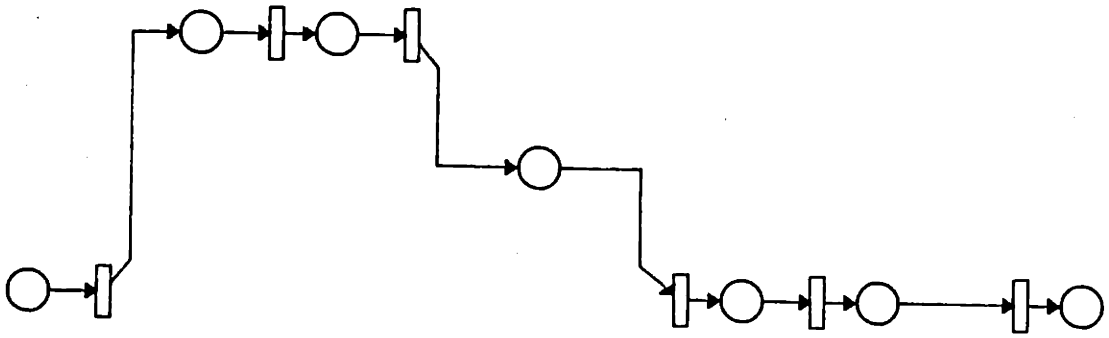


Figure 4.42: Information Flow Path No. 5 for the Parallel Case

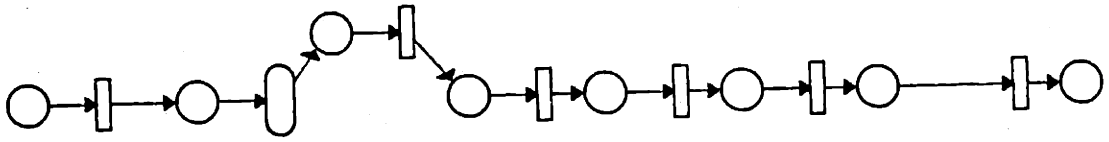


Figure 4.43: Information Flow Path No. 6 for the Parallel Case

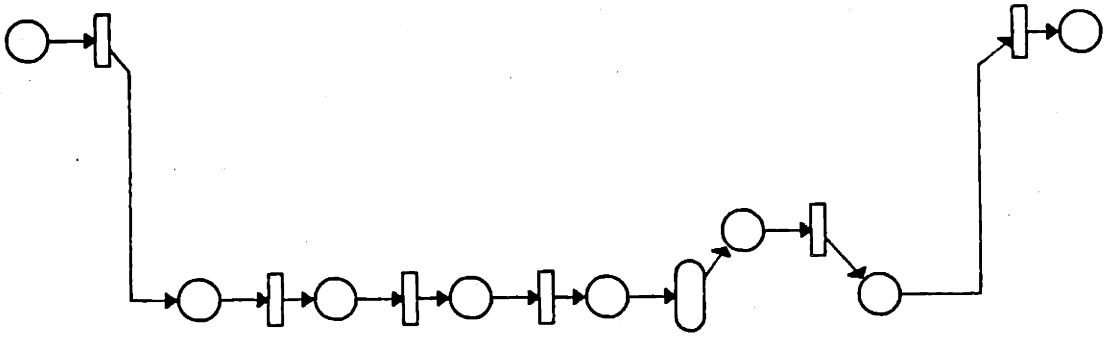


Figure 4.44: Information Flow Path No. 7 for the Parallel Case

Chapter 5

CONCLUSIONS

5.1 Introduction

This chapter restates the problem, highlights the PN/CAD Software System attributes and uses, and draws some conclusions. Finally, possible future enhancements of the System are discussed.

The purpose of the present work is to develop a Petri Net Computer Aided Design (PN/CAD) Software System for creating organizational architectures of arbitrary complexity. The Petri Net representation must be used for modeling such organizations. The System should provide tools for analysis and manipulation of organizational attributes and a standard input/output interface.

5.2 Highlights of the PN/CAD System

In this thesis, a Petri Net Computer Aided Design Software System is developed. The PN/CAD System is used for constructing and analyzing arbitrary organizational architectures . It can create or decompose supernodes, implement switches, combine generic architectures to form a large organizational structure, store information represented by places, transitions, switches or connectors, and produce specific structural properties of the Petri Net. The PN/CAD System is composed of four Functions, see Fig. 3.1 :

- A Graphics Editor, used for the interactive generation of the Petri Net structure of the organization.

- A Text Editor, used to store, modify and retrieve attributes assigned to places, transitions, switches and connectors.
- An Analysis Function, used to generate the interconnection matrix, the incidence matrix and the information flow paths of an organization.
- A Hardcopy Function, used to produce a hardcopy of the graphic image of the organizational architecture and its information flow paths.

In addition, at Top Command Level, a graphics command interpreter evaluates the commands and activates the requested mode of operation. A File System provides a large set of auxiliary services such as File Request, File Manipulation and File Managment. Grammar-Rule algorithms check for the proper graphical construction of the organization, and enforce the constraints during analysis of organizational architectures.

The PN/CAD Software drives a wide variety of output devices such as: various printers, two and six pen plotters, the Professional Graphics Display monitor, when it operates with the Professional Graphics Controller, and various monitors when it operates with the Enhanced Graphics Adapter.

5.3 Design Overview

The interactive organizational design process involves three phases. The first phase involves the conceptual and graphic design stages. At first the conceptual classification of the organization must take place. A set of Petri Net generic structures is identified from the organizational model, and constructed graphically by invoking the Graphics Editor. These structures are used as building blocks for composing the overall architecture. For example, in the case of developing a DDMO architecture, two, three, or four stage Petri Net models of the single interacting DM, may serve as building blocks. A basic library of models, representing designer's or system contraints, may be constructed and used. Structural decomposition is used to resolve an existing architecture into constituent

components. Then, the notion of graphical abstraction is introduced for representing a subnet as a single entity. A subnet may be defined as a supernode. In such a case, supernodes replace subnets in the graphical structure, thus creating a coarsed version of the organizational model. A case where an n-decision switch is present in the organization, the designer has the tools to implement a particular decision strategy and generate the respective organizational structure. The PN/CAD Software System has its own metafile system to store the graphics structure of the organization.

The second phase involves the construction of an alphanumeric data structure by invoking the Text Editor. The data structure contains attributes which correspond to representations of places, transitions, switches and connectors in the graphic image. Attributes may include processing algorithms and delays for transitions, token capacity for places, functional relations and probabilities for connectors, file names for subnets, etc.

The third phase involves the creation of an analytical description of the organizational model. Data processing algorithms access the metafile, process its contents, and generate the incidence matrix. The incidence matrix serves as input to an algorithm which produces a matrix containing the minimal supports of S-invariants. The matrix is used to create the information flow paths.

5.4 Conclusions

The PN/CAD Version 1.0 Software System is used for the graphical and analytical description of organizational architectures. The System has been tested and it has been found to have features that make it capable of being adapted easily to the designer's needs.

The Graphics Editor capabilities are essential for the growth and flexibility of the design. The Text Editor provides easy attribute access. The attributes can be used from Graphics Editor commands to generate the family tree of an organizational architecture by implementing different switch settings. The Analysis Function provides a set of

descriptive analytical tools. The Hardcopy Function produces standard output on a plotter or printer. Novice users may access the HELP facility.

The internal structure of the PN/CAD System is modular. The systems programmer may improve already existing algorithms or test new ones easily. New Functional Modules may be defined and added. In terms of analytical description, the PN/CAD Analysis Function can be used only to a limited extent. Succesful enhancements of the PN/CAD Software System will influence modeling, analysis and design of organizations in both theoretical and practical terms.

5.5 Enhancements of the PN/CAD System

Further enhancements of the PN/CAD System may be implemented at different levels, from algorithm improvements, to a larger number of available commands within each Functional Mode, to more Functional Modes and a better user interface.

The systems programmer must introduce new algorithms and expand the library of analytical tools available. The motivation behind it is that structural properties are used for stability tests and evaluation of the system performance. Boundedness and liveness are the standard structural properties used to test the stability of the Petri Net model against any physical constraints or designer's specifications. In terms of these two properties, other analysis problems can be posed, such as reachability and coverability. The mathematical representation of these properties is related to the solution of linear algebraic equations. The corresponding modular implementation fits into the Structural Analysis Mode.

Simulation is one of the tools available to the system designer to give a feeling of how the system behaves. For this reason, once a stability analysis has been carried out, the system designer may be interested in obtaining the state space of the Net. Each state is identified by its marking, and the transition to the next state is defined by a set of firing rules. Each time a transition fires, a token is deleted from each of the input places, and created in each of the output places. This token flow can be shown

graphically. If the user associates a time delay (deterministic or stochastic) with each transition, and assigns an initial marking, a Petri Net simulation can be implemented. From the simulation results the total response time of the Petri Net can be calculated.

The above notions can be implemented at a Functional Level, using a set of algorithms, see Fig. 5.1. The design of the Simulation Mode algorithms must be done keeping in mind the applications at hand, the performance evaluation and property verification of the Petri Nets. Next, the sequence of tasks for such an algorithm is presented:

- decide on the type of simulation to be carried out,
- define the set of rules,
- enter Petri Net attributes, using the Text Editor,
- enter initial marking of the net,
- check for syntactical consistency of attributes,
- carry out stability analysis,
- start execution of the Petri Net,
- display graphically token movement,
- show marking vector and store any related information

The means of communication between the designer and the PN/CAD System is the keyboard. The keyboard is used primarily to enter text when in text mode or pick graphic commands when in graphics mode. When in graphics mode, however, the positioning of the cursor using keyboard commands is slow. Also, when selecting a location, the cursor resolution is below standard. There is a need to introduce a better interface, such as a mouse. The mouse is a small device which the designer can hold and roll it on a flat, smooth surface. A pointer on the screen follows the motion of the mouse. In order for an action to take place the designer must press and

release the mouse button. The mouse provides fast pointer movement and very good location resolution. The current implementation of GKS supports the mouse but there are compatability problems. In order to access the mouse library, a device driver must be written in assembly language. The PN/CAD System is written in IBM Professional Fortran, while the mouse library can be accessed directly by Microsoft Fortran.

Another way to improve the PN/CAD System is to introduce windows. For example, when the user is in Graphics Mode, all information flow paths of an organization may be diplayed in separate windows.

The PN/CAD System may be improved in many ways. The system designer should keep up with the new software in the market and implement only the functions that will enhance the designer's ability to use the system effectively.

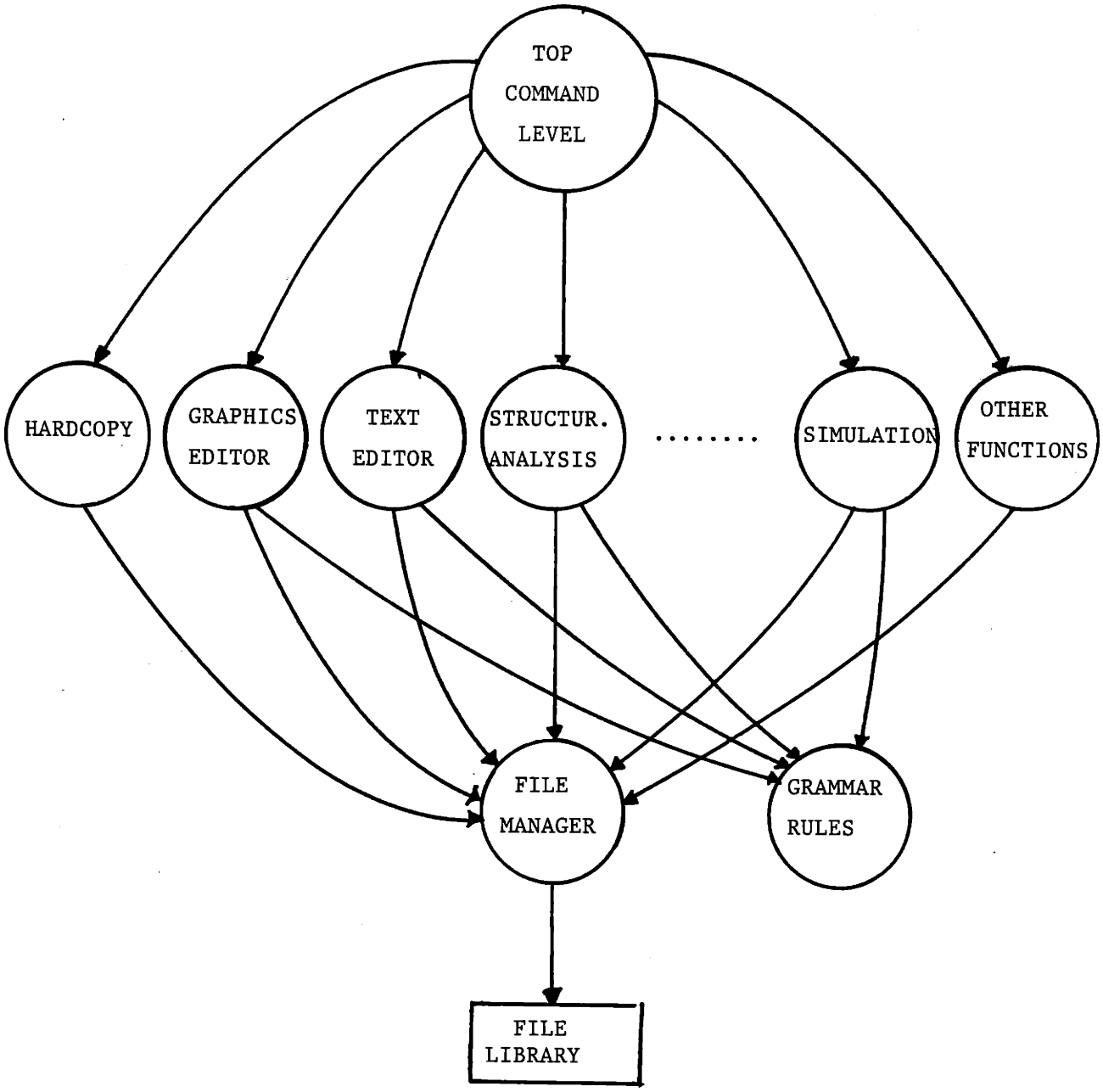


Figure 5.1: System Chart for Future Enhancements

Bibliography

[1] S. K. Andreadakis and A.H. Levis, "Design Methodology for Command and Control Organizations," *Proc. 1987 C^2 Research Symposium*, Washington D.C., June 1987.

