A Study of the Recoverability of Computing Systems

Philip Meir Merlin

TECHNICAL REPORT #58 - November 1974

UNIVERSITY OF CALIFORNIA

Irvine

A Study of the Recoverability of Computing Systems

A dissertation submitted in partial satisfaction of the
requirements for the degree Doctor of Philosophy
in Information and Computer Science

by

Philip Meir Merlin

Committee in charge:

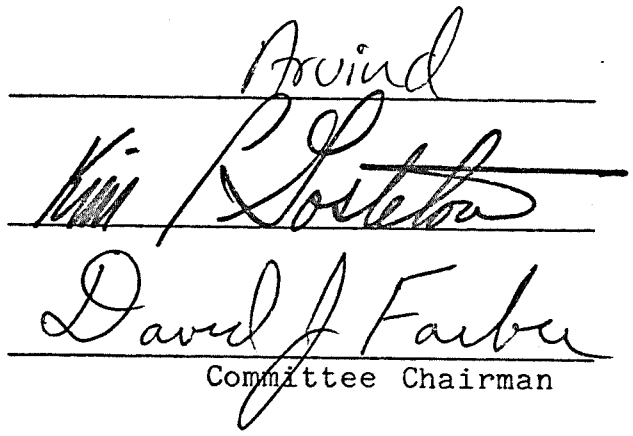
Professor David J. Farber, Chairman

Professor Kim P. Gostelow

Professor Arvind

1974

The dissertation of Philip Meir Merlin is approved, and it is acceptable in quality and form for publication in microfilm:



Committee Chairman

University of California, Irvine

1974

DEDICATION

To Bracha, and my Mother.

CONTENTS

LIST OF FIGURES

ACKNOWLEDGMENTS

I am indebted to Professor David J. Farber, committee chairman, who provided excellent support, guidance and encouragement throughout my education at UCI.

There are many others who have assisted in my studies here. In particular, I would like to express my sincere thanks to Professor Kim P. Gostelow and Professor Arvind, the committee members, for their insightful discussions and constructive criticism. I am grateful to Professor Peter Freeman, who contributed in many ways to the completion of my studies, and Professor Fred M. Tonge for his help during my first steps at UCI.

I thank the Faculty of Electrical Engineering at Technion, The Israel Institute of Technology, for providing the stimulating atmosphere during my formative years. In particular, I thank Dr. Jehuda Kella, my M.S. Thesis advisor, for his continuous encouragement and guidance. My gratitude is due to Professors Zvi Kohavi, Michael Yoeli, Israel Cederbaum and Michael Werner who introduced me to the fields of computer science and computer engineering.

I am grateful for the financial support received through the National Science Foundation under grant GJ-1045 "The Distributed Computer System", the Department of

Information and Computer Science, and the Graduate Division of the University of California at Irvine.

VITA

April 17, 1947 - Born - San Luis, Argentina

Nationality: Israeli

1969-1971: Lecturer, Preliminary Course,
Technion - Israel Institute of Technology,
Haifa, Israel

1970: Associate Project Engineer, Elsint,
Haifa, Israel

1970: Teacher, Technion Preliminary Course,
The Ministry of Absortion,
Haifa, Israel

1971 B.S., "Cum Laude"
Technion - Israel Institute of Technology,
Electrical Engineering

1971-1973: Assistant, Faculty of Electrical Engineering,
Technion - Israel Institute of Technology,
Haifa, Israel

1972-1973: Teacher, Bosmat-Technion,
Haifa, Israel

1972-1973: Projects Supervisor, Bosmat-Technion,
Haifa, Israel

1973 M.S., Technion - Israel Institute of Technology,
Electrical Engineering

1973- Research Assistant, Department of
Information and Computer Science,
University of California, Irvine, U.S.

PUBLICATIONS

"Communication Between the PDP-15 and the Mahatz-1 Computers"; M.S. Thesis; Technion - Israel Institute of Technology, Haifa, Israel; July 1973

"A Parallel Mechanism for detecting Curves in Pictures", with D.J. Farber; accepted for publication in I.E.E.E. Trans. on Computers; January 1975

PATENT

"The V.P.P.", with D.J. Farber and J. Sklansky; Submitted to the Board of Patents of the University of California

ABSTRACT OF THE DISSERTATION

A STUDY OF THE RECOVERABILITY OF COMPUTING SYSTEMS

by

Philip Meir Merlin

Doctor of Philosophy in Information and Computer Science

University of California, Irvine, 1974

Professor David J. Farber, Chairman

This work is an approach to the study of computer systems. Although only the problem of recoverability of processes is presented, the same method seems applicable to solve other problems as well, such as "fail-soft", "fault-tolerant", "best-effort", "security", etc..

This study is theoretic but its results are of practical interest. The theoretic study was carried out using a model of computation, the Petri-net, in which a representation of failures was introduced. A process is defined as "recoverable" if and only if after the occurrence of a failure, the process will return to one of its "legal states". In the Petri-net model, this definition can be

formally stated and studied.

Recoverability of a Petri-net, after the occurrence of any kind of failure, is formally defined. Recoverability for a particular kind of failure is extensively analyzed, and the necessary and sufficient conditions that a "state description" of a process must satisfy in order to represent a recoverable process, is found. A procedure to find a structural description of a process (a Petri-net) corresponding to any given state description of the process (the Token Machine) is developed. Thus, it is possible to design recoverable processes in two steps. First, a state description of the process is designed so that it satisfies the conditions of recoverability. Then the structure of the process is derived from its state description. In a similar way, it is possible to transform a non recoverable process into a recoverable one with the same state description, if such a recoverable process exists.

The recoverable processes represented by Petri-nets were found to have some practical limitations. In order to remove these limitations, a new model, the Time-Petri-Net, is defined. This model is a modification of the Petri-nets. In the Time-Petri-Net, practical examples of recoverability were demonstrated. The Time-Petri-Net has many interesting properties outside the area of recoverability. A future general study of this new model looks to be profitable.

Chapter 1

INTRODUCTION

In this work, we will deal with the modeling of systems, trying to gain insight into the design of recoverable processes. From one hand, many models of computation have been developed. On the other hand, models at the hardware level have given solutions to the problems of hardware reliability. Our work studies the recoverability problems at the system level, using a model of computation and applying an approach motivated by that used in the hardware area.

1.1 Background

During the last few years, computing systems have become more and more complex and, in many cases, are composed of several, almost independent, units working in an asynchronic parallel mode. This characterization seems to be clear for distributed computer systems as well as for the centralized systems. Today, "one" computer is composed by several CPUs, memories, I/O processors, pieces of software, and each of these units are almost independent. Several

computer networks, connecting tens of computers and serving a wide range of users, have been developed and are operating. This increase in the complexity of the computing systems will, probably, continue in the future [FALK74].

With the increased number of elements that compose computer systems, the probability of a failure of at least one of these elements is relatively high. In this case, it is important to protect the remaining elements and the operation of the system. Thus, it is necessary to organize the systems so that if parts of its elements are malfuctioning, the rest of the elements will continue to work correctly. The total performance of the system can be reduced by a failure (and several users can be affected), but one or a few elements may not cause the entire system to collapse. This philosophy is called the "best-effort" [FARB72, FARB73, ROWE73, METC73].

To deal with the complexity of these systems, the concept of "process" was introduced [DENN66, SALT66, HORN66, DENN71, GOST74, and many others]. To study the properties of processes, several models have been proposed [PETR62, ESTR63, KARP69, GOST71, CERF72]. But a great amount of obscurity covers parts of these concepts. Many properties of processes are not well understood to date. The concept of failure recoverable processes has been almost totally ignored. Without a better understanding of this concept, it is unlikely to expect a good implementation of the

"best-effort" philosophy in complex systems that will be designed in the near future.

This manuscript proposes an approach to the study of failure recoverable processes and for the analysis of the possible sequences of events that happens when a failure occurs. By "recoverability" we mean that after the occurrence of a failure the control of the process is not lost, and after several steps it will return to "normal execution". Note the difference between this concept and the concept of "correctness of results". In the present work, we do not deal with the problem of correctness. We are concerned with control recoverability, that is, structural recoverability. Our approach is motivated by the philosophy that we can accept the situation in which a user get some erroneous results, but we do not accept the possibility that a single error (or failure) may cause the entire break-down of the total system.

The approach presented in this work is based on a model of process behavior, the Petri-nets, in which we have introduced the concept of a failure.

This dissertation solves only a small part of the problem that we belive it is necessary to solve in order to achieve the implementation of the "best-effort" philosophy. But this work not only solves a practical problem (with the aid of a theoretical handling), but it shows a possible way of solving many other important open questions. I believe

that the most important achievement of this research is that I have demonstrated that a formal definition of our "world" and a theoretic analysis of this definition can lead to the solution of many of the practical problems .

1.2 Related Work

Many efforts have contributed, or in some way are related to this research.

First, the development of several models of computation. Among them the Petri nets [PETR62, HOLT68, HOLT69], the UCLA Graph Model of Computation [ESTR63, MART66, BAER68, BOVE68, RUSS69, VOLA70, GOST71, FERN72, CERF71], the Karp and Miller - Slutz model [SLUT68, KARP69], the coordination net [PATI70] and the Rodrigez model [RODR69]. All these models have many common features. For example, Gostelow proved the equivalence of Petri nets and the UCLA model [GOST71], and coordination nets are based on Petri nets [PATI70].

These models are used, in the main, to study the problems of parallel computation like deadlocks, resource allocation, coordination , etc.. Although much work has been done, most of the results remain limited to research circles and they have had little influence in practical designs.

In contrast to this models, "hardware" research has succeded in developing powerful tools for logic-failure

detection and location [ARMS66, ROTH66, ROTH67, SELL68, YAU71, KOHA71, KOHA72], for hazard detection [YOEL64, EICH65] and for many other problems. These tools have had a strong impact at the design level.

Today several recoverable systems have been built (for example the DCS at University of California Irvine [FARB72, FARB73, ROWE73]) and it is expected that in the near future many other systems will be designed. But the designer of these recoverable systems have had no tools similar to the mathematic tools that the designer of, for example, a logic network tester has.

This work tries to supply the designer of recoverable systems with a design tool. The method is based on the Petri net model of processes, but it is, in some sence, influenced by the philosophy of the mathematical tools developed for hardware.

1.3 Outline of this dissertation

In chapter 2, the model of processes is defined and a representation of failures is introduced. The concept of recoverability is defined and some implications of these definitions, for the case of single and multiple failure of the same type, are studied.

Chapter 3 and 4 show a way of designing recoverable processes, for a give kind of failures ("loss of token" failures). Chapter 3 develops a method of designing the

structure of a process when a state description is given. Chapter 4 points out the conditions that a state description of a process must satisfy in order to be implementable as a recoverable process. Using the method of chapter 3, the structure of this process can be derived. The theorems that prove the statements made in chapter 3 and 4 are presented in appendices A and B respectively.

Chapter 5 shows some practical limitations of the recoverable processes modeled by Petri-nets. In order to remove these limitations, chapter 6 defines a new model, the Time-Petri-net (TPN). Chapter 6 also studies the TPN, especially from the point of view of recoverability.

Chapter 7 examines a practical example, the recoverability of a communication protocol between two processes.

In chapter 8, a summary and suggestions for further exploration are given.

Chapter 2

THE MODEL

In this chapter the process model is presented. The proposed model is based on a variation of Petri nets and it includes a way of representing a possible failure. This chapter also presents my definition of "recoverability".

2.1 Petri Nets (PN)

Petri nets were developed by Carl Adam Petri and further elaborated by Anatol Holt [HOLT68].

Petri nets model "conditions" represented by nodes and "events" represented by transition bars. The holding of a condition is represented by placing a token on that node. Directed arcs connect nodes to bars and bars to nodes. A transition bar (event) can fire (occur) if all the nodes (conditions) input to that transition bar (event) have tokens (hold). When a transition bar fires, it removes one token from each input node and places one token on each output arc.