[2] J.L.Peterson *Petri Net Theory and the Modelling of Systems*, Prentice Hall, Englewood Cliffs, New Jersey, 1981

[3] D. Tabak and A.H. Levis, "Petri Net Representation of Decision Models," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol SMC-15, No 6, Nov/Dec 1985.

[4] C.A. Petri, "Introduction to General Net Theory," Lecture Notes in Computer Science, No 84, Springer Verlag, Berlin, FRG, pp. 1-20, 1979.

[5] P.S. Thiagarajan, "Some Aspects of Net Theory", Lecture Notes in Computer Science, No. 207, Springer Verlag, Berlin, FRG, 1985.

[6] W. Reisig, *Petri Nets: An Introduction*, Springer Verlag, New York, 1982.

[7] P. Remy, "On the Generation of Organizational Architectures," MS Thesis, Report LIDS-TH-1630, Laboratory for Information and Decision Systems, MIT, Cambridge, MA, 1986.

[8] V. Bouthonnier and A.H. Levis, "Effectiveness Analysis of C3 Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-14, No. 1, January/February 1984.

[9] P.H. Cothier, "Assessment of Timeliness in Command and Control," S.M. Thesis, Report LIDS-TH-1391, Laboratory for Information and Decision Systems, MIT, Cambridge, MA, 1984.

[10] V.Y. Jin, A.H. Levis and P. Remy, "Delays in Acyclical Distributed Decisionmaking Organizations," *Proc. Fourth IFAC Symposium on Large Scale Systems : Theory and Application*, Zurich, Switzerland, 1986.

[11] S.K. Andreadakis and A.H. Levis, "Accuracy and Timeliness in Decisionmaking Organizations," *Proc. 10th IFAC World Congress*, Munich, FRG, July, 1987.

[12] H. Hillion, "Performance Evaluation of Decisionmaking Organizations Using Timed Petri Nets," SM Thesis, Report LIDS-TH-1590, Laboratory for Information and Decision Systems, MIT, Cambridge, MA, 1986.

[13] K.L. Boettcher and A.H. Levis, "Modeling the Interactive Decisionmaker with Bounded Rationality," *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-12, No 3, May-June 1982.

[14] C.E. Shannon and W. Weaver, *The Mathematical Theory of Communication*, University of Illinois Press, Urbana, Illinois, 1963.

[15] R.C. Conant, "Laws of Information which Govern Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol SMC-6, No 4, April 1976.

[16] H. Alaiwan and J.M. Toudic, "Recherche des Semi-Flots, Verrous et des Trappes dan les Reseaux de Petri," *Technique et Science Informatiques*, Vol. 4, No. 1, Dunod, France, 1985.

[17] S.K. Andreadakis, *personal communication*

LABORATORY FOR INFORMATION AND DECISION SYSTEMS

COMPUTER AIDED DESIGN FOR PETRI NETS (PN/CAD)

USER'S DOCUMENT

Document Number:PRIMER87 1.00

Author: John Kyratzoglou

Date: August 15, 1987

System: IBM 5170 PC AT

Abstract: This is an introductory document containing general information and reference material on how to use the Petri Net Computer Aided Design System.

Table of Contents

List of Figures

Appendix A

User's Document

A.1 System Configuration

This section provides useful technical information on the hardware needed that allows you to use the PN/CAD System. It describes the two modes of operation and the procedures for installing the drivers on your system.

The PN/CAD applications package can operate in two modes. The mode of operation depends on the hardware installed on your system. The PN/CAD System can operate in the Professional Graphics Controller (PGC) mode or the Enhanced Graphics Adapter (EGA) mode. Each mode requires different hardware configuration.

A.1.1 PGC Mode

The PGC mode requires an IBM PC Professional Graphics Controller (No. 1501) and an IBM 5175 PC Professional Graphics Display (PGD). The Controller fits into two adjacent slots in an

- IBM 5161 PC Expansion Unit,
- IBM 5160 PC XT, or
- IBM 5170 PC AT

You need to install the VDIDYPGD.SYS device driver for the PGD monitor, the VDI Controller, the drivers for the peripheral devices and to modify the AUTOEXEC.BAT

and CONFIG.SYS files according to instructions given in a later paragraph in this section.

A.1.2 EGA Mode

The EGA Mode requires an IBM Enhanced Graphics Adapter (No. 1200). The Adapter provides

- support for the IBM 5154 Enhanced Color Display, with a 640 × 350 resolution, 16 color-graphics and an 8 × 14 character box for text in graphics mode,
- enhanced support for the IBM 5153 Color Display, in either 320 × 200 medium definition graphics or 640 × 200 high definition graphics modes, 16-color graphics and an 8 × 14 character box
- graphics support for the 5151 Monochrome Display, in 640 × 350 graphics mode with an 8 × 14 character box in graphics mode and a 9 × 14 character box in text mode.

The IBM Graphics Memory Expansion Card (No.1201) expands the IBM EGA's 64Kb memory to 128Kb and increases the color range. The IBM Graphics Memory Module Kit (No.1203) expands the Expansion Card's memory to 256Kb providing additional graphics functions, such as panning, scrolling and additional pages of graphics data. Both cards are needed, when you develop or run the PN/CAD applications package. These can be installed in sockets inside the IBM EGA Card. If an IBM EGA card is being used to replace the IBM Monochrome Display and Printer Adapter, an IBM Printer Adapter or a Serial Parallel port must be added to replace the printer adapter function. In such a case the Monochrome Display and Printer Adapter must be removed.

The EGA card can be installed in any expansion slot in an:

- IBM 5170 PC AT (Not on 5161 Expansion Unit)

- IBM 5160 PC XT
- IBM PCs with a serial number higher than No.0300960 (the serial number is on the back of the unit).

The user needs to install the appropriate EGA device driver. The driver installed in this mode depends on the monitor available. The user must install the VDI Controller, the drivers for the peripheral devices, and to modify the AUTOEXEC.BAT and CONFIG.SYS file according to instructions given in a later paragraph in this section.

A.1.3 Technical Information

Hardware Requirements: To run the PN/CAD System you need:

- IBM 5170 PC AT or any compatible,
- 512 Kb memory,
- an IBM fixed disk and one dual sided high density (1.2Mb) diskette drive,
- the PGC or the EGA card with the corresponding monitor, as described in sections A.1.1 and A.1.2,
- IBM 8087 or IBM 80287 Math Co-Processor,
- a printer (optional),
- a two or six pen plotter (optional).

Software Requirements:

- IBM PC Disk Operating System (DOS) 2.1 or higher,
- A set of DOS device drivers for the:

- Virtual Device Interface (VDI) Controller,
- IBM PGD monitor (PGC mode),
- IBM EGA and its supported monitors (EGA mode),
- IBM Printer (Graphics, Color, Compact, etc.),
- IBM two or six pen plotter.

- For application development you need one of the following language bindings
  - IBM Professional Fortran 1.0 Compiler,
  - IBM Pascal 2.00 Compiler,
  - IBM Lattice C 2.0 Compiler and,
  - the files shown in the table below:

Professional Fortran	Lattice C
GKS.LIB	CGKS.LIB
PFGKS.LIB	GERR_HND.C
PFGKS.OBJ	GERR_LOG.C
GERHND.FOR	LINK.EXE
GERLOG.FOR	
LINK.EXE	

Packaging

The PN/CAD Software is distributed on two diskettes and the User's Document. The diskettes contain all PN/CAD software, the VDI Controller, supported device drivers and related code.

Diskette 1 contains:

AUTOEXEC.BAT
A sample batch file which is automatically executed when start DOS. The file may set the DOS path to search specified directories, or to load device drivers.

You must insert the INIT_VDI.EXE for the VDI system. It can contain DOS SET commands that you supply to set logical names to physical names. This file is used to start the IBM 5170 PC AT at LIDS with a #5175 PGD monitor to run the PN/CAD Software.

CONFIG.SYS
A sample procedure file containing a list of DOS VDI device drivers names. This file is the configuration file of the IBM PC AT.

INIT_VDI.EXE
This file re-initializes the VDI Controller.

LINK.EXE
The IBM PC Linker 2.3 is used to link the GKS language libraries.

GKS.LIB
Library required by Fortran binding.

PFGKS.LIB
Professional Fortran binding library.

PFGKS.OBJ
Professional Fortran binding object module.

GERHND.FOR
Sample error handler routine for Fortran.

GERLOG.FOR
Sample error logging routine for Fortran.

The subdirectory DRIVERS contains all device drivers:

VDIDY004.SYS

IBM PC 320 × 200 four color medium resolution graphics driver.

VDIDY006.SYS

IBM PC 640 × 200 two color high resolution graphics driver.

VDIDY008.SYS

IBM PCjr 160 × 20016 color low resolution graphics driver.

VDIDY009.SYS

IBM PCjr 320 × 200 four color high resolution graphics driver.

VDIDY00D.SYS

IBM Enhanced Graphics Adapter 320 × 200 16 color driver.

VDIDY00E.SYS

IBM Enhanced Graphics Adapter 640 × 200 16 color driver.

VDIDY00F.SYS

IBM Enhanced Graphics Adapter Monochrome monitor driver.

VDIDY010.SYS

IBM Enhanced Graphics Adapter with Enhanced Color Display 640 × 350 16 color driver.

VDIDYPGD.SYS

Professional Graphics Display Monitor

VDIPLSIX.SYS

IBM 7372 Color Plotter (six pen) graphics driver.

VDIPLTWO.SYS

IBM 7371 Color Plotter (two pen) graphics driver.

VDIPRCOL.SYS
IBM Color Printer graphics driver.

VDIPRCOM.SYS
IBM Compact Printer graphics driver.

VDIPRGRA.SYS
IBM Graphics Printer driver.

VDI.SYS
A file that contains the resident portion of the VDI Controller.

Diskette 2 contains the PN/CAD software and its supported files:

INSTALL.BAT
A file containing the installation program.

PNCAD.EXE
The executable code of the Petri Net Computer Aided Design (PN/CAD) Software.

PN.BAT
Command to run PN/CAD Software.

DM1.GRF, DM2.GRF
Demonstration files.

DIR.GRF, DIR.TXT, DIR.SUP, DIR.SCR
Directory files.

You will need to edit two DOS files as part of the installation. The configuration procedure files are CONFIG.SYS and AUTOEXEC.BAT. The DOS uses these files during the system initialization process.

CONFIG.SYS file: when editing the CONFIG.SYS file you select options specific to your system. A specific device driver is required for each graphics device. The driver must be supported by the VDI Controller and the driver's name is recognized by its abbreviation (see description Diskette 1 list of files). Each driver must be identified in the CONFIG.SYS file. For example,

```
device=\drivers\vdidypgd.sys
device=\drivers\vdiprcom.sys
device=\drivers\vdiplsix.sys
device=\drivers\vdi.sys
```

The above commands indicate that the Professional Graphics Display (PGD mode), compact printer and six pen plotter device drivers are available for the PN/CAD Software. The VDI.SYS file must be included and listed after all device drivers. You have the option of installing the device drivers in the root directory. In this example, the device drivers are installed in the DOS subdirectory DRIVERS (you have to indicate to DOS through the AUTOEXEC.file where it must look for the device driver files.

AUTOEXEC.BAT file: when you edit the AUTOEXEC.BAT file include the line:

```
INIT_VDI.EXE
```

to re-initialize the VDI Controller. The AUTOEXEC.BAT file can contain SET commands used for setting logical names to physical names. The statements in the AUTOEXEC.BAT file corresponding to the CONFIG.SYS statements, listed earlier are:

```
path c:;c:\dos;c:\drivers
init_vdi
set DISPLAY = vdidypgd.sys
set PRINTER = vdiprcom.sys
set PLOTTER = vdiplsix.sys
mode com1:9600, n, 8, 1, p
```

The first command specifies the path where all the drivers are residing. The second command re-initializes the VDI Controller. The SET commands assign logical names to physical names (device driver filenames). The VDI finds the logical names when the PN/CAD program sends any output to a device. The DOS MODE command installs a serial interface in order for the IBM Color Plotter to communicate with the program.

A.2 Installation Procedure

A.2.1 Introduction

The section describes how to prepare backup copies of your PN/CAD Master Diskettes and how you can install the PN/CAD Software.

A.2.2 Backup Procedure

The PN/CAD System comes with two 1.2Mb floppy diskettes, with the name

1. PN/CAD System Master Diskette #1
2. PN/CAD System Master Diskette #2

Before you copy the PN/CAD System make sure that you have a backup copy of the diskettes. The user must make sure that he has two 1.2Mb properly formatted diskettes available. The computer must have an 1.2 Mb high density floppy disk drive. To create a directory on the hard disk, use the DOS command

```
c: > mkdir cad
```

The following command changes the DOS current directory to the current directory path plus CAD:

```
c: > cd cad
```

Then, copy the contents of the diskette to the current directory CAD, by typing,

```
c: > copy a:
```

Insert the new diskette into the drive, and type,

```
c: > copy *  a:
```

This command copies files stored in the CAD directory into the diskette in drive a: . If the process is implemented successfully, delete all files from the CAD directory. If the Master Diskette has a subdirectory, copy all files from the subdirectory to the CAD directory. Insert the new diskette into the drive and create a subdirectory on the floppy diskette. For example,

```
a: > mkdir drivers
```

Then, switch to DOS directory CAD, and copy all files from the CAD subdirectory to DRIVERS one,

```
a: > copy *.sys a:\drivers
```

Copies all files with .SYS as file type to DRIVERS subdirectory.

A.2.3 Installation Process

You must either create a new directory or have an empty one available. To create a new directory, from the root directory, type at the DOS prompt:

```
c: > mkdir cad
```

To change from the root directory to the CAD directory, type

```
c: > cd cad
```

Then, the following prompt will appear on screen,

```
c:\CAD >
```

When you are in the CAD directory, insert the PN/CAD backup diskette in the floppy disk drive slot and type:

```
c:\CAD > a:

a: > install
```

This will run the install.bat batch file which puts the PN/CAD System and the necessary files on the hard disk. As files are transferred the INSTALL command may display messages reporting its progress.

A.3 Sample Petri Net Editing Sessions

A.3.1 Login Procedure

To log onto the computer, first be sure that the terminal is ready to operate. You should also examine which peripheral devices are connected to the computer and must compare with the list of GKS supported devices, in order to load the correct device drivers. Section A.1 describes possible system configurations and how to load device drivers.

Assume a proper system configuration and installation of all PN/CAD files on the hard disk. Then turn on the computer. A series of informational and system messages will be displayed on the screen; finally the system prompt appears '*c* :> '. This is an indication that you are at the DOS enviroment and the computer is waiting for a command. Then, change to the directory where the PN/CAD system resides, i.e. CAD subdirectory.

A.3.2 Starting the Session

To start the PN/CAD system from the DOS prompt type the command:

```
c:\cad > pned
```

In a few seconds the first screen appears. The first PN/CAD screen, shown in Fig. A.1, consists of a Choice Menu. The System requests from you to provide the configuration and to activate the workstations. You are responsible for providing the proper configuration. If you do not, the System displays error messages. If the proper choice is made, the System displays the PN/CAD opening screen, see Fig. A.2. Pressing *Enter* puts you at Top Command Level and you are ready to start the Petri Net Editing Session, see Fig. A.3.

The following commands are available at Top Command Level:

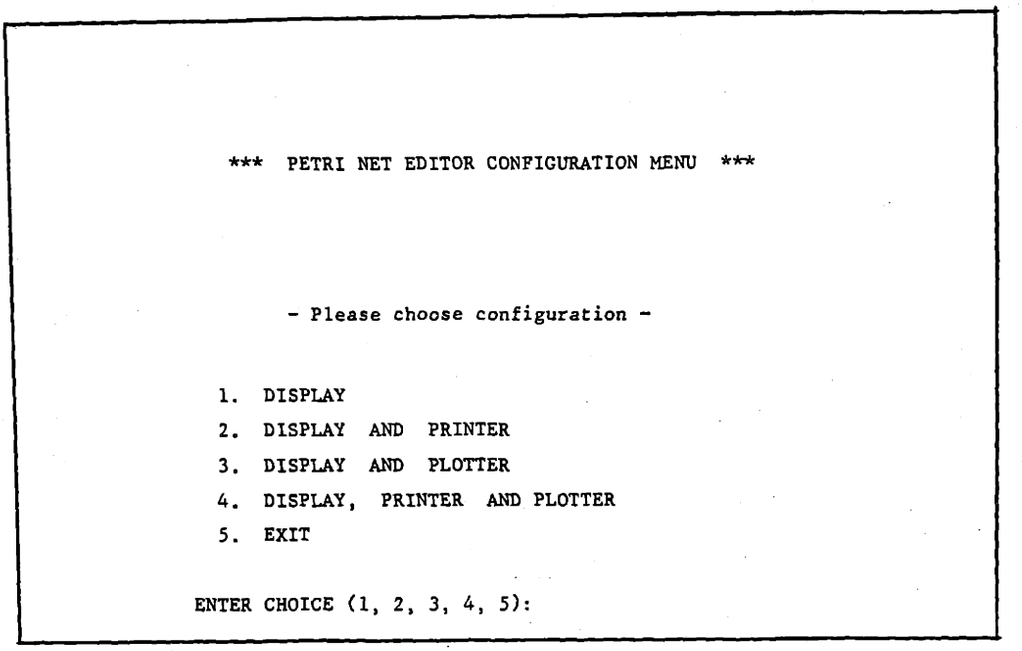


Figure A.1: PN/CAD Choice Menu.

Graphics Editor Allows you to generate a Petri Net graph

Text Editor Allows you to store and modify attributes represented by elements on a Petri Net graph.

Analysis Allows to create an analytical description of the Petri Net representation.

Hardcopy Allows you to produce a hardcopy on the printer or plotter.

Help Displays information on commands available at the current mode.

Exit Exits to DOS.

Moving the Cross Hair: To move the cross hair around you have to use the auxiliary keypad functions. The *Up, Down, Left* and *Down* arrow keys move the cross

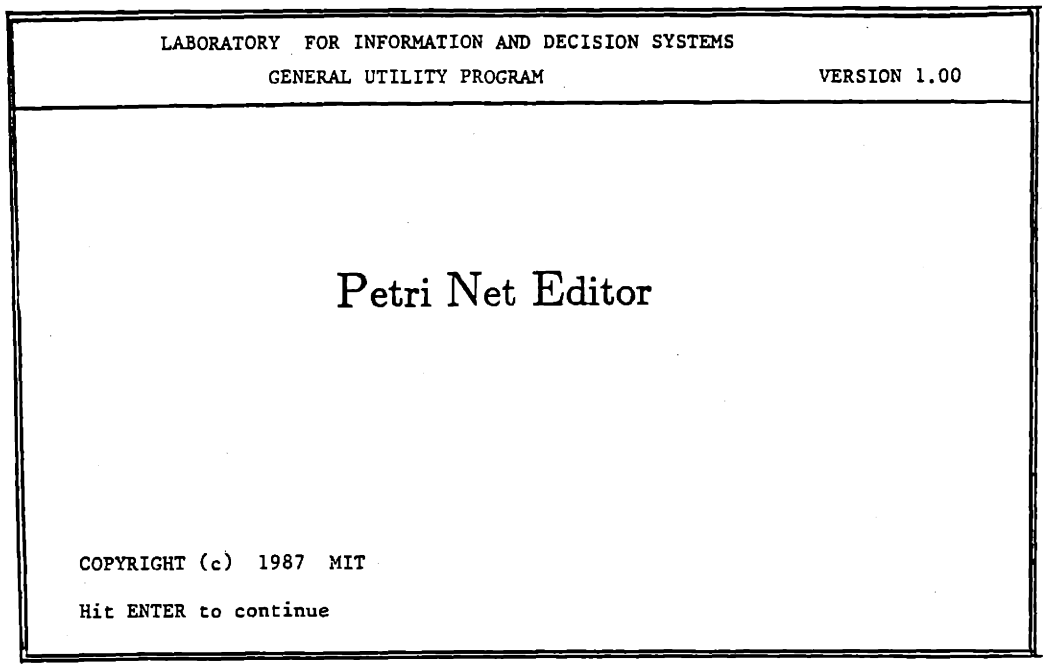


Figure A.2: PN/CAD opening screen.

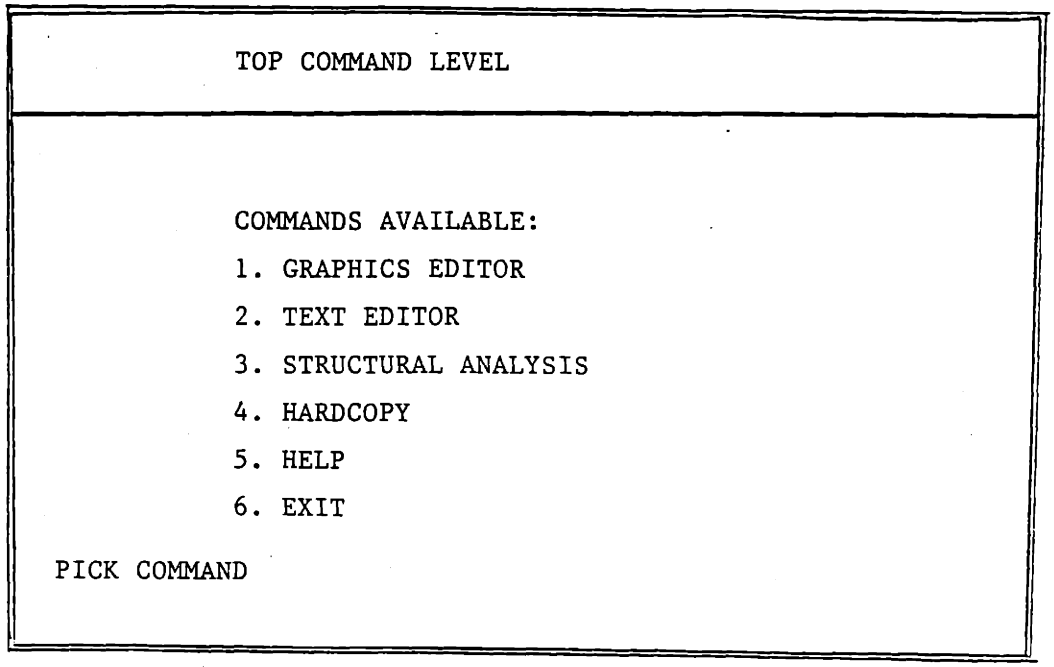


Figure A.3: Top Command Level Screen

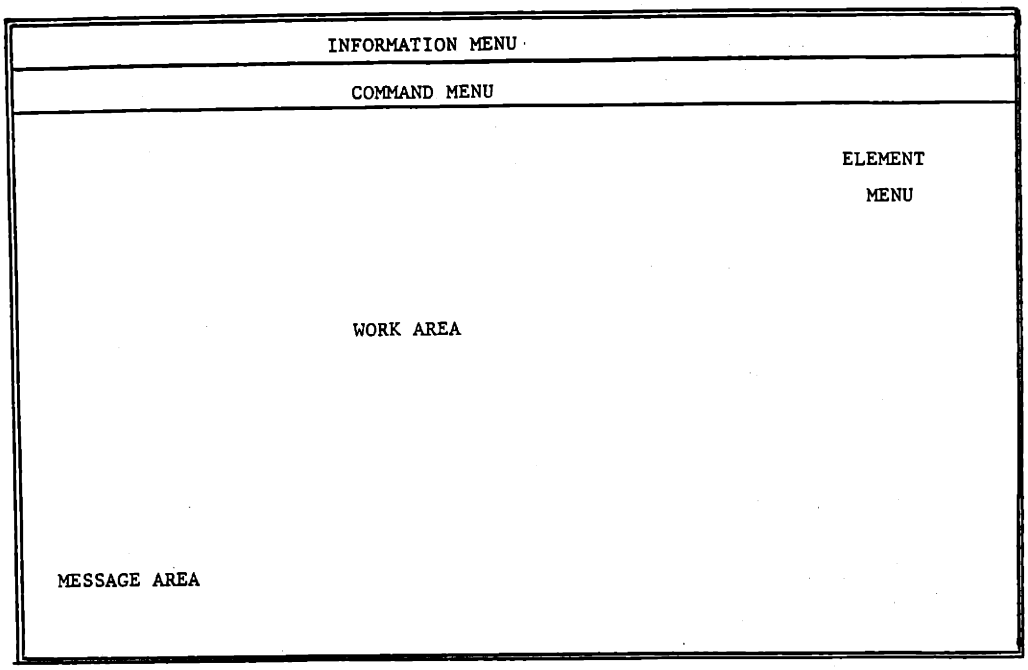


Figure A.4: Screen Organization

hair in vertical and horizontal directions. The *Home, PgDn, End* and *PgUp* keys move the cursor diagonally. The cursor moves, in any direction, 20 pixels per stroke. To get a finer resolution, 5 pixels per stroke, you have to press the *Ins* key and use the arrow keys again. To undo the mode, press the *Ins* key. Practice by moving the cross hair around.