Figure 2.1 shows, as an example of Petri net use, the model of a simple protocol, P3, connecting between processes

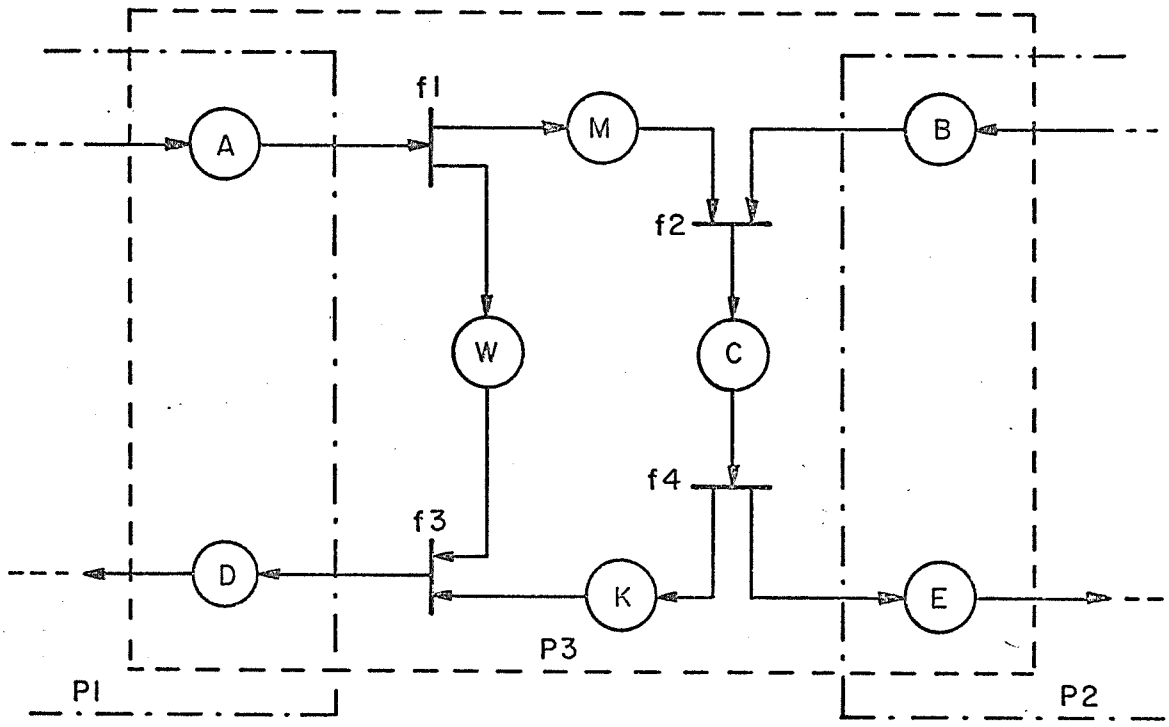


Figure 2.1: Petri-net representation of processes

P1 and P2. Note that protocols are a special kind of process [GOST74]. In this protocol, when P1 is ready to send a message (A holds a token) f1 fires. Then a message is sent (M holds a token) and the sender enters the state of waiting for the acknowledge (P3 records this fact since W holds a token). When P2 is ready to receive the message (B holds a token) f2 will be fired, at which point C gains a token (the message is received). In this position f4 can fire and causes a token to pass to E and to K. A token on E is the notification to P2 that the message was received, and the token in K represents the fact that the acknowledge signal is sent to P1. Now f3 can fire, removing a token from W and from K, and putting a token on D. The token on D represents the notification to P1 that the transfer was finished. P1 can remove the token from D, and again trigger the process P3 by putting a token on A. On the other side, P2 can remove the token from E and to start his part of the protocol activity by putting a token in B.

The state of the Petri net is defined by the collection of names of the nodes holding tokens. The number of instances of a node name in a state is equal to the number of tokens the node holds in this state. All the possible states in which a Petri net can stay and the possible transitions between them define a state machine called Token Machine (TM). The TM for the net of figure 2.1 is shown in figure 2.2, assuming the initial condition AB.

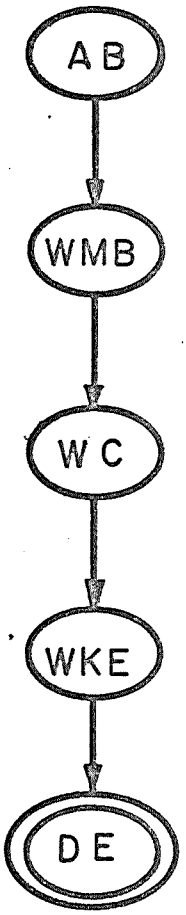


Figure 2.2: Token Machine for the PN of figure 2.1

The description above is correct only if every component of the protocol P3 functions properly. But, what happens if the message M is lost? What is the behavior of P1 and P2 under this failure? With the tools developed to date we can not answer these questions in general. In the next section we will develop new tools for a better understanding of these questions.

2.2 The Representation of Possible Failure

Suppose that a condition in a Petri net may fail. In this case, a token held by this condition may disappear. It is possible to represent this by adding a new branch to the TM. This branch will represent the possible flow of the execution when the "problematic" token disappears. For illustration, suppose that the message M in figure 2.1 can be lost. It means that when M holds a token, this token can disappear. This situation can be represented by adding an arc from state WMB (figure 2.2) to a new state WB. Since in state WB no bars can fire, WB is a final state. This new TM is shown in figure 2.3; thick lines represent the TM in the case that no failures occur, and thin lines describe the paths added since a failure. On the arc connecting these two parts, we write the name of the failing condition (M in the example). The same representation will be used in all TM figures in this dissertation. This new machine, including the TM and the added paths for possible errors,

will be called Error Token Machine (ETM).

In the ETM, we call those states that also exist in the TM, "legal states". The other states are called "illegal states". Note that a process may be in an illegal state only if a failure has occured.

From figure 2.3, we conclude that the process represented in figure 2.1 is not recoverable from a failure in M, because under such failure the execution sequence arrives to a state (WB) where there is no way to return to normal execution (a legal state).

Note that in general, the ETM for a possible failure includes all the possible paths in which the execution can flow if this failure occurs. This means that it is necessary to add to the TM new branches exiting from all the states that include the possible faulting condition.

At this point we can state the conditions for recovery:

> "A process P is recoverable from failure F if and only if in the ETM of P for failure F, all the directed paths through illegal states arrive to legal states in a finite number of steps."

Thus, after a failure, the execution sequence must return to normal execution after a finite number of steps.

In this work, we will limit our study to processes that have finite TM. From the properties of directed graphs, for the case of finite TM we can derive an equivalent set of conditions for recoverability:

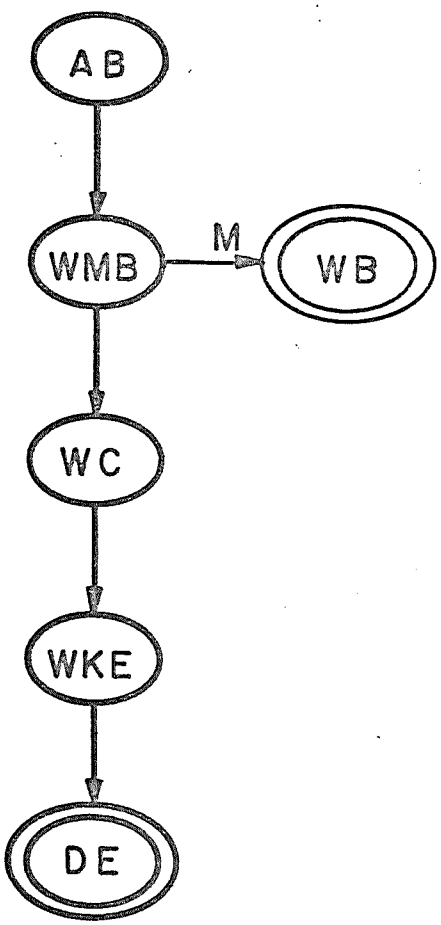


Figure 2.3: ETM for figure 2.2 under a failure in M

"A process P is recoverable from failure F if and only if in the ETM of P for failure F:

1. the number of illegal states is finite,
2. there are no final illegal states,
3. there are no directed loops including only illegal states."

These three conditions we will call the Conditions of Recoverability (COR). In the following chapters we will use this second definition of recoverability.

In the previous discussion we have dealt only with the case that a token may disappear. But, in the same way, because of a fault, a node in the Petri net may generate a token. This situation may also be represented by adding new branches to the correspondent nodes in the TM. The approach is similar to the previous case.

2.3 Multiple Failures

In this section the previous approach is extended to the case that several failures can occur. This extention is introduced by the two following examples.

2.3.1 Example

Figure 2.4 describes a variation of the protocol shown in figure 2.1. Assume (for simplicity) that arc T can be activated only after "a long time". This assuption is not mathematically represented in our model and we leave this

point to further discussion in chapters 6 and 7.

Figure 2.5 shows the ETM for this protocol (assuming initial condition AB) for the case that M may fail. The ETM of figure 2.5 shows that in spite of the failure in M the execution sequence (in case of failure) returns to a legal state (DE). This shows that the protocol described in figure 2.4 is recoverable from one failure in M.

Suppose now, that a second failure may occur in M. In this case, a new branch will be added, but now to the node LMB (figure 2.5). This procedure can be applied again for any given number of possible failures. Note that this procedure is suitable not only for the case of multiple failures in one condition, it can be applied in general when several conditions may fail.

Figure 2.6 shows the ETM for the process of figure 2.4 in the case that three failures in M may occur. In this ETM we can see that the process is recoverable from one or two failures but not for the third. Paths 1 and 2 return to a legal state, but not path 3.

2.3.2 Example

Figure 2.7 describes another variation of the Petri net shown in figure 2.1. In this case, there is a protocol process (P9) connecting the process P7 and P8. Note the appearance of the ill-defined function T that is described in figure 2.4. Figure 2.8 shows the ETM (assuming initial

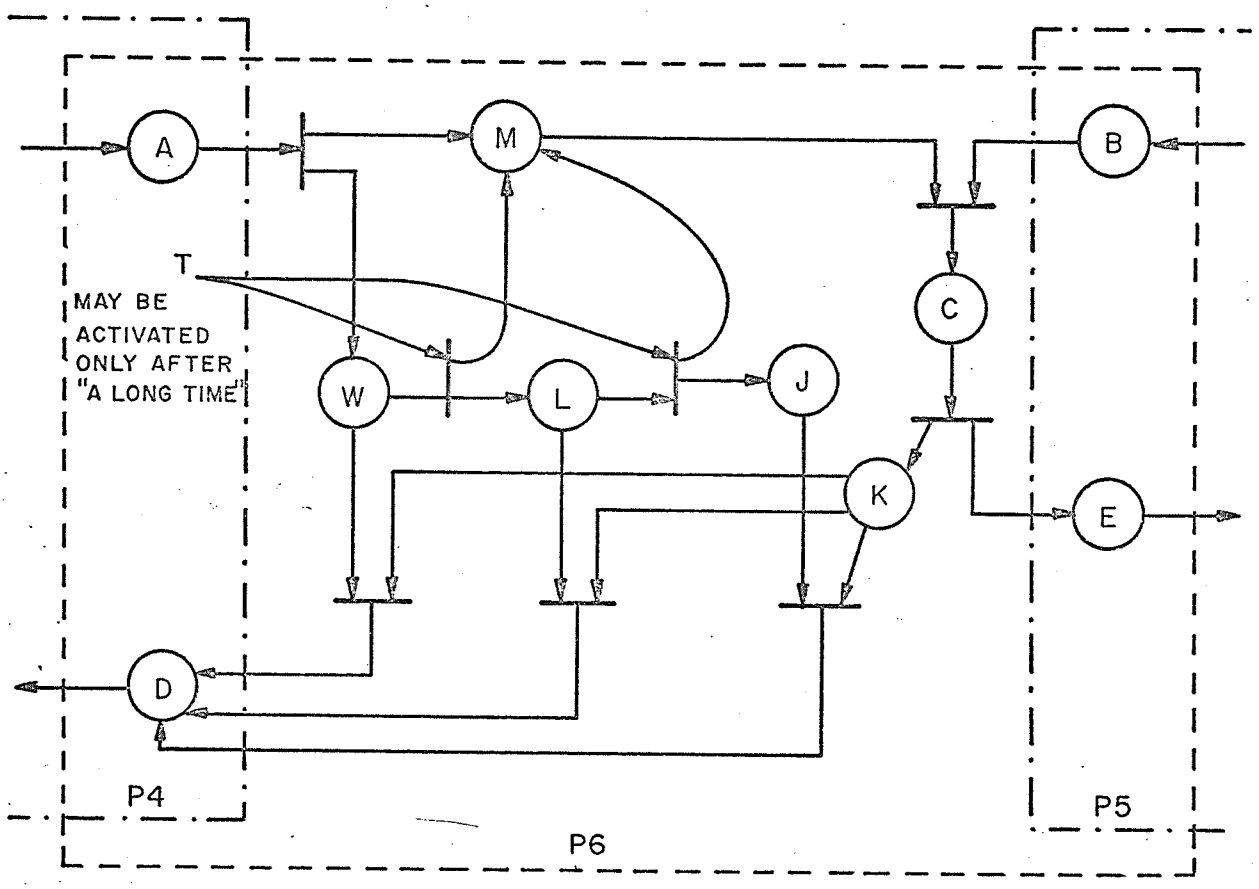


Figure 2.4: Petri net of example 2.3.1

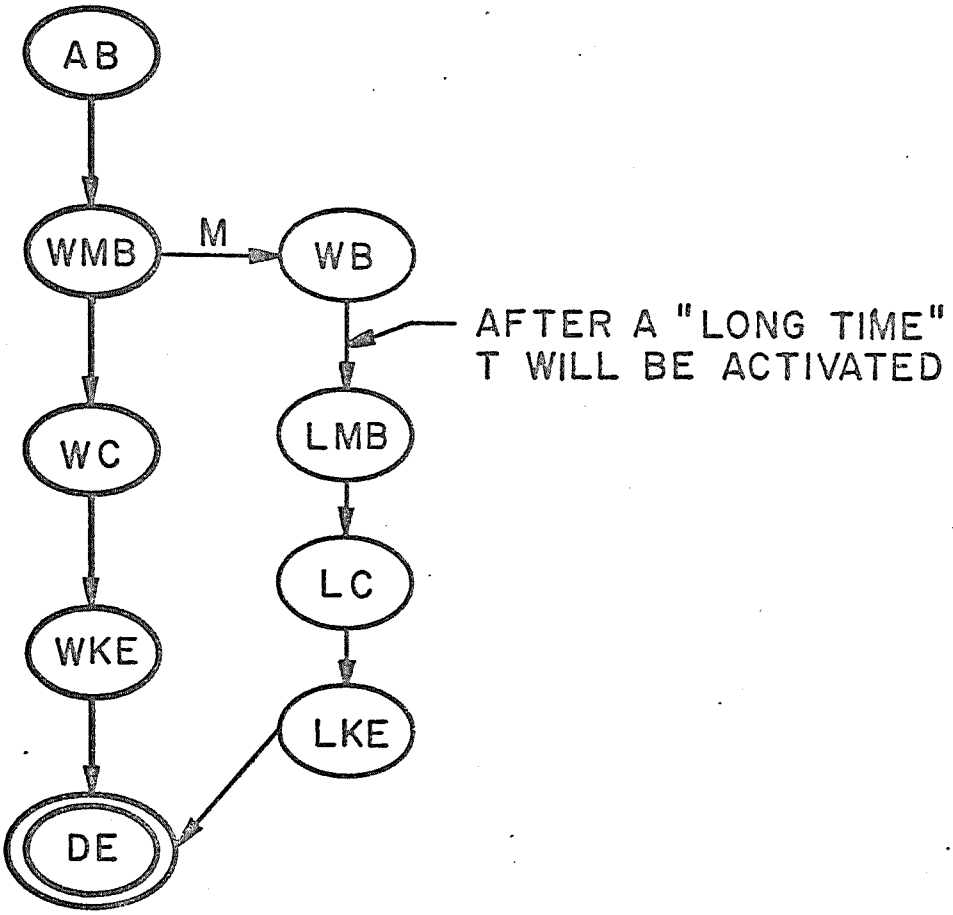


Figure 2.5: ETM of example 2.3.1

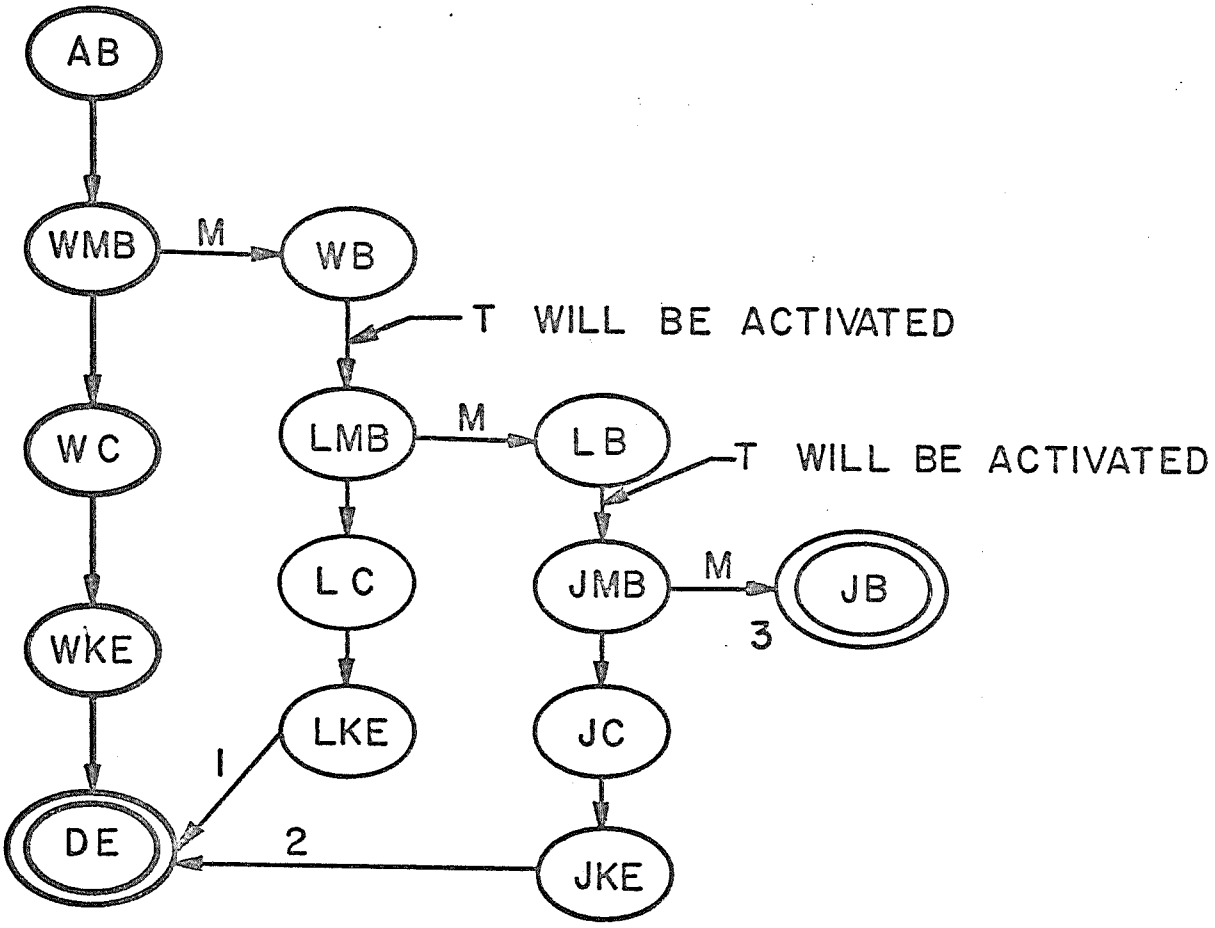


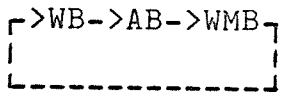
Figure 2.6: ETM of example 2.3.1

condition AB) for the protocol P9 in which a failure in M may occur. In this case, the new branch for the failure returns to a legal state (WMB->WB->AB). This demonstrates that this failure is recoverable.

Suppose now, that two faults may occur in M. In this case, the sequence in the ETM of figure 2.8 will be:

AB->WMB->WB->AB->WMB->WB->AB

This means that for one or two failures, the machine executes the same sequence around the loop:



The result is similar for more than two errors in M. The loop will be executed as many times as a failure in M may occur. But always, the execution sequence will return to a legal state. This shows that this protocol is recoverable from any number of failures in M.

2.3.3 Test for Recovery Under Multiple Failures of the Same Kind

In the case of multiple failures of the same kind, ETM(i) denotes the ETM showing all the possible paths for the occurrence of up to i failures of the same kind. For example, using this notation, figure 2.5 shows the ETM(1) of the PN of figure 2.4 and figure 2.6 shows the ETM(3) for the same PN. In a similar way, we denote IS(i) the set of states of ETM(i) that do not exist in ETM(i-1). In other

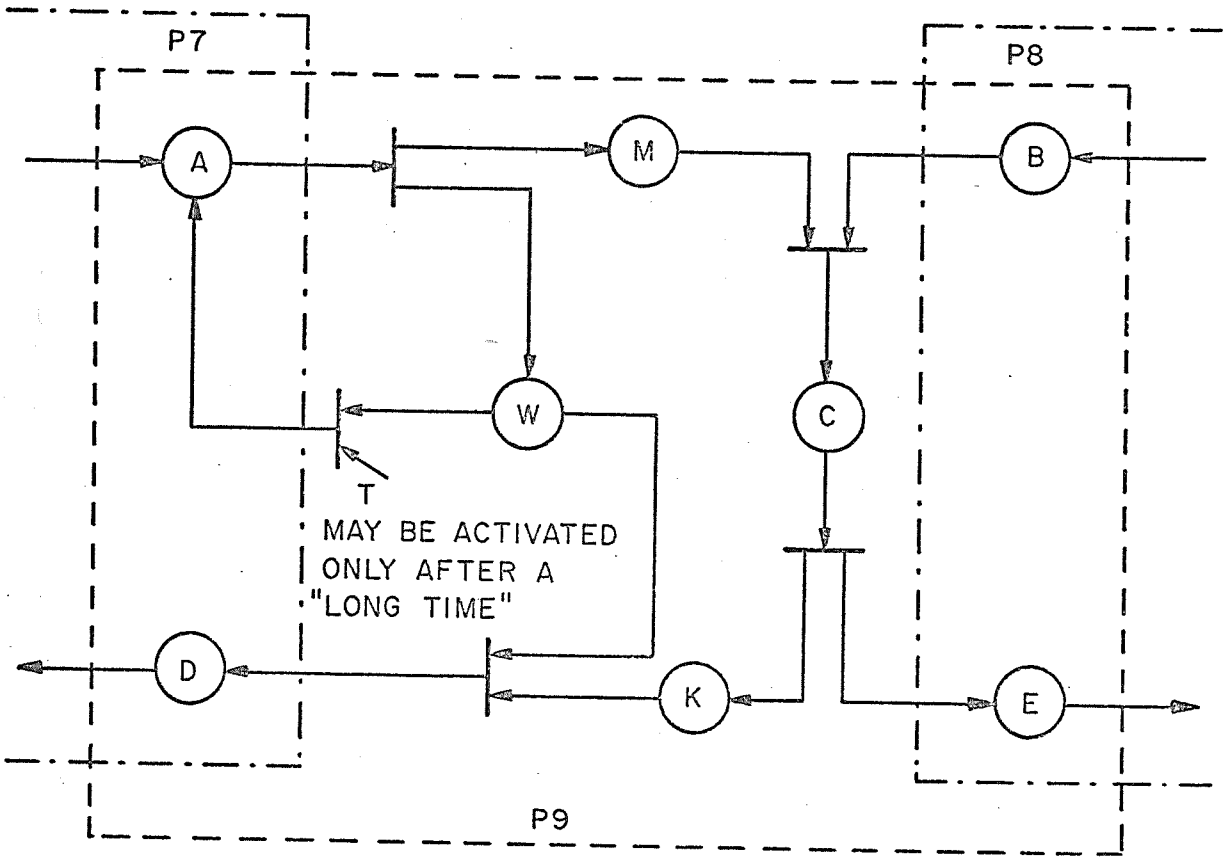


Figure 2.7: Petri net of example 2.3.2

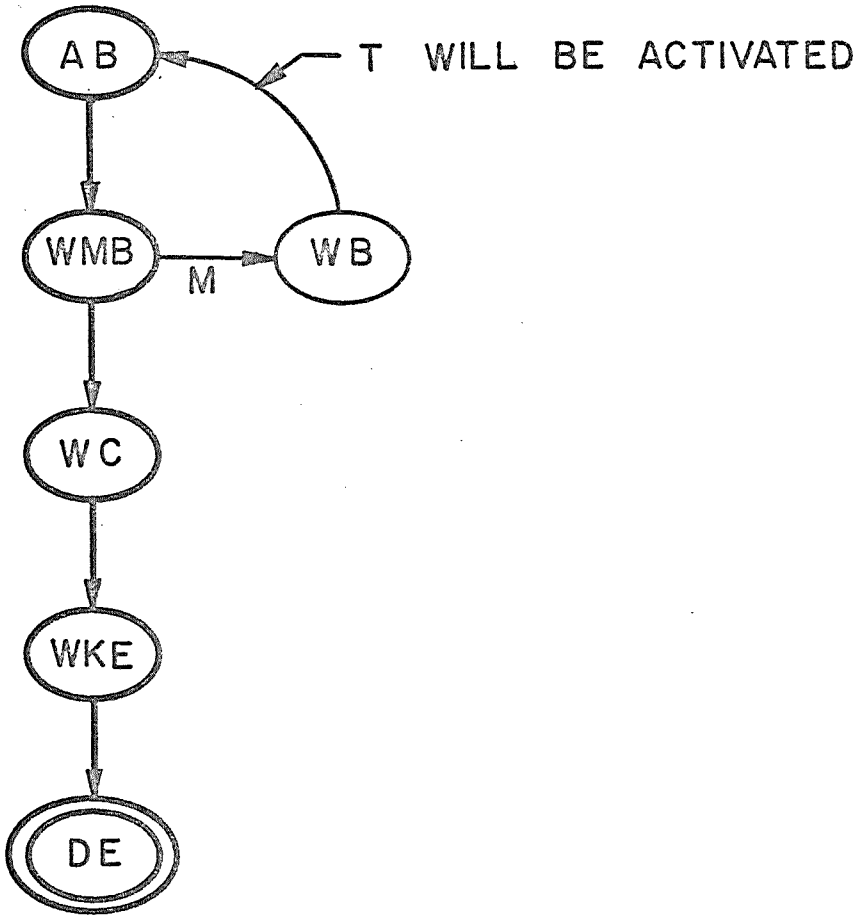


Figure 2.8: ETM of example 2.3.2

words, IS(i) denotes the new illegal states added to the ETM as a result of the occurrence of the i-th failure. Using this notation, the TM can be denoted by ETM(0) and the legal states by IS(0). For example, in figure 2.6:

IS(0) = [AB;WMB;WC;WKE;DE]

IS(1) = [WB;LMB;LC;LKE]

IS(2) = [LB;JMB;JC;JKE]

IS(3) = [JB]

and in figure 2.8:

IS(0) = [AB;WMB;WC;WKE;DE]

IS(1) = [WB]

IS(2) = Ø

By definition, ETM(i) includes all the states and transitions of ETM(i-1), and ETM(i) depends only upon ETM(i-1). This means that ETM(k) can be built iteratively. Starting with the TM, and adding the new paths for the occurrence of a failure , the ETM(1) is built. Then, starting with the ETM(1), and adding the new paths for the occurrence of a failure, the ETM(2) is built, and so on until ETM(k) is built.

Applying a similar definition of recoverability as in 2.2, a PN is recoverable after the occurrence of k failures in F if and only if in the ETM(k) for this failure the three conditions COR (see section 2.2) are satisfied.