Issuing Commands: To select a command you must move the cross hair, using the keyboard, and place it on top of the command. When *Enter* is pressed the command is highlighted, indicating that the command has been selected.

Screen Organization: The main part of the screen is used to display the contents of a buffer. The buffer contains either numeric data (translated into a graph by a program) or text, see Fig. A.4. At the top of the screen the Information Menu shows file attributes or indicates the main mode of operation. The Command Menu has a listing of the commands available at that mode. A Message Area is located at the bottom of the screen to display system messages, or to prompt you for a dialog. When you are at Level Two of the Graphics Editor Mode, the PN/CAD System provides you the Element Menu. The Element Menu contains the toolkit for constructing graphs.

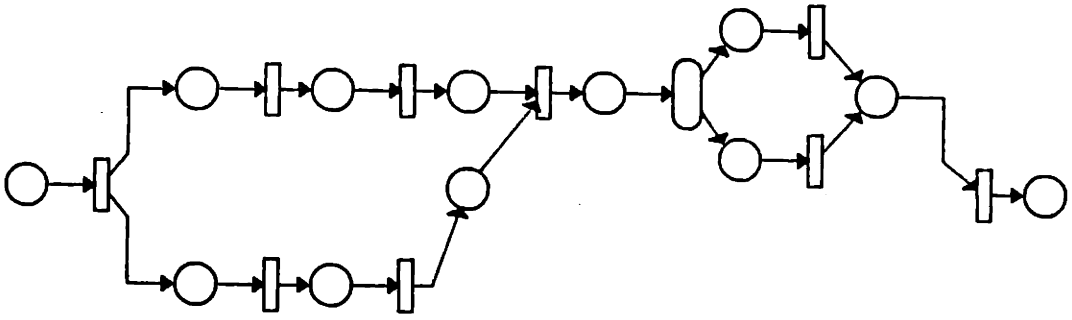


Figure A.5: Petri Net Graph

Using the Graphics Editor: Now, let us suppose that you want to construct the Petri Net graph shown in Fig. A.5. To do it, you must invoke the Graphics Editor. To invoke the Graphics Editor place the cross hair on top of GRAPHICS EDITOR text and press *Enter*. The Graphics Editor command is highlighted. The screen is updated and a window appears. Now, you are at Level One of the Graphics Editor. The following commands are available at Level One:

Show Displays the requested Petri Net graph. The numeric representation of the graph is stored in a buffer. Show prompts you to enter the name of the buffer.

Help Displays information about each command.

Exit Terminates the Graphics Edit session and returns control to Top Command Level.

Edit_File Invokes Level Two of the Editor for the creation or modification of graphics structures.

Show a Graphics File: To see what is inside a graphics file, invoke the SHOW command, type the file name DM1.GRF at the prompt and press *Enter*. There is a file

with such a name in your directory. The file is then displayed on the screen. If the file is not in the directory, SHOW displays a message informing you that the file is not in the filebase.

Using Help Command: The HELP command displays information on the commands available at that mode. For example, by invoking the HELP command at Level One of the Graphics Editor, the System displays the HELP text for SHOW, EXIT, and EDIT_FILE commands. To see the next HELP screen, press *Enter*.

EDIT_FILE: The EDIT_FILE command invokes Level Two of the Graphics Editor and starts an interactive graphics editing session. The screen is updated and a new set of commands is listed on the Command Menu. The Editor prompts you to enter the name of the file to be edited. If the file does not exist in the filebase a new file is created. A message in the Information Menu indicates the status of the file. If the file exists the Editor transfers the contents of the file you specify to temporary buffers and displays the graph on the screen.

A.3.3 A Sample Graphics Editing Session

The objective is to construct graphically the organizational architecture shown in Fig. A.5 . First, you should construct generic structures that may be combined to form the overall architecture. One such generic architecture has already been constructed and stored into DM1.GRF file. The second architecture, shown in Fig. A.6, is going to be constructed using the Graphics Editor. Enter the file name DM2.GRF at the file prompt, and press *Enter*. The Information Menu displays, the status of the file 'NEW', and the file name 'DM2.GRF'. You are ready to create the Petri Net graph. To draw a place, select its icon from the Element Menu, and press *Enter*. The icon is highlighted. Press *Enter* again and the cross hair is displayed at the center point of the screen. Move the cross hair onto the screen and put it at a desired location and press *Enter*. The place is drawn at that spot. To select the same or another element, place the cross hair on top of the element and press *Enter*. The connectors should be drawn last. After all other elements are drawn, see Fig. A.7, you select the connector

Figure A.6: Second generic architecture

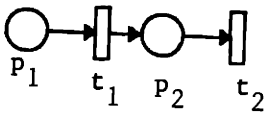


Figure A.7: Graph construction

element to connect the nodes. Place the cross hair on top of node P_1 (place, transition or switch) and press *Enter* . Then, move the cross hair around. A rubber band line moves with the cross hair to preview the line. Put the cross hair on top of node T_1 (place, transition or switch) and press *Enter*. A line is drawn with an arrow at its second end indicating a connection from node P_1 to node T_1. The graph is completed by making all the connections.

Ending a Graphics Editing Session: When you want to stop a graphics editing session, you invoke commands that either save your work from the editing session or delete what you have done. The edits that you make during a graphics editing session are stored in four numeric buffers. When you save or delete edits, you are really saving or deleting the contents of the numeric buffers. Therefore, when you create a new file, you decide whether or not to store the numeric buffers copy as a new file. When you work on a copy of an existing file, you decide whether to incorporate your edits into the file or delete your work from the editing session.

Exit: You can save the edits during a graphics editing session by invoking the EXIT command. The screen is updated and you are at Level One of the Graphics Editor.

Quit: You can delete all the edits you make during an editing session with the Quit command. QUIT command works with both the new and existing files. If you have

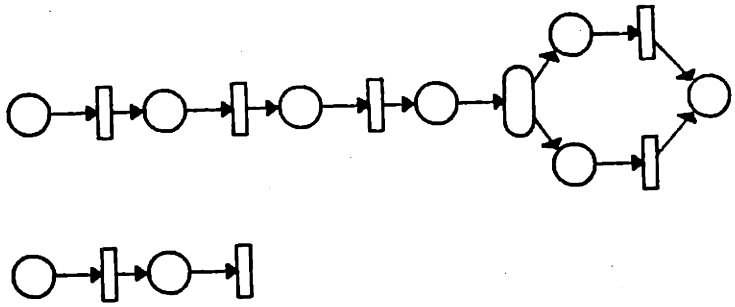


Figure A.8: Implementation INSERT command

started an editing session and you are dissatisfied with your edits, it may be easier to start a new editing session and use the QUIT command.

In the present example, invoke the EXIT command to save the graphic edits. The edits are stored in the file 'DM2.GRF'. To produce the final structure invoke the Graphics Editor once again by placing the cross hair on top of the EDIT_FILE command and pressing *Enter* . Create a new file 'DDM.GRF'.

Insert: Once you are in graphics edit mode again, invoke the INSERT command to insert the generic DM1.GRF and DM2.GRF files into your new edits. The INSERT command prompts you to enter the file name. Type one of the two files listed above, and press *Enter*. The INSERT command request from you to put the Net at a suitable location. Both structures are shown in Fig. A.8. When a structure is inserted the left most element of the inserted graph is placed at the location requested by the INSERT command. To complete the structure use the Element Menu commands.

Delete: In case you make a mistake you can use the DELETE command to delete any element. To delete an element, put the cross hair on top of the DELETE command

and press *Enter*. The DELETE text is highlighted and a message appears in the Message Area. Put the cross hair on top of the element to be deleted and press *Enter*. The element is deleted and the new graph is redrawn.

Copy: The COPY command copies a region from one location to the other within the numeric buffers. The region is not deleted from the original location. To copy a region, invoke the COPY command from the Command Menu and move the cross hair onto the screen surface. To select the region press *Enter*, and move the cross hair diagonally in any direction. A rubber band rectangle previews the shape of the selected region. Press *Enter* twice when the rectangle includes the graph you want to copy. The elements selected are those within the region. All elements crossing the boundaries of the region are excluded. Next, COPY requests the location to place the net. When you select the location, press *Enter* and the region is displayed.

Coarsen: Let us assume that you have created a new graphics file 'PAR.GRF' and you have inserted the contents of the file 'DDM.GRF' in it. Suppose you want to create a supernode of the structure enclosed by the dotted line. The COARSEN command allows you to do this. The COARSEN command prompts you to pick, using the keyboard, the input and output nodes of structure. At the start, COARSEN displays a message and prompts for the input nodes, i.e. P_4. Put the cross hair on top of each input node and press *Enter*. Each input node is highlighted. After all input nodes are selected, place the cross hair on top of the END text. Selecting END signals that all input modes are selected. Then COARSEN requests you to select all output nodes, i.e. P_7. Invoking END once again signals that all output nodes are selected. COARSEN requests you to use the rubber band rectangle, in a similar fashion as in the COPY command, to specify the region, see Fig. A.9. All input and output nodes must be of the same type, in this example places P_4 and P_7. A set of algorithms checks if the proposed structure may be classified as a supernode. In this example, the structure is classified as a supertransition. The screen is updated and a supertransition appears in place of the selected subnet, see Fig. A.10.

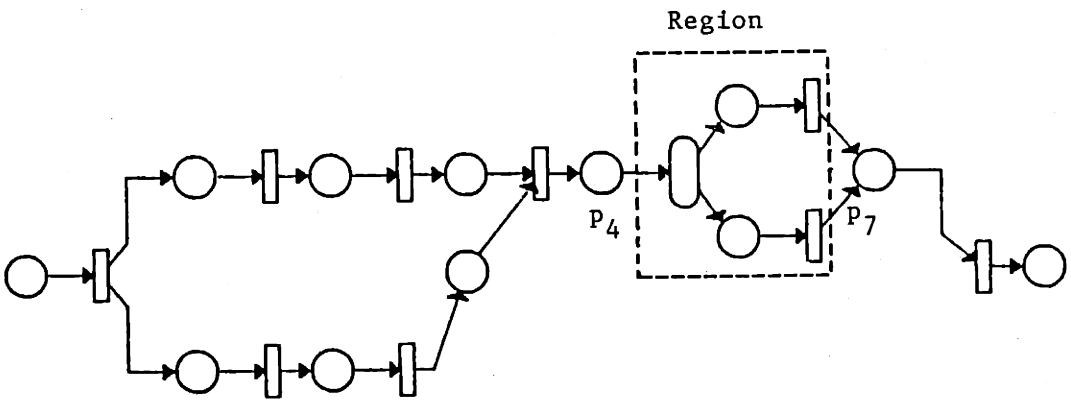


Figure A.9: Coarsen Region

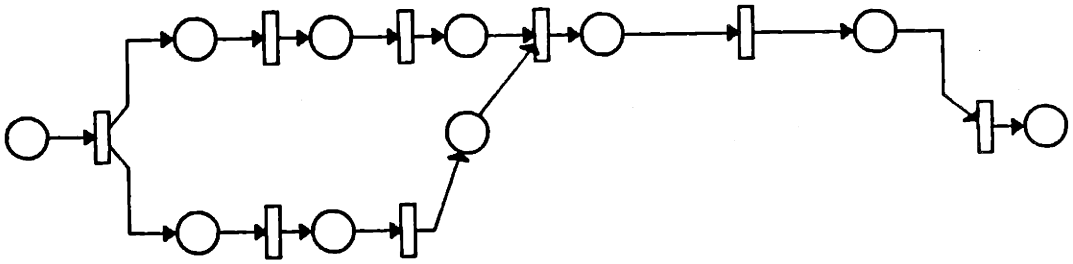


Figure A.10: Aggregated version of the structure

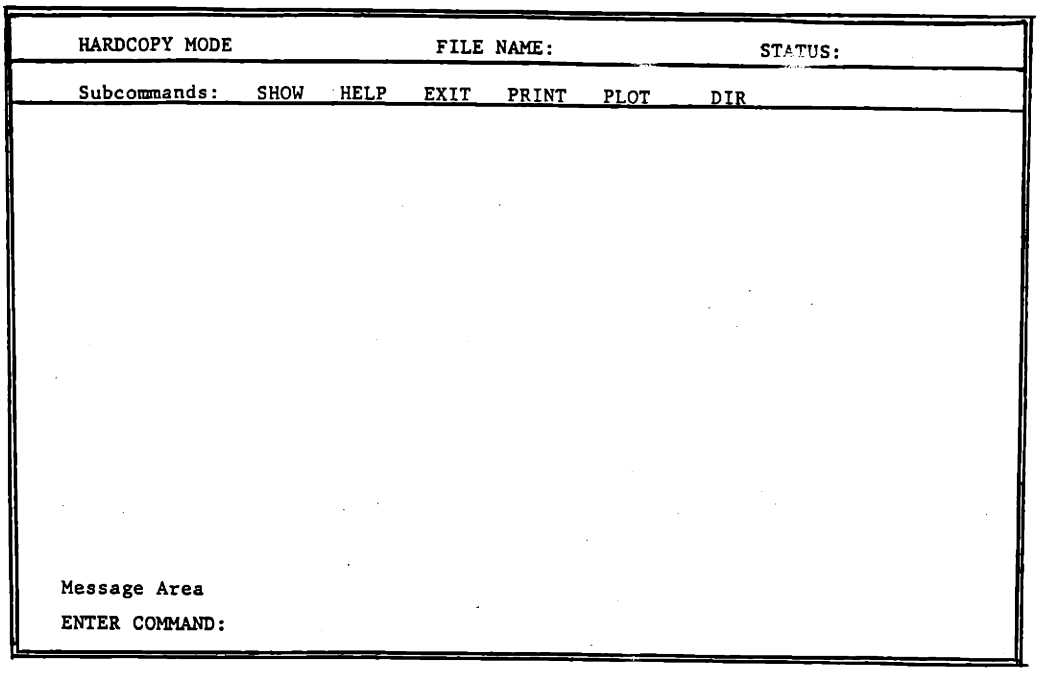


Figure A.11: Hardcopy Opening Screen

A.3.4 Hardcopy Function Session

From the Top Command Level put the cross hair on top of the Hardcopy text and press *Enter*. The command is highlighted and the screen is updated. You are placed in Hardcopy Mode. There are five commands available at that mode. The Show command displays the Petri Net graph stored in a .GRF file. The Help command displays information for each Hardcopy command. The Exit command returns control to Top Command Level. The Plot or Print command sends the requested file to the respective output device. When you invoke the PRINT or the PLOT command the System requests you to enter the name of the file. Only files having .GRF or .SIV as file types are accepted. The file is displayed on the screen before it is printed or plotted. The Hardcopy opening screen is shown in Fig. A.11. The System requests from you a Yes or No reply, indicating whether the flow path displayed on the screen is to be printed or plotted.

A.3.5 Structural Analysis Session

To enter, from Top Command Mode, the Structural Analysis Mode put the cross hair on top of Structural Analysis text and press *Enter*. The Structural Analysis opening screen is shown in A.12. The Show command displays the Petri Net graphs stored in .GRF files. The DIR command provides a directory listing of files. The DIR request you to enter file specifications. If you press *Enter* all .GRF files are listed. To see all .TXT files type,

```
ENTER FILE-SPEC:*.txt
```

To generate the interconnection matrix invoke the MATRIX command. The System requests the file name containing the Petri Net graph. Files with .GRF as file type are accepted. The interconnection matrix of the PAR.GRF file is displayed on the screen. The interconnection matrix is stored in the file PAR.MTR while the incidence matrix is stored in the file PAR.INC. To generate the information flow paths of the Petri Net graph stored in the PAR.GRF you have to invoke the S_INV command. The S_INV requests you to enter respective matrix file, PAR.MTR. The information flow paths are generated and are displayed on screen upon your request, see Fig. A.13

A.3.6 A Text Editing Session

The text editor is an interactive utility program which allows you to create or recall from the file library a text file, to enter or to modify its contents, and to save or to delete the work you have done during the text editing session. The text file contains information for all the elements of the Petri Net structure. The Text editor works only with text files that contain specific text editing and formatting capabilities.

The purpose of the Text editor is to allow you to store information represented by places, transitions, switches or connectors in the graphics structure. In some cases, the information may be used for syntactical or semantic purposes, while in other cases for simulation or performance evaluation of systems. In general, the stored information

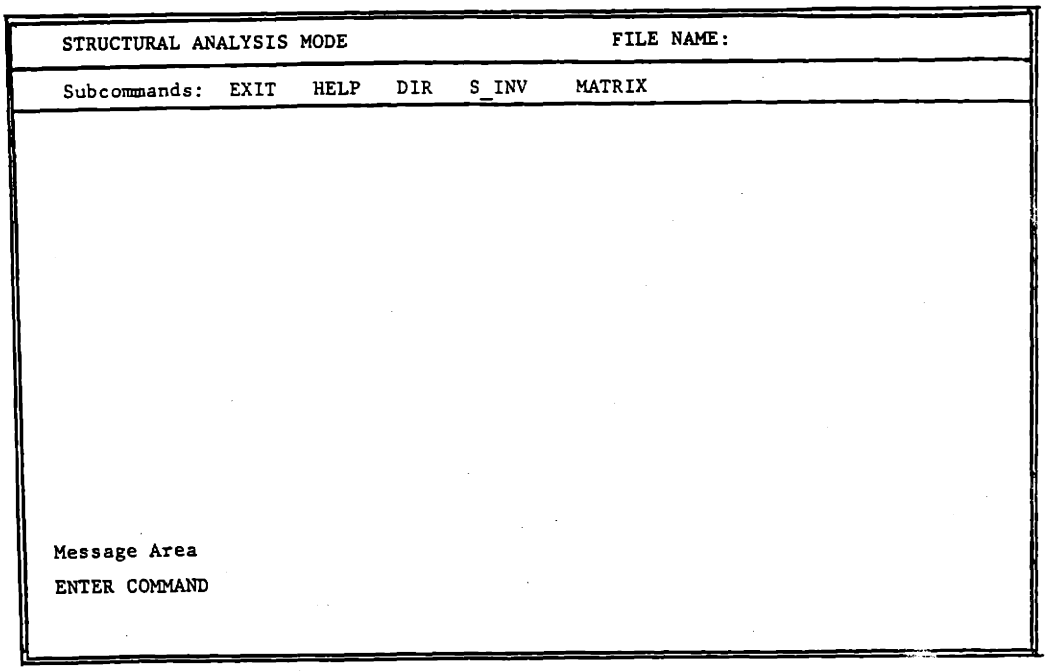


Figure A.12: Structural Analysis Mode opening screen

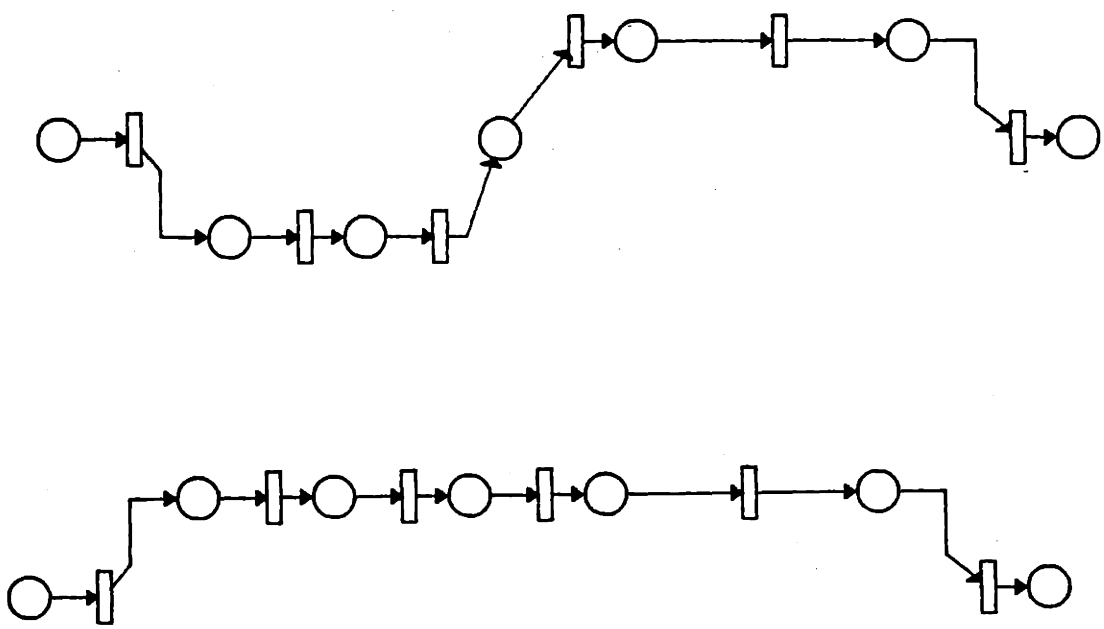


Figure A.13: Information flow paths

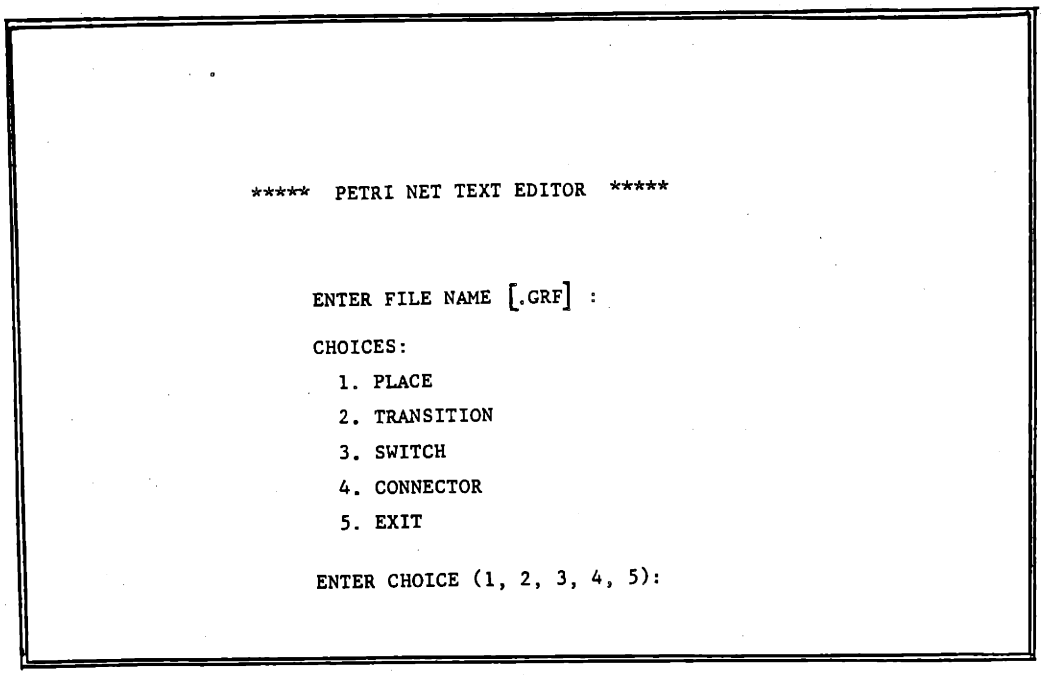


Figure A.14: Text Editor opening screen

is retrieved and used for analysis and design of complex organizational architectures. The Petri Net Text editor offers specific features that make text editing easier and the information stored more accessible. These features include:

- An on-line HELP facility.
- Menu driven actions.
- Interactive information access in the form of queries.
- Display of stored information.
- Node record editing.
- Internal command parser.

For the operation to take place, you have to be in Text Mode. When in Text Mode, you can recall from the library or create the files containing the text material. The Text Editor is invoked from Top Command Level. The Text Editor opening screen is shown in Fig. A.14. A short description of the features of the Text Edit Mode follows:

The HELP facility provides descriptive messages about menu choices, ways to implement them, and how to construct text edit records of the elements. The Text Menu allows you to examine the choices you have available without choosing any. If the user requests so, the application program calls the functions implementing the command and performs the action. The text menu consists of a number of menu items listed vertically in the middle of the screen. The Menu items (or list of text mode commands) are:

- modify a text file
- display a text file
- node record editing mode
- exit

The Modify command allows you to enter new information or modify the old one. You may modify the text records sequentially, according to the record number assigned by the numbering routine during the graphical creation of the Net. The Display command displays the stored information for visual examination. The Node Number line editing mode is useful for those that prefer random access of the records. Each record can be accessed by specifying the element type (place, switch, transition or connector) and its corresponding assigned logical number. The Exit command terminates the session and exits to higher level.

There are four direct access files created or accessed for each Petri Net graph. These files are used to store information about places, transitions, switches and connectors of a Petri Net structure. Each record of a file may contain information relative to capacity, algorithms/procedures/functions, probabilities, logical or mathematical expressions, comments, existence of subnets or any other information associated with each of the above four elements. As the name of the file indicates, the file organization is of the direct access type as opposed to sequential.