In the same way, a PN is recoverable after the occurrence of <u>any</u> number of failures in F if and only if

there exists a k0 such that for any k>k0,

$$ETM(k) = ETM(k0)$$

and ETM(k0) satisfies the conditions COR.

In general, the ETM(i) has the following properties:

1. The ETM(i) is formed by the ETM(i-1) and the new paths through IS(i). All these new paths start at the nodes of the set IS(i-1), because the occurrence of the i-th failure implies that (i-1) failures occurred before.
2. If:

   $$ETM(i) = ETM(i-1)$$

   then:

   $$ETM(k) = ETM(i)$$

   for any k greater than i.

   This property derives from the previous one and can be easily proved by induction.
3. If:

   $$ETM(i) = ETM(i-1)$$

   and if the PN is recoverable from i-1 failures in F then it is recoverable from any number of failures in F. This property derives from the previous one and from the definition of recoverability.
4. If a PN is not recoverable from i failures in F it is also not recoverable from k (k greater than i) failures in F.

Using these four properties of ETM(i) we can design an algorithm that tests for recoverability after the occurrence

of k failures in F.

ALGORITHM (to test for recoverability after k failures in F):

(Suppose P is a process and F a possible failure.
The data structure ETMI represents ETM(i)
The vector ISI represents IS(i)
The data structure TM represents the TM
COR(ETMI) is equal to true iff COR applied on ETM(i) are satisfied.)

```
(1) ETMI := TM ; ISI := nodes of TM
(2) for I := 0 step 1 until K-1
    do begin
(3)         ETMI := ETMI + new paths for errors F in the
                                nodes ISI;
(4)         ISI := nodes of the new paths;
(5)         if (ISI = ∅) then stop (P is recoverable from any
                                    number of failures in F);
(6)         if not COR(ETMI)
                then stop (P is recoverable from up to I
                                    failures in F)
(7)     end
(8) stop (P is recoverable from at least K failures in F)
```

This algorithm has several interesting properties:

1. The trivial case is when after the first execution

of (3) there is a directed loop including legal and illegal nodes, and F is not included in the symbolic description of any illegal state. In this case, in the second execution of (3), there will be no change in ETMI , so that ISI is null and the algorithm terminates. Figures 2.7 and 2.8 (section 2.3.2) show an example of this case.

2. At this point, we do not know if there exists a procedure to compute the value of K for any given PN, so that the algorithm can decide if P is recoverable from any number of occurrences of F.
3. "Recoverability from <u>any</u> number of failures F" properly includes the case that after the occurrence of certain number of failures F in which the control continues executing a sequence of legal states which never include the condition F. In this case, the failure F <u>cannot</u> <u>occur</u> again. So that, "recoverable from any number of failures F" actually means that faults in F will not inhibit the process from returning to "normal execution".

2.4 Discussion

In this chapter, a method has been developed for checking if a process is recoverable from a given possible failure. The fact that all the examples that introduce the method are related to protocols does not restrict its

generality. All the methods presented and all the characteristics described in this work apply to processes in general.

In section 2.3 the method has been extended for the case that several failures of the same kind can occur. In section 2.3.3 an algorithm is presented. This algorithm tests the behavior of a process under any given number of occurrences of a given failure, from the point of view of recoverability. The same approach can be generalized to study the case of any number of occurrences of several distinct failures.

The presented method allows one not only to know if a process is recoverable under failures, but also to know what are the possible sequences of events under those failures.

In the following chapters, an approach to the problem of designing recoverable processes is developed. We will assume that the structure of a process is described by a PN. The PN will be designed in two steps. In the first step the desired TM will be built, in the second, a PN that implements the given TM is constructed. As shown in the following chapters, if the TM is built according to certain rules, and if the PN is constructed correctly, then the PN will be guaranteed recoverable. The next chapter analyses the problem of building a PN that executes a given TM.

Chapter 3

THE PETRI NET OF A GIVEN TOKEN MACHINE

Chapter 2 shows the importance of the TM, and its generalization, the ETM, in the algorithms that test if a process is recoverable under a given failure. As a consequence, for a better understanding of recoverability, the properties of the TM have to be studied.

The TM is defined from the Petri net. In this chapter, the properties of the TM are studied, and the problem of constructing a Petri net corresponding to a given TM is analysed. In this work, we restrict our study to finite TM's only.

3.1 General Explanation

Figure 3.1 shows a TM. The problem is how to build a Petri net so that its TM is the one shown in figure 3.1.

We can assume that the set of "conditions" (or "places") is given by the Boolean union of all the characters of the states' names in the TM. In the example:

$$[S \cup (A \cup B) \cup (C \cup A) \cup (C \cup D) \cup (C \cup B) \cup (B \cup D) \cup (B \cup B) \cup C] =$$

$$= [S, A, B, C, D]$$

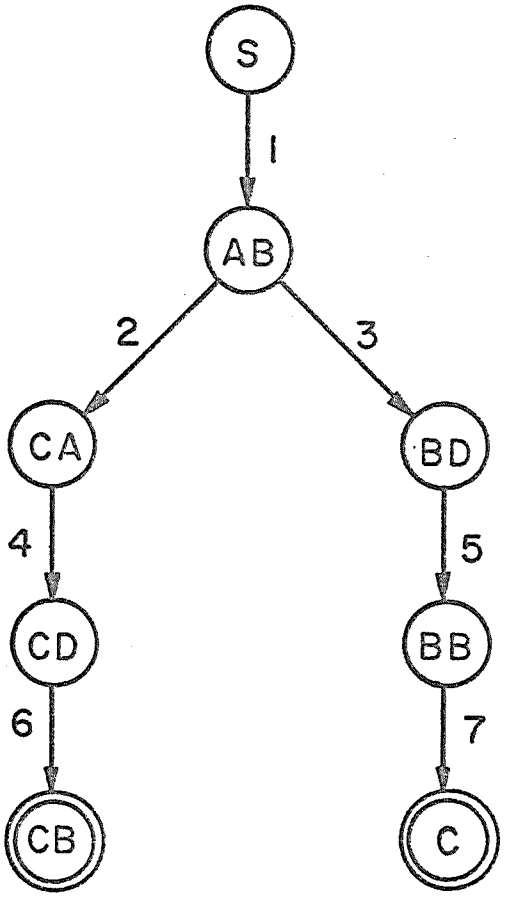


Figure 3.1: A Token Machine (TM)

Each state in the TM corresponds to a possible state in the Petri net. Each arc in the TM corresponds to a firing of some transition in the Petri net. Thus, for each arc in the TM there is a bar in the PN that can carry out this transition. For example, arc 6 in figure 3.1 represents a transition from state CD to state CB. That is:

CD->CB

Figure 3.2 shows the two possible ways of implementing this transition in the PN.

Arc 5 represents the transition:

BD->BB

Figure 3.3 shows the possible bars corresponding to this transition.

One of the bars of figure 3.2 and one of figure 3.3 must be in the PN. But, the same bar is in figure 3.2(b) and in figure 3.3(b). In this case, a single bar is sufficient to execute the arcs 6 and 5 of the TM. The other possibilities are also legal, but, two bars are required.

The example above shows that a TM can be implemented by different Petri nets, with different numbers of bars. The PN with the minimal number of bars that implements a given TM is called the Minimal Petri Net (MPN) of the TM.

Arc 2 (figure 3.1) represents the transition:

AB->AC

Figure 3.4 shows the possible bars corresponding to this transition. Note that the bar in figure 3.4(b) can fire

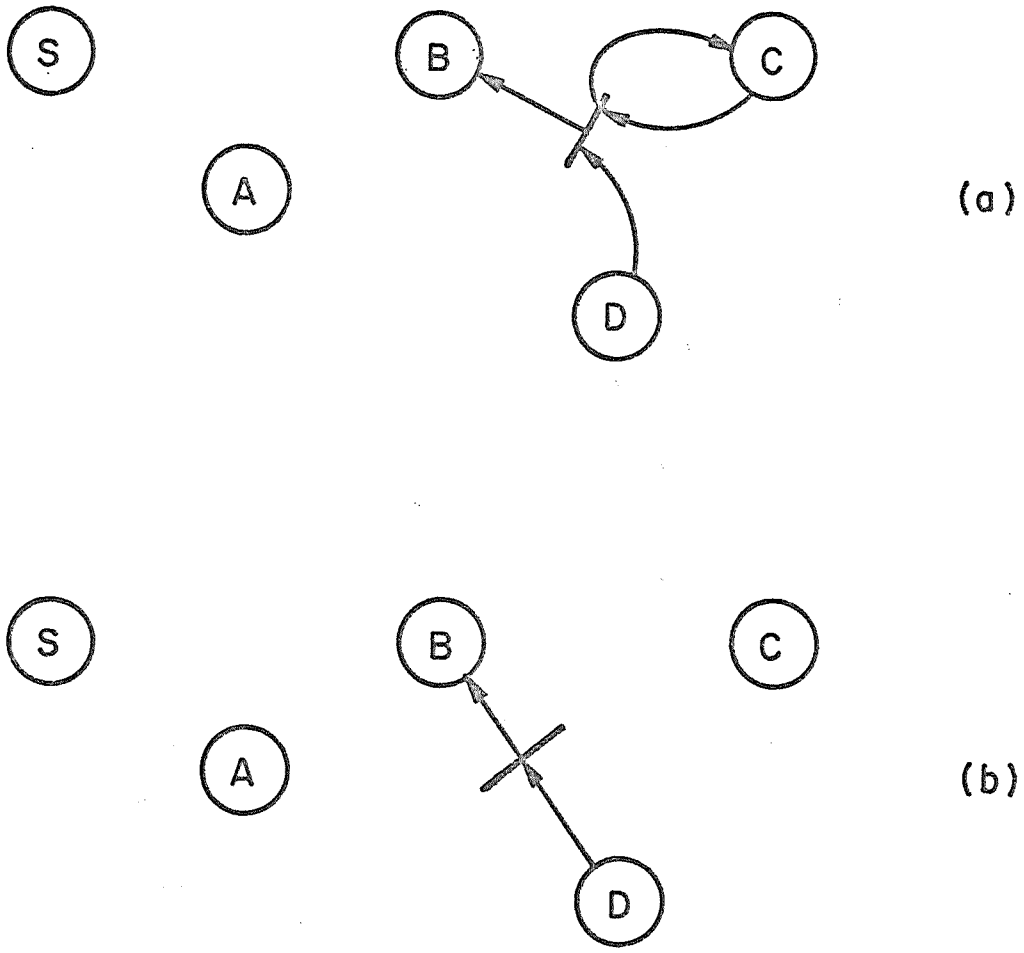


Figure 3.2: The Possible Implementations of CD->CB

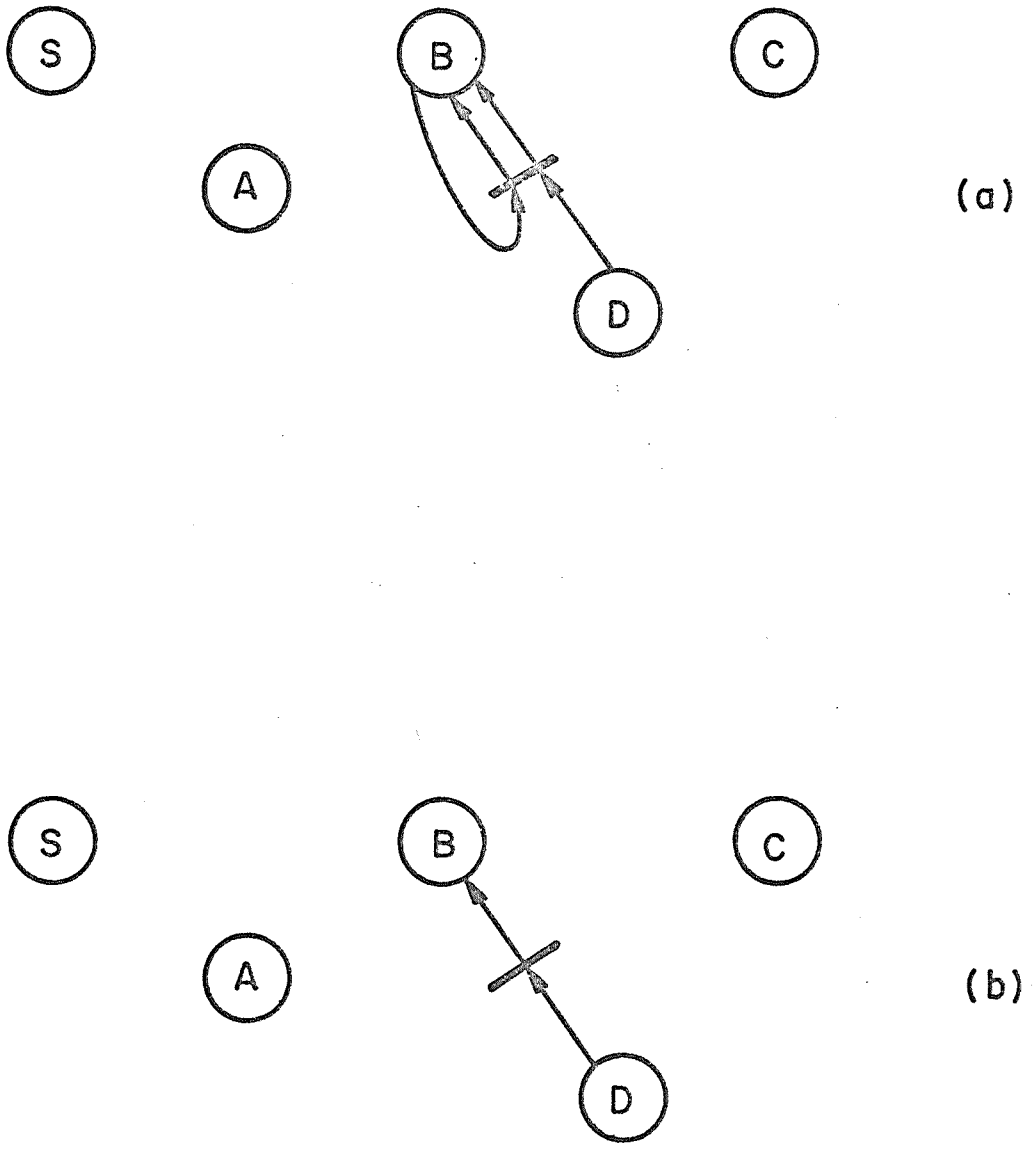


Figure 3.3: The Possible Implementations of BD->BB

each time that B has a token. The states BC, BD, BB are legal states in the TM. In this case, the bar of figure 3.4(b) can fire also the following transitions:

BC->CC

BD->CD

BB->CB

But, since in the TM, there are no arcs corresponding to these transitions, they are not allowed in the Petri net. This means that only the bar of figure 3.4(a) can be used to implement the arc 2 of figure 3.1.

The example above shows that in the implementation of a PN corresponding to a TM there exists two problems:

1. the PN may not be unique,
2. it is necessary to prevent unallowed transitions that can appear as side effects of the implementation of allowed arcs.

Arc 2 (figure 3.1) is an example of this last problem. The only possible way of preventing unallowed transitions is to add conditions to the input set of the bars that execute the allowed transitions. In figure 3.4(a), the firing from B to C is limited by the condition A. But, if this approach is adopted for all the transitions, the number of bars will increase unnecessarily, as shown in figures 3.2(a) and 3.3(a). In this case, two bars are needed instead of just one.

In the following sections the construction of the Petri

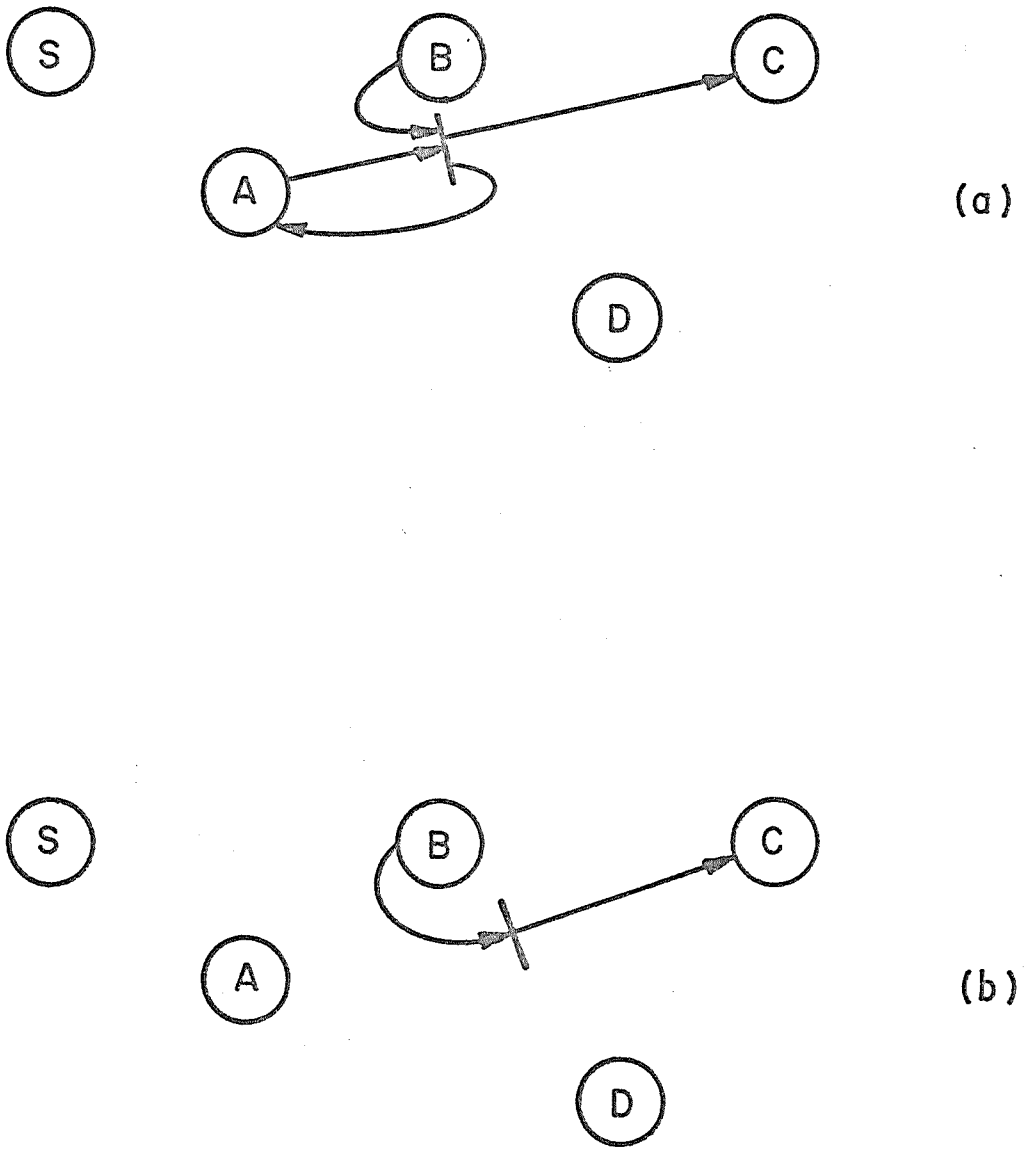


Figure 3.4: The Possible Implementation of AB->AC

net of any given Token Machine is formally studied.

3.2 Formal Definitions

The first nine definitions in this section are related to the mathematical entity called bag ([CERF71], [KNUT69], [RULI71], and [GOST71]). Our notation differs a bit from the previous publications on this subject. These changes were motivated by the willingness to present a notation suitable to the special needs of this work.

3.2.1 A bag is a collection of elements which may have more than one instance of the same member. A bag is denoted by angle brackets. As an example:

$$B = \langle c,b,c \rangle$$

denotes the bag B of two instances of c and one instance of b. Since a bag is unordered B can also be denoted by:

$$B = \langle c,c,b \rangle = \langle b,c,c \rangle$$

In this work, we also use the following alternative notation:

$$B = 2.c + b$$

which means: "the bag B is composed of two times the element c and one time the element b".

Note that a set is also a bag, and all the operations on bags are defined for sets.

3.2.2 The function #(b,A) gives the number of instances of the element b in the bag A.

As an example:

$$\#(a,\langle a,b,a,c,a,b\rangle) = 3$$

3.2.3 b is a member of bag A ($b \in A$) iff $\#(b,A) \geq 1$.

3.2.4 The domain of a bag A (Dom A) is the set of elements used to form the bag. In formal notation:

$$a \in \text{Dom } A \iff a \in A$$

and:

$$\forall a\ (a \in \text{Dom } A) \Longrightarrow \#(a,\text{Dom } A) = 1$$

Example:

$$\text{Dom } \langle a,b,a\rangle = [a,b]$$

3.2.5 Bag A is a subbag of bag B ($A \leq B$) if there is in B an instance for each instance of the elements of A. Formally:

$$A \leq B \text{ iff } (\forall a \in (\text{Dom } A))\ (\#(a,A) \leq \#(a,B))$$

Example:

$$\langle a,b,a,b,b,c\rangle \leq \langle a,a,a,b,c,c,b,b,d\rangle$$

The following statements are proven in [CERF71]:

1. $A \leq B$ & $B \leq A \iff A = B$
2. $A \leq B$ & $B \leq C \iff A \leq C$

We denote $A \not\leq B$ the fact that A is not a subbag of B.

3.2.6 The sum of two bags ($A + B$) is a bag composed by all the instances in A and B. Formally, ($C = A+B$) if $(\forall c \in ((\text{Dom } A) \cup (\text{Dom } B)))\ (\#(c,C) = \#(c,A) + \#(c,B))$

Example:

$$\langle a,b,a\rangle + \langle a,b,c,b\rangle = \langle a,a,a,b,c,b,b\rangle$$

The subtraction of two bags (A - B) is the bag C, if:

$$C + B = A$$

We do not deal with "negative elements", so that this operation is defined only when $B \leq A$. Formally, when $B \leq A$, $(C=A-B)$ if

$(\forall c \in ((\text{Dom } A) \cup (\text{Dom } B)))\ (\#(c,C) = \#(c,A) - \#(c,B))$

3.2.7 A transition is an ordered pair of bags. Let A and B be bags, and:

$$t = A \rightarrow B$$

then t is a transition. The right and left part of t are given by the functions RHS and LHS. That is:

1. LHS(t) = A
2. RHS(t) = B

3.2.8 The Remainder of a transition (RE(A -> B)) is the largest bag C such that:

1. $C \leq A$;and
2. $C \leq B$

i.e. RE(t) is all the instances of the elements that are common to RHS(t) and LHS(t).

3.2.9 The Minimum of a transition (MIN(t1)) is a transition t2, such that if:

$$t1 = A \rightarrow B$$

(A and B are bags) then:

1. LHS(t2) = A - RE(t1)
2. RHS(t2) = B - RE(t1)

Note that LHS(t2) and RHS(t2) have no elements in common. In other words, the minimum of a transition is another transition whose left and right hand sides have no elements in common.

By definition, any transition t is:

t = RE(t) + LHS(MIN(t)) -> RE(t) + RHS(MIN(t))

Example:

arc 5 of figure 3.1 can be denoted by:

t5 = <B,D> -> <B,B>

then:

1. LHS(t5) = <B,D>
2. RHS(t5) = <B,B>
3. RE(t5) = <B>
4. MIN(t5) = <D> -> <B>

3.2.10 A Petri net (PN) is a directed graph defined by the 4-tuple [B,C,A,S0], where:

B = [b1,....,bm] is a finite (possible empty) set of transition bars.

C = [c1,....,cn] is a finite (non-empty) set of conditions.

The elements of the sets B and C are the nodes of the PN.

A = [a1,....,aq] is a finite set of directed arcs. Each arc connects either a condition to a bar

or a bar to a condition. In general, more than one arc may connect, in the same direction, the same two nodes.

S0 = non-empty bag. The elements of S0 are members of C. This means that:

$$\text{Dom } S0 \leq C$$

S0 denotes the initial distribution of tokens in the conditions C.

3.2.11 The Input Conditions of a bar (IC(bi)) is a bag A that satisfies the following conditions:

1. the elements of A belong to the set of conditions in the PN (the set C of definition 3.2.10). In other words:

$$\text{Dom } A \leq C$$

2. bag A has exactly k instances of cj (j = 1,2,..,n) iff there are k arcs from cj to bi.

3.2.12 The Output Conditions of a bar ((OC(bi)) is a bag A that satisfies the following conditions:

1. as in the definition of input conditions, A satisfies:

$$\text{Dom } A \leq C$$

2. bag A has exactly k instances of cj (j = 1,2,..,n) iff there are k arcs from bi to cj.

3.2.13 A Legal State (Si) of the PN is either the bag S0 (see definition 3.2.10) or a bag that is a possible result of the firing algorithm (3.2.14).

3.2.14 The Firing Algorithm

A bar bi can fire if IC(bi)$\leq$Sj (Sj is a legal state). If bi fires, a token is removed from each condition in IC(bi), and a token is placed in each condition of OC(bi). The new distribution of tokens defines a new legal state Sk.

Following the definitions, the new state is equal to:

$$Sk = Sj - IC(bi) + OC(bi)$$

3.2.15 A Token Machine (TM) of a PN is defined as a tuple [S,T], where:

S = [S1,S2,...] is a set of bags. Si belongs to S if and only if Si is a legal state of the PN. The set S may be finite or infinite.

T = [t1,t2,...] is a set of transitions of the form tk = Si -> Sj; tk belongs to the set T if and only if in state Si a bar

can fire bringing the PN to state Sj.

3.2.16 A non-superfluous PN is a PN=[B,C,A,S0] such that for the corresponding TM=[S,T]:

1. for each ci in C there exists at least one element in S in which ci is included and
2. for each bar bi in B there exists at least one state in S in which bi can fire.

3.3 Formal Analysis

Suppose that a finite TM=[S,T] is given. The problem is to find a PN=[B,C,A,S0] so that its TM is the given one. We limit our work to non-superfluous PN's (definition 3.2.16).