The program uses the direct access mode to retrieve and store text records in the file. A text record consists of data fields, where information is stored, and a record

Data Code - Meaning	*Example*
C - capacity	C:3
P - probability	P:0.7
F - function	F:g2
L - boolean expression	L:'a=b'
T - text	T:undecided
D - delay	D:0.5
ST - Supertransition	ST:'T15.TRN'

Table A.1: VALID DATA CODES

number indicating the record's position in the file. Consequently, a record in the file can be accessed either at random by specifying its record number, or sequentially by going through all previous records.

Each particular information entry is called a data object and occupies a data field. Each data object is specified by its type code and data. It has the format:

```
tc:data
```

where:

`tc`	`type code`
`data`	`data, alphanumeric, real, integer or logical`

Each data object is preceded by its type code followed by a colon (:), followed by its data. A text record may contain any number of data objects. These data objects must be separated by a semicolon (;). As an example,

```
tc1:dt1;tc2:dt2;tc3:dt3;
```

Type codes may be entered using either upper or lower case lettering. A semicolon after the last data object is optional. A short list of valid type codes is given in Table A.1, below:

The information stored in the record can be parsed by a command parser. The command parser exploits the structural breakdown of the data object and interprets the contents of the fields. The algorithmic implementation of the parser targets on the semicolon delimiter and extracts the alphanumeric characters between the semicolons. It then passes control to the command interpreter. The interpreter checks the parsed information against valid type codes and data and subsequently interprets.

A.4 PN/CAD Command Description

This section contains entries for each of the PN/CAD System commands. Each command's purpose, graphic inquiry and options are described. Information about each command can be found on-line via the HELP command.

The following convention is used in the command description and implementation:

typing Lower-case typing should be used always. The System differentiates between lower and upper case lettering.

[ ] Square brackets around an argument indicate the argument is optional.

file-spec The name of the file to be processed.

All the commands are identified graphically by the Inquire Current Pick Identifier routine. All options and file specifications are entered through the keyboard. When a file extension is not entered, the default extension for that mode is assumed. Fig. A.15 shows the command structure.

ANALYSIS

Invokes the analysis mode for structural analysis of a Petri Net. It is implemented at Top Command Level.

Graphics Inquiry:

```
STRUCTURAL ANALYSIS
```

Messages:

ANALYSIS MODE ENABLED	The structural analysis mode has been invoked succesfully.

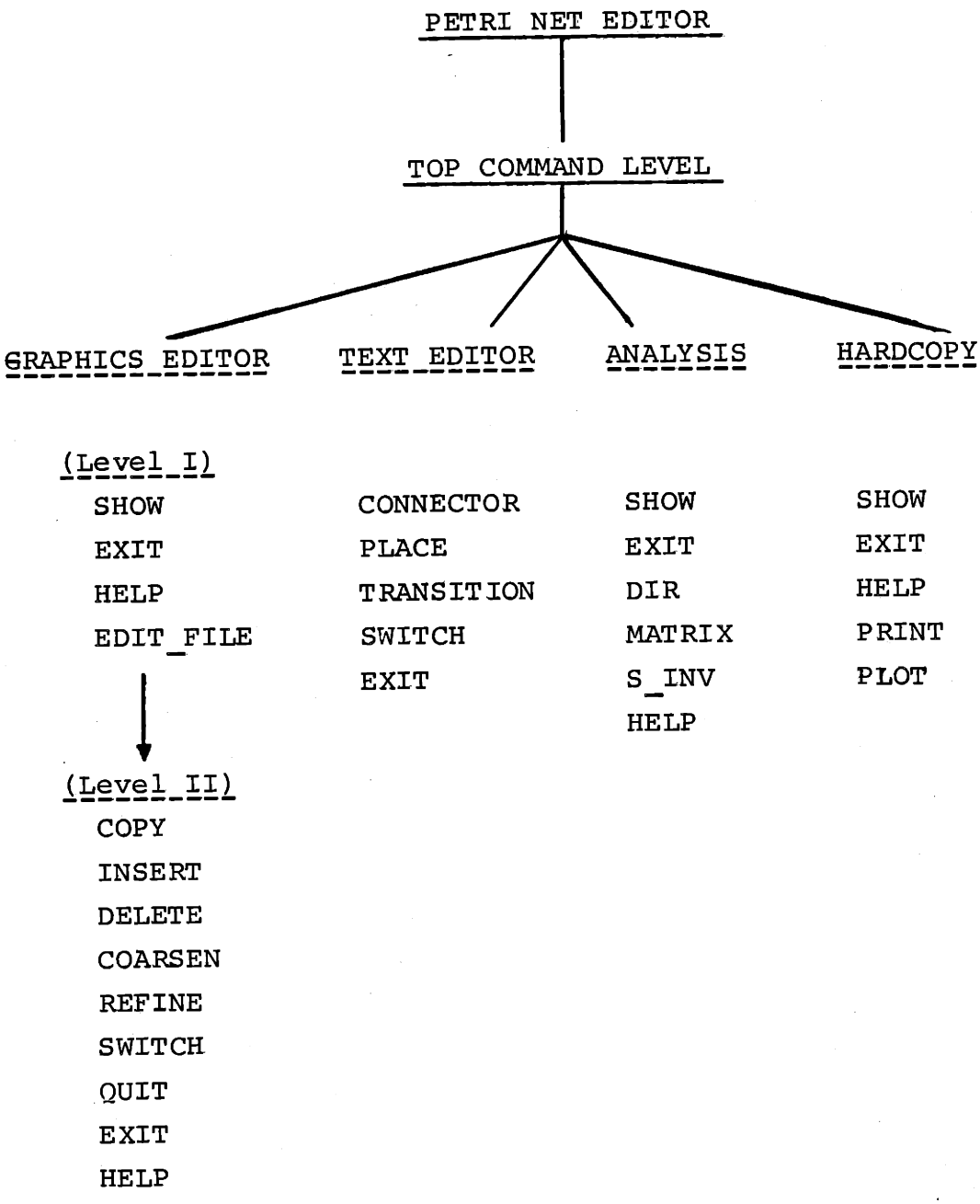


Figure A.15: Command Tree Structure

COARSEN

This command simplifies a Petri Net structure and allows the creation of supernode structures. It is implemented within the Graphics (Level Two) Editor.

Graphics Inquiry:

```
COARSEN
```

This command allows the formation of an aggregate (group) that leads to Petri Net simplification. It also may be viewed as a Petri Net reduction technique that allows the creation of supernodes. The Petri Net structure of the supernode is specified by its input and output nodes and by all the elements included in a rectangular frame. Once a supernode is specified the overall structure is substituted by the supernode.

Messages and Prompts:

```
COARSEN MODE ON
```

This message indicates that the COARSEN command has been invoked succesfully and asks the user to must specify the input and output nodes and the region's boundary. A set of procedures check if the Petri Net reduction rules are fulfilled. Otherwise, the message

```
REDUCTION RULES VIOLATION
```

appears and the coarsen mode is turned off. Only two levels of reduction are allowed for an original Petri Net structure.

COPY

Copies a region from the Petri Net graph and places it at a user's specified location. It is implemented within the Graphics (Level Two) Editor.

Graphics Inquiry:

```
COPY
```

This command creates a duplicate Petri Net structure of a region and puts it at a specified location. The contents of the region are all the elements within the bounds of a rectangular frame. The region includes all places, transitions and switches which have their geometric center inside the boundary and all connectors which have their starting and terminal point within.

Messages and Prompts:

```
COPY MODE ON, SPECIFY BOUNDS
```

This message indicates that the COPY command has been invoked succesfully and asks the user to specify the region's boundary.

```
LOCATE REGION
```

The COPY command requests from the user to specify the new location of the copied region. A set of procedures check for overlap. If there is overlap the message

```
PETRI NET OVERLAP
```

appears and the COPY mode is turned off.

DELETE

This commands deletes the picked graphics element or all the elements included within a specified region. It is implemented within the Graphics (Level Two) Editor.

Graphics Inquiry:

```
DELETE
```

Prompt:

```
DELETE MODE ON, ENTER OPTION:
```

The message indicates that the delete command has been invoked succesfully, and requests from the user to enter an option.

Options:

-e deletes the picked element. The picked element is highlighted.

-r deletes all the elements within a region. The region's boundaries are specified by a rectangular frame. All elements within the frame are deleted.

The default is to delete a single element from the graphics structure. Once an element is deleted the graphics structure is updated.

DIR

Provides a list of files or information about a file. It is implemented within the Top Command Level and within the Analysis Mode.

Graphics Inquiry:

```
DIR
```

Prompts:

```
ENTER FILE-SPEC: [file-spec]
```

where file-spec specifies the names of one or more files to be listed. The syntax of the file specification determines what file(s) should be listed:

- if you do not enter a file specification, the DIR command lists all .GRF files,
- The wild card identifier (*) may be used in any part of the file specification, when a user invokes the DIR command.

Example:

ENTER FILE-SPEC: *.mtr lists all existing files having .MTR as file extension.

EDIT_FILE

Invokes the Graghics (Level Two) Editor to create or modify a graphics file. It is implemented within the Graphics (Level One) Mode.

Graphics Inquiry:

```
EDIT_FILE
```

EDIT_FILE allows you to enter the main graphics editor to create and modify graphics files. The graphics files accepted must have .GRF as file extensions. The EDIT_FILE command initiates an interactive graphics editing session in which you can edit graphics files.

Prompt:

```
ENTER FILE NAME: file-spec
```

where file-spec is the name of the file to be edited. The file name of the file to be edited is specified by the user. Once a file is specified, it has a status OLD or NEW. The file name and status are displayed in the File Information Area.

EXIT

Terminates processing of the current mode of operation. It exists in all four Functional Levels.

Graphics Inquiry:

```
EXIT
```

The execution of this command terminates the current mode of operation and returns control to the higher level. If the command is executed from within Analysis, Hardcopy, Text or Graphics (Level One) mode, control is returned to Top Command Level. If the command is executed from the Top Command Level, control is returned to the DOS operating enviroment. If the command is executed from the Graphics (Level Two) mode, the graphics session is terminated, all temporary files are deleted while all new and re-edited files are saved and stored.

GRAPHICS EDITOR

Invokes the Graphics Editor for the graphic creation and modification of a Petri Net. It is implemented at Top Command Level.

Graphics Inquiry:

```
GRAPHICS EDITOR
```

Messages:

GRAPHICS MODE ENABLED	The Graphics Editor has been invoked succesfully.

HARDCOPY

Invokes the Hardcopy Facility. It is used to generate a hardcopy on the plotter or printer of the graphical image of the Petri Net. It is implemented at Top Command Level.

Graphics Inquiry:

```
HARDCOPY
```

Messages:

HARDCOPY MODE ENABLED	The hardcopy facility has been invoked succesfully.

HELP

Invokes the HELP facility to display information about the commands available and information about the use of each command. All help files are stored in the system's HELP library. Additional help files may be added to expand the set.

Graphics Inquiry:

```
HELP
```

The requested information is displayed on the default output device (monitor) one screen at a time. Pressing *Enter* will advance you to the next screen. At the bottom of the page HELP displays a message informing you if there is more information to be displayed or not. The Help facility is available at all modes, displaying information for the use of the commands available at that mode. The Help facility exists in all four Functional Levels.

INSERT

This command allows you to merge graphics files. It is implemented within the Graphics (Level Two) Editor.

Graphics Inquiry:

```
INSERT
```

Prompt:

```
ENTER FILE NAME: file-spec
```

where file-spec is the graphics file containing the Petri Net structure. Only files with extension .GRF are accepted. Once a valid entry has been typed, the system prompts the user with a message,

```
INSERT MODE ON, LOCATE NET
```

that is, the INSERT mode has been invoked and the system asks the user to place the Petri Net structure on the screen. A cross-hair cursor denotes the position. Once the location is picked, a set of procedures check if the inserted structure overlaps with the existing one. If it does, the following message is displayed,

Message:

```
PETRI NET OVERLAP
```

and the insert mode is turned off.

MATRIX

It calculates the interconnection and incidence matrix of a Petri Net. It is implemented within the Analysis Mode.

Graphics Inquiry:

```
MATRIX
```

This command determines the interconnection matrix of a Petri Net. Once the matrix has been produced, it is displayed on the screen and it is stored in a file with the same file name as the original file but with extension .GRF. The same algorithm calculates the incidence matrix and stores the results in a file with extension .INC.

Prompt:

```
ENTER FILE NAME: file-spec
```

where file-spec is the name of the file containing the Petri Net structure. The MATRIX command displays the prompt message requesting a reply. Only file specifications with file extension .GRF are accepted for processing. MATRIX reads the user's reply as a command and interprets the response. For a valid file entry, it scans the file library and requests the file.

Options:

`-i`	`Display incidence matrix`
`-c`	`Display interconnection matrix`

The default is to display the interconnection matrix.

Messages:

FILE DOES NOT EXIST IN THE LIBRARY The requested file does not exist.

PRINT or PLOT

Queues a file for printing/plotting, on a specified system printer or plotter. It is implemented within the Hardcopy Mode.