Since the PN is non-superfluous, each of its conditions is represented at least in one of the states of the TM. In this case, the set of conditions of the PN is given by:

$$C = Dom\ (S1 + S2 + \ldots\ldots + Sn)$$

Now, it is necessary to find the set of bars and the set of arcs, such that:

1. All the transitions of T are implemented
2. Only the transitions of T are implemented.

Theorems A.1 to A.5 (Appendix A) give a formal proof of relations existing between a TM and its corresponding PN. We present here only the results necessary to continue our discussion. The reader interested in a more detailed and

formal study of these relations is refered to appendix A. The important results of the theorems in the appendix are:

3.3.1 If there exists a TM transition tr = Si -> Sj so that MIN(tr) = a -> b and there exist a bar bf such that:

1. IC(bf) = a + x (x is a bag)
2. OC(bf) = b + x and:
3. x $\leq$ RE(tr)

then tr in TM is effected by bf in PN. In other words, these are conditions such that if bar bf satisfies them, bf will effect transition tr in the TM. (This statement is proven in A.1)

3.3.2 If the TM transition tr = Si -> Sj with MIN(tr) = a -> b is effected by the bar bf then:

1. IC(bf) = a + x (x is a bag)
2. OC(bf) = b + x and:
3. x $\leq$ RE(tr)

(this theorem is proven in A.4)

These two previous propositions state the necessary and sufficient conditions that a bar must satisfy in order to effect a TM transition. In other words, TM transition tr = Si -> Sj with MIN(tr) = a -> b is effected by the PN bar bf <u>if and only if</u>

1. IC(bf) = a + x
2. OC(bf) = b + x

3. $x \leq RE(tr)$

3.3.3 If there is a TM transition exiting from a legal state, then there is no other transition with the same "minimal transition" exiting from the same state. (Proven in A.2)

3.3.4 TM transitions with different "minimal transition" must be implemented by different PN bars.(Proven in A.5)

The fact that transitions with different "minimal transition" must be implemented by different bars, has an important practical implication. This fact means that the set of transitions can be divided into different groups, each group having the same minimal transition, and each group implemented independently of the others.

Suppose the members of a group are:

$$tk_p = Si_p \text{ -> } Sj_p \qquad ; (p=1,2,....,r)$$

and the minimal transition for all the members of the group is:

$$MIN(tk_p) = a \text{ -> } b \qquad ; (p=1,2,....,r)$$

The statement in 3.3.1 (and also in A.1) shows that each member of this group can be implemented by a bar bp that satisfies:

1. $IC(bp) = a + xp$ (3.3.5)

and 2. $OC(bp) = b + xp$ (3.3.6)

where xp satisfies:

3. $xp \leq RE(tk_p)$ (3.3.7)

The statement in 3.3.2 (also in A.4) shows that the only way to implement them is by a bar that satisfies these three requirements. As shown above, there is certain freedom in the election of the xp. This means that the members of any one group can be implemented in different ways, leading to different PN's. The same xp can be chosen for different transitions (in the same group). In this case, the same bar effects several transitions. The number of different bars that implement the group is equal to the number of different xp's chosen for this group. The trivial case is:

$$xp = \emptyset \qquad ; (p=1,2,....,r)$$

then, the entire group is implemented by only one bar.

At this point, we know how to implement all the transitions corresponding to a certain group. The remaining problem is to implement only the transitions existing in the given TM. After an xp is chosen for equations (3.3.5), (3.3.6) and (3.3.7), then by definition of the firing algorithm (3.2.14), bar bp can fire in all states Si such that:

$$IC(bp) \leq Si$$

By the statement 3.3.4 (and proven in A.5), all the transitions executed by bp have the same minimum:

$$a \rightarrow b$$

and by the statement in 3.3.3 (and proven in A.2) for each state Si there can be only one transition with the minimal

transition:

$$a \rightarrow b$$

Thus, when the machine is in state Si_p, then bp will effect only legal transitions that exist in the given TM, and when the machine is in a state:

$$Sk \quad ; \quad (k \neq i_p) \quad ; \quad (p=1,2,....,r)$$

if bp fires, it will effect a transition that does not exist in the given TM. So, in order to prevent the implementation of transitions that do not exist in the given TM, xp has to satisfy the condition:

$$IC(bp) = a + xp \not\leq Sk \quad ; \quad \text{for all } Sk \neq Si_p \qquad (3.3.8)$$

In order to implement <u>all</u> the transitions of the group, and <u>only</u> all the transitions of this group, it is necessary to find a set [xq], subset of the bag <xp>, so that:

$$\forall p \; \exists q \quad xq \leq RE(tk_p) \qquad (3.3.9)$$

and

$$\forall q \; \forall k \quad (k \neq i_p) \Longrightarrow a + xq \not\leq Sk \qquad (3.3.10)$$

Note that it is enough to check (3.3.10) for the states Sk that include a. If Sk does not include a, (3.3.10) is satisfied independently of xq.

In general, the following cases may exist:

1. only one set [xq] satisfies (3.3.9) and (3.3.10). In this case, there exists only one PN implementation of the TM transition of the group tk_p.
2. Several sets [xq] satisfy (3.3.9) and (3.3.10). In

this case, there exist several PN implementations of the TM transitions of the group tk_p.

3. There is no set [xq] that satisfy both (3.3.9) and (3.3.10). In this case, there is no PN corresponding to the given TM. This case holds <u>if and only if</u> there is a state Si_p which is a subbag of a state Sk , ($k \neq i_p$). Theorems A.6 and A.7 (in appendix A) prove this statement.

3.4 Examples

The previous section (also A.6 and A.7) gives necessary and sufficient conditions such that a group of transitions can be implemented. Thus the TM can be implemented by a PN, if and only if all of its groups can be implemented.

Depending on the properties wanted in the PN (minimal number of bars, some special configuration, etc.), there exist several algorithms for choosing the elements of the sets [xq] (when such set exist). One possibility is to choose $xq = xp = RE(tk_p)$. But this solution gives a large number of bars. In the following examples, some possibilities are shown.

First, the implementation of the TM of figure 3.1 is shown. The groups of transitions are shown in table 3.1

<u>For group 1</u>:

$$IC(b1) = \langle S \rangle + x$$

since no states outside the group include S, x=Ø is chosen.

TABLE 3.1: Groups of transitions of the TM of figure 3.1

group	minimal transition	transition	RE(t)
1	S -> AB	S -> AB	ø
2	B -> C	AB -> AC	A
3	A -> D	AB -> BD AC -> CD	B C
4	D -> B	DB -> BB DC -> BC	B C
5	BB -> C	BB -> C	ø

TABLE 3.2: Groups of transitions of the TM of figure 3.6

group	minimal transition	transition	RE(t)
1	A -> B	A -> B	ø
2	A -> C	A -> C	ø
3	B -> CD	B -> CD	ø
4	C -> BE	C -> BE	ø
5	C -> E	CD -> ED	D
6	B -> D	BE -> DE	E
7	DE -> F	DE -> F	ø

Then:

$$IC(b1) = \langle S \rangle$$

$$OC(b1) = \langle A,B \rangle$$

For group 2:

$$IC(b2) = \langle B \rangle + x$$

The states outside the group that include B are: <B,D>, <B,B>, <B,C>. If $x = \langle A \rangle$, conditions (3.3.9) and (3.3.10) are satisfied. Thus:

$$IC(b2) = \langle A,B \rangle$$

$$OC(b2) = \langle A,C \rangle$$

For group 3:

$$IC(b3) = \langle A \rangle + x$$

Since no states outside the group include A, then $x=\emptyset$ is chosen. Then:

$$IC(b3) = \langle A \rangle$$

$$OC(b3) = \langle D \rangle$$

Also for groups 4 and 5, $x=\emptyset$ is chosen in the same way. The corresponding PN is shown in figure 3.5.

In the following example we will try implementing a PN that has the TM shown in figure 3.6. The groups of transitions are given in table 3.2.

For group 3:

The only transition of this group is

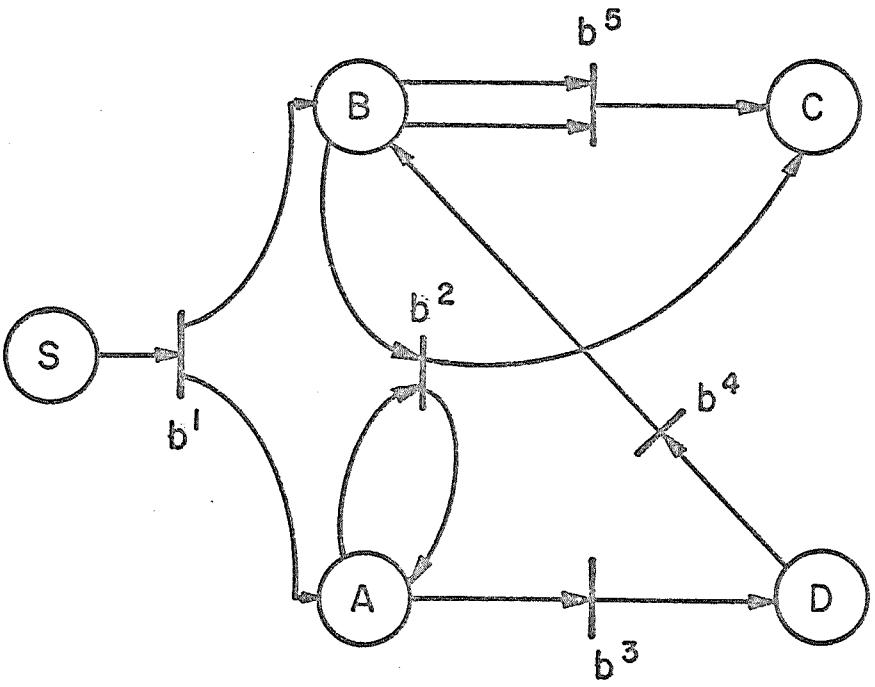


Figure 3.5: A PN that implements the TM of figure 3.1

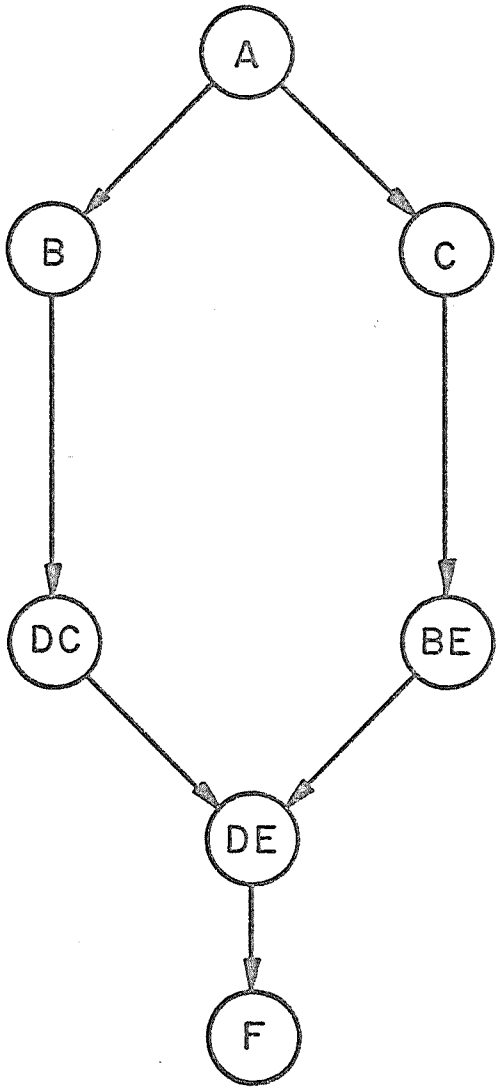


Figure 3.6: A Token Machine (TM)

$$B \rightarrow CD$$

But, BE is also a legal state, and there is no transition of group 3 exiting it. Since there exists:

$$B \leq BE$$

then, as discussed in the previous section (and proved in A.7) group 3 can not be implemented by a PN. Thus, there is not a PN corresponding to the TM of figure 3.6.

If we try to implement the transition:

$$B \rightarrow CD$$

then:

$$IC(b3) = \langle B \rangle$$

but in this case, a transition:

$$BE \rightarrow CDE$$

is also implemented, and this transition does not exist in the given TM.

Chapter 4

THE TM AND THE LOSS OF A TOKEN

In this chapter, we will study the structure of the TM's which are recoverable from a failure of type "loss of token". The results of this chapter yield a better understanding of the concept of "recoverable", and the design of recoverable processes. Note that if the TM of a recoverable process is designed, then the structure of the process can be found (in terms of a PN), using the tools developed in chapter 3.

This chapter deals only with single failures of type "loss of token", as defined in 2.2. The same ideas are applicable to other kinds of failures, but we do not study this generalization in this manuscript.

In section 2.2, it was pointed out that a PN is recoverable from a single failure if and only if the corresponding ETM satisfies the three conditions COR. I.e:

1. the number of illegal states is finite,
2. there are no terminal illegal states, and
3. there are no directed loops including only illegal states.

The following sections of this chapter examine how these conditions are reflected in the TM, and what is the general structure of the TM such that the conditions are satisfied.

4.1 Definitions and Properties

The definitions of this section are introduced in order to simplify the discussions in the following sections. These definitions divide the states and transitions of the TM, and the bars of the PN, into different subsets. Each subset is defined by certain properties of its elements.

In this chapter (and also in the subsequent ones) we will suppose that F is the name of the failing condition. This means that a token in F may disappear.

4.1.1 The set of all the states in the TM is divided into two subsets:

1. $S^1 = [Si \mid F \not\leq Si]$
2. $S^2 = [Si \mid F \leq Si]$

S^1 is the subset of all the states that do not include the failing condition F. On the other hand, S^2 is the subset of all the states that include F.

Properties:

By definition, the following properties hold:

1. S^1 and S^2 have no common elements.
2. Any legal state belongs to S^1 or S^2. I.e:

$$S = S^1 + S^2$$

3. A failure can occur only when the TM is in one of the states of set S^2. In the states of S^1, F does not exist (does not hold tokens), so that it can not lose one.