Graphics Inquiry:

```
PRINT or PLOT
```

Prompt:

```
ENTER FILE NAME: file-spec
```

where file-spec specifies the name of the file to be printed/plotted. Only files with file extension .GRF and .SIV are accepted. The PRINT/PLOT command displays in the Message Area a prompt message requesting interactively the name of the file whose contents are to be printed/plotted on the default system printer/plotter. The PRINT/PLOT command reads the response from the terminal as a command and interprets the response accordingly. It is assumed that the appropriate device drivers are loaded and the correct choice has been made during system configuration.

Messages:

FILE DOES NOT EXIST IN THE LIBRARY	The requested file does not exist.
FILE NOT PRINTED/PLOTTED	Drivers are not loaded or incorrect system configuration.

QUIT

This command ends the current editing session without saving the contents of the main buffer. It is implemented within the Graphics (Level Two) Mode.

Graphic Inquiry:

```
QUIT
```

This command returns you to the Graphics Level One Mode and saves nothing from the current session.

Prompts:

```
Do you want to delete the contents? (y/n):
```

The QUIT command asks you if you have second thoughts about deleting the contents of the buffer.

REFINE

This command augments a Petri Net structure and allows the decomposition of supernode structures. It is implemented within the Graphics (Level Two) Editor.

Graphics Inquiry:

```
REFINE
```

This command allows the decomposition of an aggregate (group). It also may be viewed as a Petri Net augmentation technique.

Prompts:

```
REFINE MODE ON, PICK NODE
```

This message indicates that the REFINE command has been invoked succesfully and asks the user to specify the supernode. A set of procedures check if the picked element is a supernode or not. If it is, it prompts

```
AUGMENT (Y/N)? y/n
```

and asks if the user wants to augment the current Petri Net structure. A set of procedures check if the syntax is correct.

Messages:

If the picked element is not a supernode, the message

```
NO SUPERNODE
```

is displayed and the REFINE mode is disabled.

SHOW

Displays on the default output device (screen) the contents of the graphical file. Implemented in all Functional Levels.

Graphics Inquiry:

```
SHOW
```

The requested file is displayed graphically on the screen and allows pre-examination of the contents of the file. It is implemented in all modes.

Prompts:

```
ENTER FILE NAME: file-spec
```

where file-spec specifies the name of the file to be showed. Only files with the extension .GRF and .SIG are accepted. You cannot use the wild card character in the file specification during the implementation of this command.

The SHOW command displays in the Message Area a prompting message requesting interactively the name of the file whose contents are to be displayed on the terminal screen. The SHOW command reads the response from the terminal as a command and interprets the response accordingly.

Messages:

FILE DOES NOT EXIST IN THE LIBRARY	The requested file does not exist.
FILE EXTENSION INCOMPATIBLE	The requested file does not have .GRF or .SIG as file extensions.

S_INV

It determines the minimal supports of S-invariants of a Petri Net and displays both the matrix and the data flow paths graphically, one at a time, on screen. It is implemented within the Analysis Mode.

Graphics Inquiry:

```
S_INV
```

This command determines the minimal supports of S-invariants of a Petri Net. Once the matrix has been produced, it is displayed on the screen.

Prompt:

```
ENTER FILE NAME: file-spec
```

where file-spec is the name of the file containing the Petri Net interconnection matrix. The S_INV command displays the prompt message requesting a reply. Only file specifications with file extension .MTR and .INC are accepted for processing. S-INVARIANTS reads the user's reply as a command and interprets the response. For a valid file entry it scans the file library and requests the file.

```
DISPLAY FLOW PATHS (Y/N): y/n
```

The command also prompts the designer for the graphical display of the data flow paths on the monitor. The information is stored in a graphics file which has the same name as the original file but with extension .SIG.

Messages:

FILE DOES NOT EXIST IN THE LIBRARY The requested file does not exist.

SWITCH

Selects a flow path from a set of possible flow paths. It is implemented in Graphics Editor Level Two mode.

Graphic Inquiry:

```
SWITCH
```

Prompt:

```
Enter connector number (12, 15, 16):
```

It requests from the user to enter the connector number to be associated with the activated path. Once the path is activated all other paths are shut off and the net is reduced.

TEXT EDITOR

Invokes the Text Editor. It is used for insertion of attribute information about the Petri Net. It is implemented at Top Command Level.

Graphics Inquiry:

```
TEXT EDITOR
```

Messages:

TEXT MODE ENABLED The Text Editor has been invoked succesfully.

A.5 The PN/CAD Programming Environment

A.5.1 Introduction

In order to use the PN/CAD package you should have access to a specific IBM 5170 PC AT or XT model or any compatible system with an 1.2 Mb high density diskette drive. Then you must make sure there is enough disk space available to load the system and to accommodate some special PN/CAD files and any other files created during the session.

A.5.2 System Commands

When you enter the PN/CAD programming environment you receive a series of information messages, then a set of queries explaining to you what you must do in order to continue. The messages prompt you to enter an option reflecting your system configuration. Then, according to the option entered, the proper workstations will be activated. If the proper option has been chosen the PN/CAD screen will appear with a cross-hair positioned at the center of the screen. This is an indication that the software has been invoked succesfully and you are placed within the PN/CAD enviroment. The cross-hair indicates that the system is waiting to receive a command.

The Petri Net graphics command interpreter reads the commands selected by the user and interprets them as requests to run programs. Each graphics command is the name of the highest level module. There are more than 20 different graphics or text commands. Each command may have one or several options. A typical graphics command might be

```
show
```

This tells the command interpreter to invoke a special program which can display the contents of the graphics file. For this reason dialog messages appear on the screen when the system needs more information to carry out a command. In this case, the

system will request the name of the file. The command interpeter searches the current directory for the specified file. If the file exists, the contents of it will be displayed on the terminal screen.

A set of graphic commands are:

```
show      displays the contents of a graphics file
dir       provides a list of files
insert    allows for the merge of graphic files
copy      copies a specified region from the Petri Net structure
delete    deletes one or more graphic elements
coarsen   creates a supernode
refine    decomposes a supernode
```

These commands and many more were described in section A.4.

The command interpreter is case sensitive; this means that if you type an option in upper case, the command will not be interpreted properly and you will get an error message. All replies must be entered entirely in lower case.

A.5.3 File Specifications

A file is identified by its file specification. A file specification should be less than 15 characters long and be composed of two parts: the file name, a period, and the file type. The format is:

filename.typ

where "filename" is the name of the file. The file name must not start with a number or any special character. The file type is 3 characters long and is used to classify the contents of the file. The following are valid file names:

```
pdm3.mtr      A file containing the incidence matrix.
```

```
c3i.plc      A file containing attribute information for places.
ser.inc      A file containing the incidence matrix.
org.siv      A file containing the minimal supports matrix.
```

Files that contain graphics structures must have .grf, .sig, .t## or p## (where ## means a two digit number) as file types in order for the interpreter to correctly evaluate the contents. Some examples of file specifications are:

```
pdm3.grf     A file containing a graphics structure.
sdm4.sig     A file containing a graphics structure of the
             information flow path.
c3i.t12      A file containing a graphics structure of a
             supertransition.
```

The File Manager of the Petri Net system does not provide a way of storing old versions of files. If you update an existing file, then the older file is saved by DOS but with .BAK as an extension. In case you need the older version, you have to get back to the operating system level, rename the file and make sure that its name has been entered into the directory lists. In general, if you want to have more than one copy of a file, you have to create a new file with a different name and insert the contents of that file into the new one. When a new graphics file is created the File Manager creates all the necessary text and temporary files.

A.5.4 The Directory Command and Wild Cards

The PN/CAD system has its own directory system managed by the File Manager. There are four different directory files. Each directory contains a list of files with their date of creation. The directory groups files according to their type.

The Petri Net command interpreter permits the use of a wild card character in file name specification during the execution of the directory command. The wild card character is the asterisk, and it matches any string of characters. As an example, if

you had files *file1.grf, file2.grf* and, *file3.grf* in the current directory, and invoked the directory command,

```
ENTER FILE-SPEC:*.grf
```

at the prompt, the system will list all files with *.grf* as file type.

A.5.5 Keyboard Layout and Use

This section describes the layout of an IBM AT keyboard and provides to the user information about the available keys during an interactive PN/CAD session. When using the IBM system, the designer uses either the Enhanced Color Display or Professional Graphics Display video terminal. In their graphics mode, in addition to the normal character set, each terminal understands some special commands which cause it to move the cursor around on the screen or do other special functions. Each graphic package has a set of routines available and includes special command sequences which enhance the functional capabilities of the keyboard.

A.5.6 Special Key Definitions

The IBM AT keyboard layout is shown in Fig. A.16. It is divided into three parts:

1. the typewriter area, which has the alphanumeric characters found on a normal typewriter keyboard,
2. the numeric keypad, on the right, used for the graphics cursor movement and,
3. the function keys, on the left, which perform assigned functions when in graphics mode.

The same keys appear on all other types of keyboards, but they may not appear at the same location. A short description of selected keys is given below:

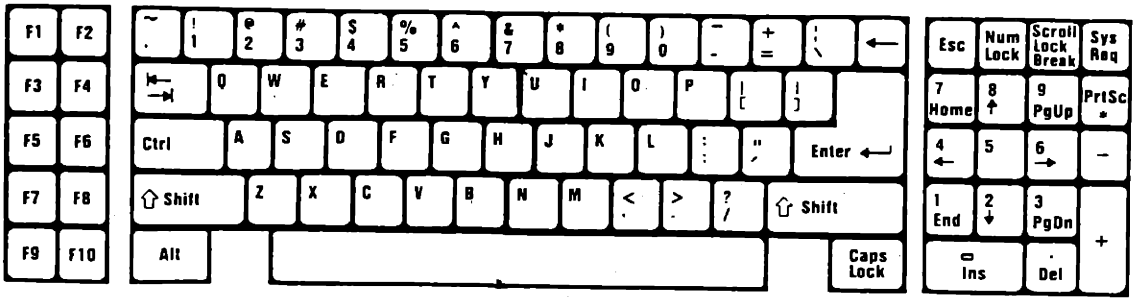


Figure A.16: IBM PC AT keyboard layout.

ENTER Sends a carriage return code to the computer. It is used to terminate.

DELETE Deletes the character last typed, so you can retype it in case you made a mistake.

ESC Allows you to send a non standard set of commands to non supported devices.

INS Allows you to move the cross hair 5 or 20 pixels per stroke.

A.6 Graphics Software Description

This section describes a number of important issues the system programmer and designer should know in order understand the graphic capabilities of the Petri Net software package. At first, a short description of the structure of the graphic software is given. Then, a summary and overview of the IBM Professional Graphics Software is provided. The use of the Virtual Device Interface Controller is also discussed.

A.6.1 Graphics Software Issues

The issues involved in the development and installation of the Petri Net Computer Aided Design package are general and can be applied to any graphics application program under development. The basic elements of a graphics application program are:

- the graphics software development package
- language binding
- device independent controller
- device drivers

The IBM Professional Graphics Software (IBM PGS) provides a set of graphics software development tools intended for use in engineering applications. The graphics package is a set of modules that can be used within an application program to manage graphical input and output and device control operations. All of the IBM PGS components are device independent, that is you can send the application program output to any supported input or output graphics device, without having to modify the graphics application. The basic components of the IBM PGS are:

- Graphical Kernel System (GKS), which is an implementation of the proposed ISO/ANSI standard level supporting segmentation features. GKS segmentation

is a collection of graphic images that are manipulated as a single unit. Transformation, scaling, rotation and highlighting are among the graphics functions available from GKS. GKS supports two dimensional graphics images and five input functions, the locator, string, pick, choice and valuator. GKS based application programs are portable among systems that support the proposed standard and are device independent.

- Graphics Development Toolkit (GDT), which is an implementation of the proposed ANSI $X3H33$ standard. It provides the means for writing device indepedent graphic software and supports new graphics hardware devices without modification to the program. It also provides a set of graphic and text functions including raster and input operations.

- Plotting System Library (PSL), which provides a set of 2-D graphics routines for creating professional quality charts, such as pie charts, bar charts, schedules, etc. It is device independent.

- Graphical File System (GFS), is a device indepedent tool used for the storage, retrieval and manipulation of graphic images. It provides a programming and an interactive interface for system programmers.

- Graphics Terminal Emulator (GTE), provides the means to interface with host computer. This program allows the IBM PC to be attached to a mainframe computer as a remote terminal and emulate either the Tektronix 4010 Series terminals or the Lear Siegler ADM 3A terminal. This program allows the IBC PC or any compatible personal computer to make use of the applications programs already developed on the mainframe computer.

The language interfaces of the GKS, GDT and PSL include the IBM Professional Fortran, Pascal Compiler 2.00 (except GKS), Lattice C 2.0 developed by Lattice, Inc. and the IBM Basic Compiler 1.00 .