Notation:

The following notation will be used:

1. S_i^1 denotes a state Si such that Si $\leqslant S^1$.

2. S_i^2 denotes a state Si such that Si $\leqslant S^2$.

3. Ai is defined as:

$$Ai = S_i^2 - F$$

Thus, the states of the set S^2 can be denoted as:

$$S_i^2 = Ai + F$$

4.1.2 The transition exiting from the states of S^2 are divided into three subsets:

1. $t^1 = [tk \mid tk = S_i^2 \rightarrow S_j^1$ for some j]

this kind of transition takes place from a state that includes F to a state that does not include F. By definition, when a bar that effects a transition of kind t^1 fires, it removes all the tokens from condition F.

2. t^2 = [tk ¦ (tk=S_i^2->S_j^2)&(if bar bf effects tk then #(F,S_i^2)=#(F,IC(bf)))]

this kind of transition takes place between two states that include F. By definition, when a bar that effects a transition of kind t^2 fires, it removes <u>all</u> the tokens from condition F. But, since S_j^2 includes F, a bar that implement a transition of kind t^2 must also place some tokens back on condition F.

3. t^3 = [tk ¦ (tk=S_i^2->S_j^2)&(there exists a bar bf that effects tk and #(F,S_i^2)>#(F,IC(bf)))]

this kind of transition takes place between two states that include F. By definition, for each transition of kind t^3 there exists at least one bar that executes it without removing all the tokens from condition F.

Note that if more than one bar effects the transition tk = S_i^2->S_j^2 then it is <u>enough</u> that <u>one</u> of these bars satisfies #(F,S_i^2)>#(F,IC(bf)) to make transition tk a transition of type t^3. Transition tk is of type t^2 <u>only</u> if <u>all</u> the bars that effect tk satisfy #(F,S_i^2)=#(F,IC(bf)).

<u>Notation</u>:

The following notation will be used:

1. t_k^1 denotes a transition tk such that $tk \lessdot t^1$

2. t_k^2 denotes a transition tk such that $tk \lessdot t^2$

3. t_k^3 denotes a transition tk such that $tk \lessdot t^3$

Properties:

1. if:

$$t_k^1 = S_i^2 \rightarrow S_j^1$$

and t_k^1 is effected by bar bf then:

$$\#(F,S_i^2) = \#(F,IC(bf))$$

This statement is proved in B.1 (appendix B).

2. By definition of the sets t^1, t^2 and t^3, each TM transition exiting from state S_i^2 belongs to one and only one of these sets.

4.1.3 When a PN (or its corresponding TM) is in a legal state S_i^2 the set B of <u>all the bars</u> is divided into four subsets:

1. b^1 = [bj | in state S_i^2 bj effects a transition t_k^1]

2. b^2 = [bj | in state S_i^2 bj effects a transition t_k^2]

3. b^3 = [bj | in state S_i^2 bj effects a transition t_k^3]

4. b^4 = [bj | in state S_i^2, bj can not fire]

Notation:

1. b_j^1 denotes a bar bj such that $bj \in b^1$
2. b_j^2 denotes a bar bj such that $bj \in b^2$
3. b_j^3 denotes a bar bj such that $bj \in b^3$
4. b_j^4 denotes a bar bj such that $bj \in b^4$

Properties:

1. When a PN (or its corresponding TM) is in state S_i^2 each bar belongs to one and only one of the sets b^1, b^2, b^3 or b^4.
2. In each state S_i^2 there can be a different distribution of the bars among the sets b^1, b^2, b^3 and b^4.

4.2 The Possible States After a "loss of token" Failure

In this section, the possible illegal states are examined. The set of all the possible illegal states after the occurrence of a failure (of type "loss of token") is found.

Suppose that the TM is in state $S_i^2 = Ai + F$. If while in this state a failure occurs, then TM will go to state Ai. In the following steps, we assume that the PN is in state Ai, and the bars that can fire in this situation will be studied.

Theorems B.2 to B.7 (appendix B) show a formal proof of

the statements presented next:

4.2.1 Bars of type b^1 in state Ai+F can not fire in state Ai. (Proved in theorem B.2)

4.2.2 Bars of type b^2 in state Ai+F can not fire in state Ai. (Proved in theorem B.3)

4.2.3 Bars of type b^3 in state Ai+F can fire in state Ai. (Proved in theorem B.4)

4.2.4 Bars of type b^4 in state Ai+F can not fire in state Ai. (Proved in theorem B.5)

In other words, only those bars that effect a transition of type t^3 from state Ai + F (bars of kind b^3) can fire in state Ai. Next, we will consider some properties of transitions effected by bars b^3. These properties are also formally proven in appendix B.

4.2.5 If bar b^3_f of type b^3 in state Ai + F implements the transition from class t^3:

$$t^3_1 = A1 + F \rightarrow A2 + F$$

then, b^3_f also implements the transition:

$$t2 = A1 \rightarrow A2$$

(this statement is proved in B.6)

4.2.6 If A1 + F is a legal state in a TM, and if there exists a transition:

$$ti = A1 \rightarrow A2$$

then there also exists a transition of type t^3:

$$t^3_k = A1 + F \rightarrow A2 + F$$

(this statement is proved in theorem B.7)

These two previous statements give one of the most important and interesting properties. In 4.2.5 it is claimed that for each transition of group t^3 from state $A1 + F$, there exists a corresponding transition from A1. On the other hand, 4.5.6 claims that for each transition from A1 there exists a corresponding t^3 transition from $A1 + F$. Thus:

> there is a one-to-one correspondence between all the transitions from a state A1 and all the transitions of type t^3 from a state $Si = A1 + F$.

The same property holds for all the states Ai. The one-to-one correspondence is beween a transition:

$$A1 \rightarrow A2$$

and a transition:

$$A1 + F \rightarrow A2 + F$$

and, furthermore, the latter is of type t^3.

After the transition from A1, the PN is in state A2. But also, A2 belongs to the set:

$$[Ai] = S^2 - F$$

since $A2 + F$ is a legal state. This means that the previous statements are applicable when the PN is in state

A2. In other words, there exists a one-to-one correspondence between the transitions from A2 and the transitions of type t^3 from A2 + F.

Thus, if the PN arrives to a state that belongs to the set:

$$[Ai] = S^2 - F$$

(this happens if a "loss of token" in F occurs) it will never leave the states of that set (recall that TM is finite). In this case, the PN will only execute transitions that correspond to the transitions of type t^3.

Note that a failure can occur in any of the states of S^2. Thus, because of a failure the PN can be in any of the states of $S^2 - F$.

4.3 The Structure of Recoverable TM's (for loss of token)

The conditions for recoverability are:

1. the number of illegal states is finite,
2. there are no terminal illegal states and
3. there are no directed loops including only illegal states.

Since this work deals only with finite TM's, the number of legal states is finite. $S^2 = [Ai + F]$ is a subset from the legal states, thus the number of states Ai + F is also finite. This implies that the number of states Ai is finite. The previous section showed that after the occurrence of a failure the PN will move only among the

states Ai. In this case, the "illegal states" are a subset of [Ai]. This implies that the number of illegal states is also finite. Condition (1) of recoverability is always satisfied for the case of "loss of a token".

In order to satisfy condition (2), all the terminal states in ETM have to be legal. In other words, if Ai + F is a legal state and there are no transitions of type t^3 exiting from Ai + F, then Ai has to be a legal state. Note that statement 4.2.6 implies (by modus tollens) that this Ai will be terminal.

Condition (3) for recoverability points out that loops comprised only of illegal states are not allowed. If, in the TM, there exist directed loops of states Ai + F, connected by transitions of type t^3, then, after a failure, there exist corresponding loops of the states Ai. In order to satisfy condition (3), at least one of the Ai states of each loop has to be a legal state. But, if a state is legal, all the successors of this state are legal (definitions 3.2.13 and 3.2.14). Thus, all the loops of states Ai, corresponding to loops of states Ai + F connected by transitions of type t^3, have to be legal.

At this point, we can summarize the previous discussion as follows:

4.3.1 A PN with finite TM is recoverable from a single failure of kind "loss of token" in F if and only if:

1. all the states Ai, corresponding to legal states Ai + F without exiting transitions of type t^3, are legal (and terminal);
2. Ai_j + F (j=1,2,..,r) are legal states connected in a directed loop by transitions of type t^3 implies that one of the states Ai_j is legal.

By definition, the difference between t_k^2 and t_k^3 is given by the implementation, that is, by the number of instances of F in IC(b). This means that there may exist certain TM's that can be implemented by either a recoverable PN or a non-recoverable PN, depending if certain transitions are implemented as t^2 or t^3. Subsections 4.3.2, 4.3.3 and 4.3.4 show a way to distinguish, directly from the TM, three kinds of TM transitions between the S^2 states:

1. implementable <u>only</u> as a t^3 transition
2. implementable <u>only</u> as a t^2 transition
3. implementable as <u>either</u> a t^2 <u>or</u> a t^3 transition.

The following properties are derived from the theorems and definitions in the previous sections:

4.3.2 All the possible implementations of a TM transition:

tj = A1 + F -> A2 + F

are <u>only</u> of type t^3 iff in the TM there exists another legal transition tk = S3 -> S4 , such that:

1. MIN(tk) = MIN(tj)

2. $S3 \leq A1$

(This statement is proven by theorems B.8 and B.9)

4.3.3 All the possible implementations of a TM transition:

$$tj = A1 + F \rightarrow A2 + F$$

are <u>only</u> of type t^2 iff in the TM there exists a legal state S3, such that:

1. $S3 \neq A1 + F$

2. $A1 \leq S3$

3. if transition tk exits S3 then $MIN(tk) \neq MIN(tj)$

(This statement is proven by theorems B.10 and B.11)

4.3.4 A TM transition:

$$tj = A1 + F \rightarrow A2 + F$$

can be implemented, by choise, as of <u>either</u> type t^3 <u>or</u> t^2 iff the conditions of 4.3.2 and 4.3.3 are not satisfied.

(This statement is deduced from 4.3.2 and 4.3.3)

Using these three properties, each TM transition between states of the set S^2 can be "marked" as implementable only as a t^2 transition, only as a t^3 transition, or as either one of them. In the case that in a TM each transition between states S^2 can be implemented only by one type of transition, the question of recoverability of its correspondent PN is very easy to answer. We have only

to check if the TM transitions, marked as transitions of type t^3, satisfy the conditions 4.3.1. If the conditions are satisfied then any PN corresponding to the TM will be recoverable. But, if the conditions are not satisfied all the possible PN-implementations will be not recoverable from "loss of token" in F. In this case, in order to achieve recoverability, the structure of the TM has to be changed. Since this change affects the other properties of the TM (or the processes represented by the TM) we do not discuss the problem of which changes are "convenient" to make. The convenience of the changes is dependent on the importance of the other properties, rather than recoverability, that we may like to keep. In general, the possible changes are: to add or delete states, or to add or delete transitions. In the next section, an example of this case is shown.

In case that, in the TM, there exist transitions that can be implemented as t^2 or t^3, the PN can be either recoverable or not, depending on how these transitions are implemented. In principle, it is possible to check the satisfaction of 4.3.1 for all the possible combinations in the election of those bars implemented as t^2 and those implemented as t^3. There exist several algorithms to reduce the number of combinations, but we consider them outside the area of this work and they are not discussed here.

Note that in general, for the same TM there may exist different choices (in the transitions implementable as t^2 or

t^3) that lead to recoverable PN's. On the other hand, there may be no choice that leads to a recoverable implementation. In this case, the handling is the same as in the previous case of no recoverable TM, that is, the structure of the TM must be changed.

In the following sections, several examples demonstrate the use of the tools developed.

4.4 Examples

Example 1:

Figure 4.1 shows an example of a TM. In this example the set S^2 is:

$$S^2 = [\ \langle F,B\rangle\ ,\ \langle F,C\rangle\ ]$$

the only transition between states of the set S^2 is the transition:

$$t4 = \langle F,B\rangle \rightarrow \langle F,C\rangle$$

Since B is not a legal state, the conditions of 4.3.2 are not satisfied. The conditions of 4.3.3 are also not satisfied since the only state, different from <B,F>, that includes <B> is <A,B>, but t3 exits <A,B> and:

$$MIN(t3) = MIN(t4)$$

Since 4.3.2 and 4.3.3 are not satisfied then there exist the conditions of statement 4.3.4. Then t4 can be of type t^2 or t^3. If t4 is implemented as a t^2 then the conditions 4.3.1 are not satisfied since state $\langle F,B\rangle \lessdot S^2$ has no exiting transitions of type t^3 and <B> is not a legal state. In

this case, the corresponding PN will be not recoverable. The PN is built using the method developed in chapter 3 and taking care to implement transition t4 as a t^2 transition. This PN is shown in figure 4.2. The corresponding ETM in figure 4.3 confirms that the PN is not recoverable since the illegal state B is terminal.

If t4 is implemented as a transition of type t^3 then the conditions of 4.3.1 are satisfied. State <F,C> is the only state from the set S^2 that has no exiting transitions of type t^3, but <C> is a legal state (condition 4.3.1(1)). There are no directed loops of transitions t^3, so that, 4.3.1(2) is obviously satisfied. In this case, the corresponding PN is recoverable. The PN is shown in figure 4.4 and the corresponding ETM in figure 4.5.