In order for the graphics application program to operate the supported input or output devices in their graphical mode, it must communicate with them through a set of graphic device drivers. Each specific device has its own device driver. The device driver

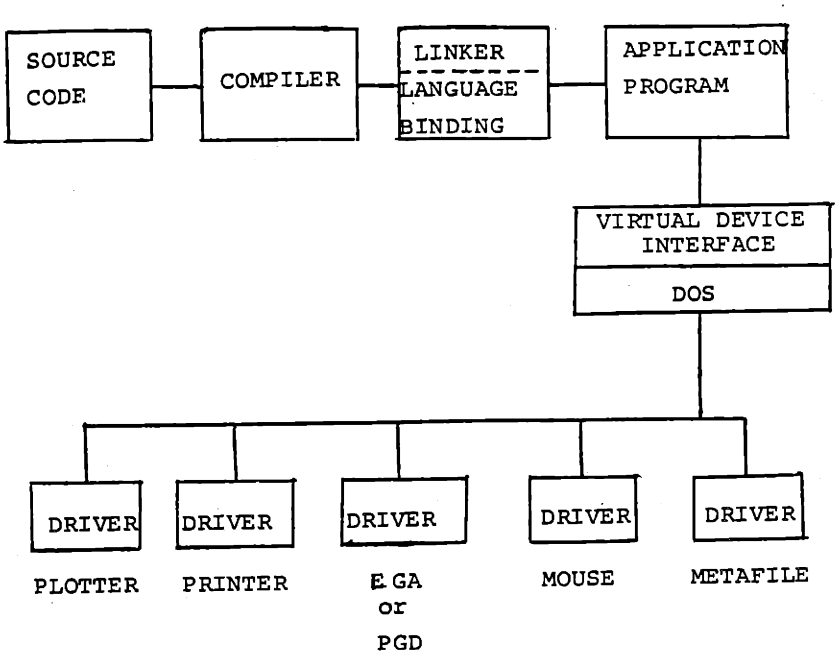


Figure A.17: The role of the VDI Controller

is a device-dependent program that serves as a information exchange medium between the graphics application program and the supported device. The device-independent graphics application program communicates with the device-dependent drivers through the Virtual Device Interface Controller (VDI), see Fig. A.17. The VDI is a layer of software at the operating system level that receives high level device independent graphics commands from the application program and passes them to device drivers. The drivers translate the high level commands into low level commands specific to each device. The VDI is a standard interface between the graphics application program, the operating system and the devices resident in the system. This feature provide to all VDI based applications program portability and device independence. VDI also provides both basic and Generalized Drawing Primitives (GDP), such as polyline, polymarker, filled areas, arcs, circles, and bar commands.

A.6.2 Other Issues

There are several considerations in choosing the graphic package to be used. For the IBM PGS there are only two graphics packages available, GKS and GDT. Both packages have a number of graphical input and graphical output functions. Graphical text input and positioning or pointing on the screen can be done using the keyboard, while pointing or positioning for graphical input can be done faster using the mouse. Both GKS and GDT reside on top of VDI. That is, graphics input and output are device independent. In addition, the extra services provided by the VDI are extended better by GKS than GDT. The language binding of each package was described earlier.

Probably the most important attributes of the graphics package are graphics input and output functions available, segmentation capabilities, storage, retrieval, and manipulation of graphics images. However, because of the diversity of engineering problems, neither may provide the graphic functions needed for the particular application. But this does not prevent the system programmer from developing his own software modules that offer the needed capabilities. In the software development of the PN/CAD package, the structure of the metafile provided by incorporating the Graphical File System within the graphics package could not generate an adequate data structure that would facilite data managment and processing.

A.7 Software/Hardware Resources

The operating system for the IBM PC processors must be DOS Version 2.1 or any other later version. There is a set of graphic standards that have been developed lately that serve as an interface and facilitate the communication of the computer system hardware and software. A list of graphics standards that may be used by a typical applications program is given below:

- Graphics Kernel System
- Graphics Development Toolkit
- Graphics File System
- Virtual Device Interface
- Virtual Device Metafile
- Initial Graphics Specification Code

The graphic standards differ by the kind of service that they provide.

The Disk Operating System (DOS) Version 2.1 or higher is an interactive, single user operating system of the IBM PC. Each IBM PC AT allows only one user to be logged on. Each user uses the IBM PC to issue commands and observe results. The IBM PC is a disk-file oriented machine; that is, most utilities and programs use disk files for input and output. Interactive design takes place by using the terminal and the PN/CAD Software. Output data is placed on files so that can be read by the CAD software package. Each IBM PC has 22Mb of total disk storage space of which approximately 20Mb can be used for file storage.

Sometimes due to hardware or software problems the system may be very slow. In anticipation of a probable hardware failure such as a disk crash, it is highly recommended to the system designer to keep a backup copy of the latest design session in system diskettes.

A.7.1 Graphics Hardware

In this section a short description of the basic devices attached to the computer is provided. Such a discussion is essential in obtaining an understanding how peripheral devices operate.

A.7.2 Graphic Displays and Controllers

A Video Display Monitor is the primary output display device. This is a raster-scan device where there is a methodical scanning of the screen area in horizontal and vertical directions. The monitor screen consists of a large number of picture elements, called pixels, in both directions. During raster-scanning, color is placed at each picture element of the image. The number of pixels in each of the principal directions may not be equal. In such case, the image formed is not square. Each pixel has its own unique location on the screen and can be assigned a set of attributes to it, such as color. An electron beam from a cathode-ray tube strikes the coated surface of the display in a raster scan way. By varying the intensity of the beam the amount of light emitted at each pixel changes. The display controller, which serves an interface between the computer and the raster-scan device, determines the intensity of the beam.

The display controller is the device that the computer uses to tell the display monitor how to color each pixel. For every pixel in the image a digital value, which determines the color of the pixel, is recorded. All recorded values are stored in the frame-buffer memory of the display controller. To generate an image, the display controller reads the values corresponding to each pixel from the frame-buffer memory. It then looks up the corresponding value in the color map table to determine the values of red, green and blue (RGB) colors needed to generate the pixel's color. These values are intensities of the red, green and blue beams of the cathode ray tube that are used to control the pixel color. However since the image generated on the display fades rapidly, the scanning process is repeated with a specified frequency.

There is a large number of displays in the market today. The differences among

them are summarized in terms of the following characteristics:

- number and size of pixels,
- resolution and,
- color range.

The number and the size of the pixels that make up the whole image are very important parameters. For example, the Professional Graphics Display (PGD) operating in the Professional Graphics Controller (PGC) mode has a size of 640 horizontal by 200 vertical pixels. The greater the number of pixels the better the image. The resolution of a display is the size of the smallest object that can be discerned. In some displays the horizontal and vertical resolutions are the same, while in some others they differ. The color range of a display determines how many different colors the pixel may take on.

A.7.3 Display Processors

The displays may be connected to the computer in number of ways. The simplest way is depicted in Fig. A.18, and represents the IBM Color Graphics Adapter (CGA). The frame-buffer memory is directly accessible by the computer. The Central Processing Unit (CPU) can read and write directly into the frame buffer. The graphics software package can do the modifications of the contents of the frame buffer. The CGA may operate in one out two modes:

1. high resolution (640×200), two color mode
2. medium resolution (320×200), four color mode

This method is quite slow, since the computer is used to execute the application program and update the frame buffer (act as a display processor). The Enhanced Graphics Adapter or Controller (EGA), is highly compatible with the CGA. It provides higher

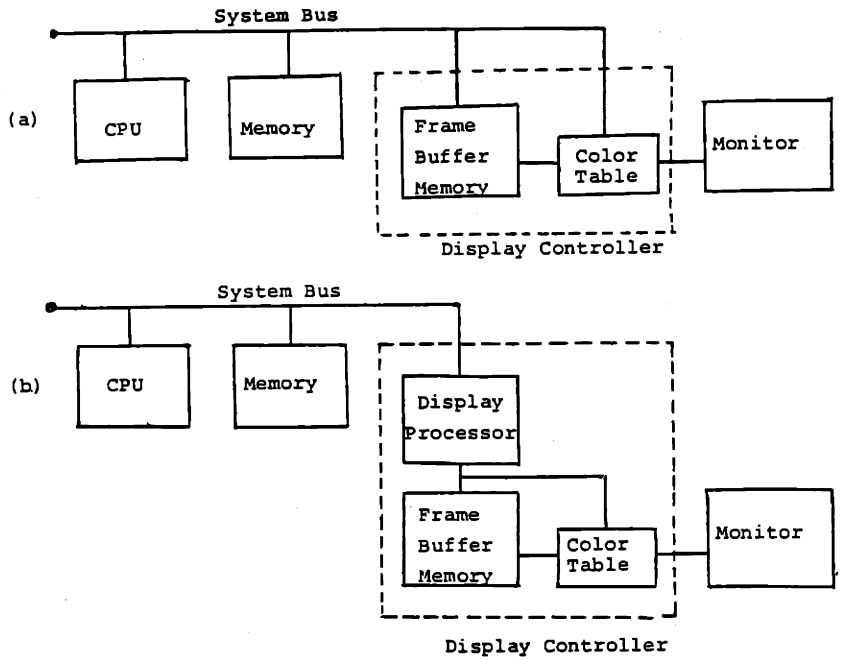


Figure A.18: Frame buffer controller-computer connection

horizontal, vertical and video frequencies than the CGA, allows the use of 64 possible colors (only 16 can be displayed at once), and supports several new function calls for the character set. The capabilities of the EGA depend on the amount of memory intalled on the board and the type of color monitor the system is connected to. The best performance is obtained when the EGA is connected to the Enhanced Color Display (ECD).

A better configuration is provided by the IBM Professional Graphics Controller (PGC) and its Professional Graphics Display (PGD). This configuration uses an 8088 microprocessor as a display processor. In this case the computer informs the display processor of all the needed updates of the display. In other words, the PGC/PGD package is a computer which has the task of providing the necessary display modifications. While these modifications are carried out by the display processor, the computer continues executing the program, thus improving the system performance. Thus the distinguishing feature of the overall system is the parallel processing. The highlights of the IBM PGC are that it

- provides expanded graphics mode
- runs in CGA emulation mode, with 640 × 200 pixel definition
- enhanced text character set
- 2-D and 3-D graphic capabilities
- 640 × 480 pixel definition when in High Function mode
- 256 colors from a palette of 4096
- Intel 8088 processor
- user re-definable color selection

The highlights of the PGD are:

- resolution of 67 pixels/in both horizontally and vertically
- addressability of 640 × 480 operating in High Function mode
- anti-reflection screen
- high horizontal, vertical and video fraquencies

Special device drivers are written and serve as logical interface between the application program and the devices being controlled by the application.

A.7.4 Output Devices

Printers: There are several type of printers available with capabilities of printing graphics. Each type of the printer can have different printing modes and a distinctive style of printing. Among the several printers are:

Dot Matrix Printers: The dot matrix printer is simple, fast, versatile and inexpensive. It the most used printer. Small dots form the characters of the dot matrix

printer, similar to the pixels of the video display. The density of the dots determines the resolution of the text or graphics image. It can be used to create elaborate graphics. A disadvantage is that the characters and images will not have entirely smooth edges.

Laser Printers: This an expensive printer, but it has speed, flexibility, high quality print and graphics, and quiteness. The graphic capabilities vary according to the machine.

Graphic Printers and Plotters: The graphic printers and plotters produce paper copies of the graphic image. They produce higher quality graphic output than the dot matrix printers. Pen plotters draw graphics output by moving a pen on paper. Multi-color plotting is possible by using several pens of different colors. There are three types of plotting mechanisms. The flatbed plotter, where the paper remains stationary while the pen moves on the X and Y axes. The drum plotter, where the plotter rotates the paper as the pen moves back and forth. The third type, is sort of combination of the first two.

A.7.5 Input Devices

Input devices allow the user to communicate with the application's program. This is the means where the designer enters information into the system, process the information, and manipulates graphic images. For graphics the best input devices are the keyboard and the mouse.

The keyboard is the most common input device. The user can type commands or reply to a message using the alphanumeric keys. The keyboard has also a set of keys able to perform functions. The purpose of the function keys is defined by the application program and may represent a macro operation to be performed by the system. The keyboard is also capable of graphic input using the directional arrow keys. These keys control the movement off the cursor (cross hair) on the video screen. By striking an arrow key the cursor moves on the screen in small increments.

The best device for steering a cursor on screen is the mouse. This is a small, hand-

size, high resolution device with one or more buttons, that is moved around on a flat surface. It can be used only for graphics input. A set of mouse functions is available to enhance the capabilities of the application's program. A mouse is good for positioning and pointing at objects on the display.

A.7.6 Interfacing

All peripheral devices such as displays, printers, plotters, mouse, require external interfaces in order to communicate with the computer. Several interfaces are now in use. The serial interface transmits data over a wire, one bit after another. Some printers use the RS232 standard protocol which is a serial interface. The parallel interface sends bits over parallel wires. Some printers use parallel interface, so the user should check for compatible interfaces.

The system bus, is an internal interface linking the central processing unit of the computer with the special purpose chips of the machine.