Example 2:

In this example, a recoverable PN corresponding to the TM of figure 4.6 is to be designed. In this TM, the states of the set S^2 are:

S^2 = [<F,F,A> , <F,F,B> , <F,F,C> , <F,A> , <F,B> , <F,C>]

The transitions between the states of the set S^2 are:

t6, t7, t8, t10, t11, t12.

The transition t6 can be implemented only as a transition of kind t^3 since:

t6 = <F,F,A> -> <F,F,B>

or:

t6 = <F,A> + <F> -> <F,B> + <F>

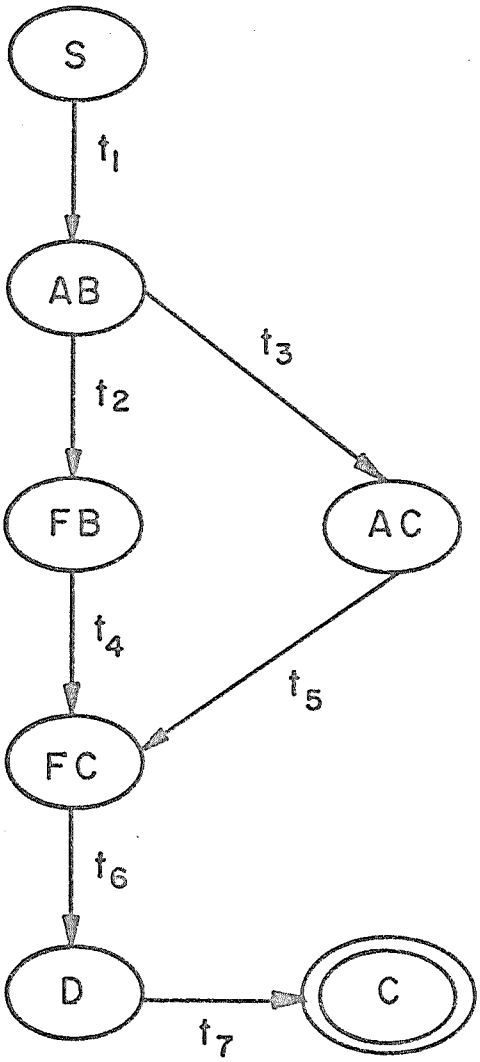


Figure 4.1: A Token Machine (TM)

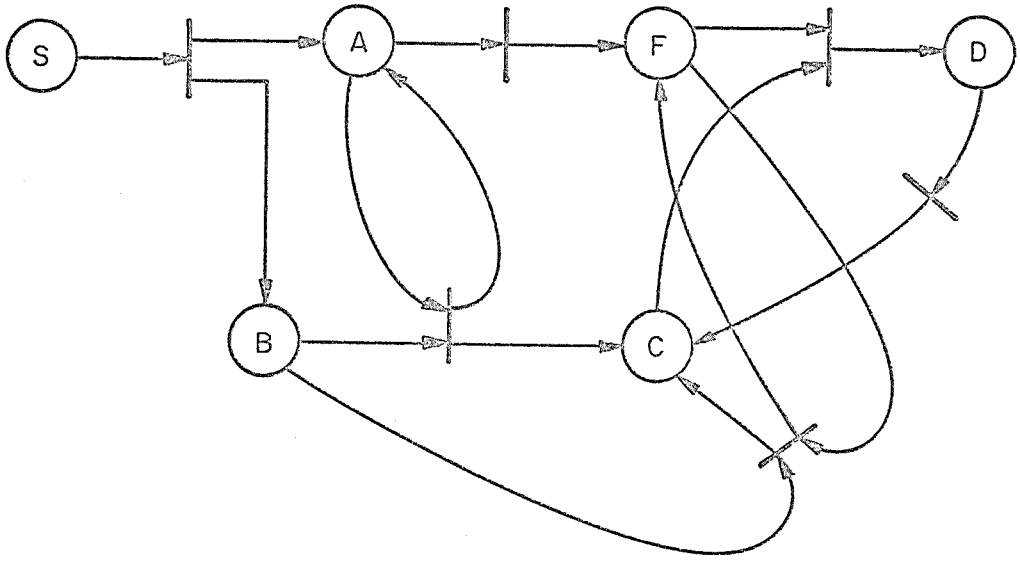


Figure 4.2: A PN for the TM of figure 4.1

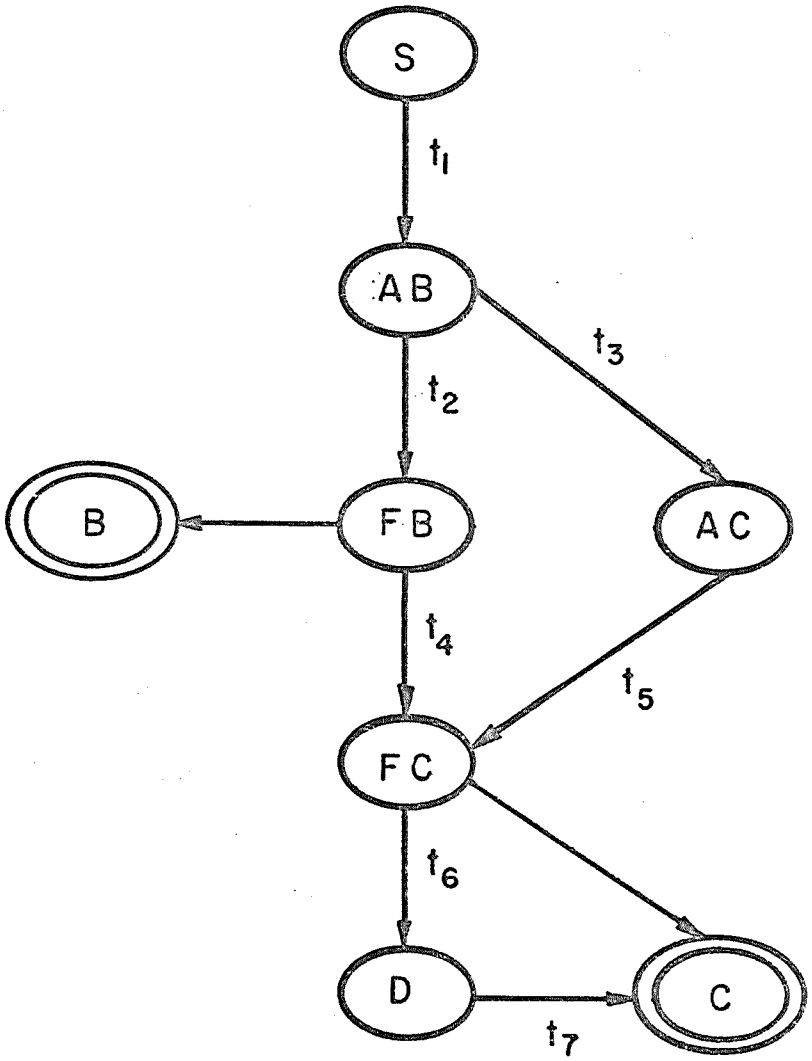


Figure 4.3: The ETM for the PN of figure 4.2

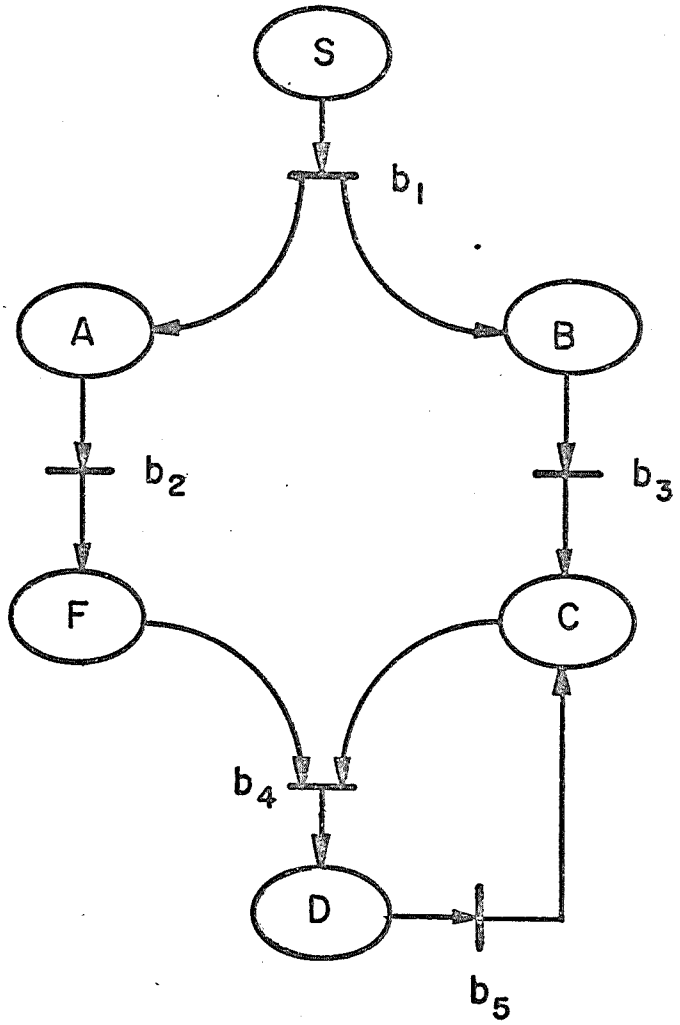


Figure 4.4: A PN for the TM of figure 4.1

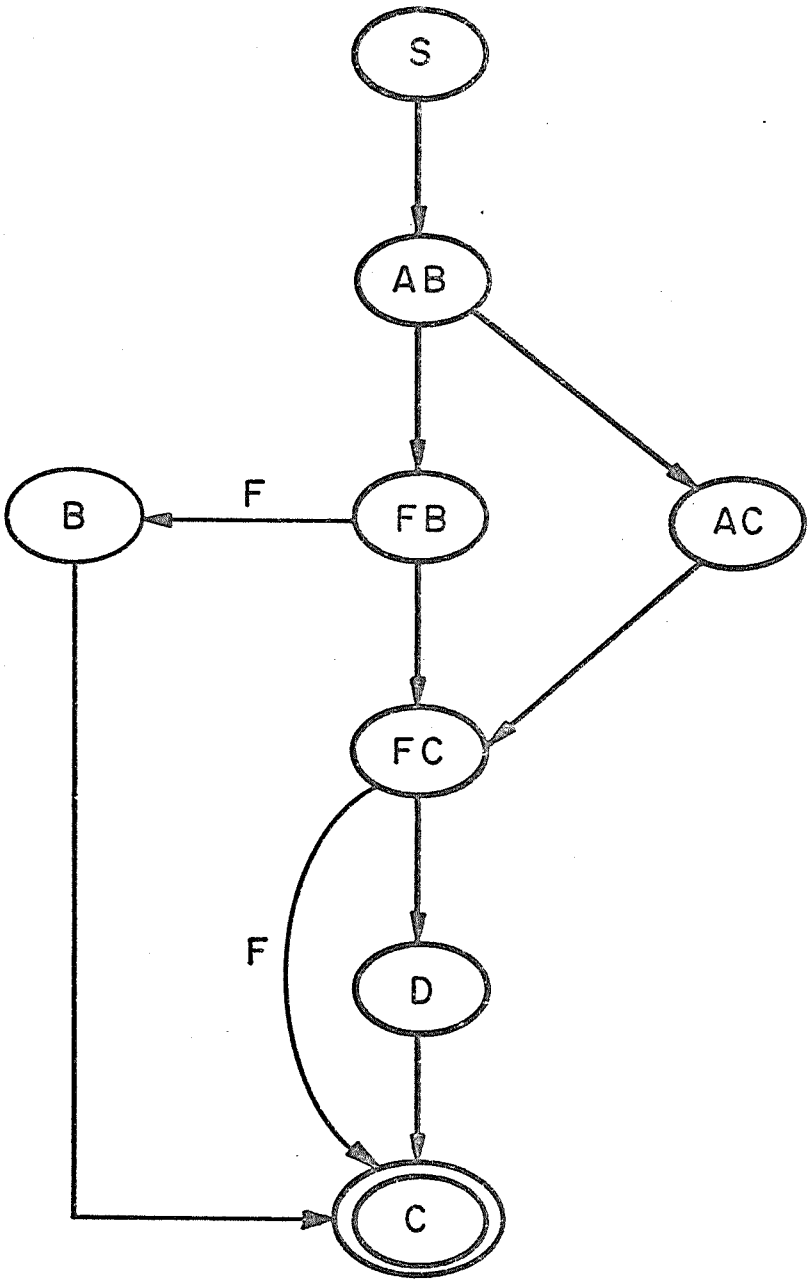


Figure 4.5: The ETM for the PN of figure 4.4

and there exists a legal transition:

$$t10 = \langle F,A \rangle \rightarrow \langle F,B \rangle$$

and this satisfies the conditions of statement 4.3.2. In the same way as t6, also t7, t8, t10, t11 and t12 can be implemented only as a transition of kind t^3. In this example, all the transitions between states of S^2 can be implemented only as transitions of kind t^3. Figure 4.7 shows the t^3 transitions of the TM of figure 4.6. In this case, there exist two loops of states of the set S^2, and there is no states of S^2 without exiting transitions t^3.

For the loop:

$$FFA \rightarrow FFB \rightarrow FFC$$

there exist corresponding legal states FA, FB, FC.

For the loop:

$$FA \rightarrow FB \rightarrow FC$$

there exist corresponding legal states A, B, C. Thus conditions 4.3.1 are satisfied for any implementation of the TM. An implementation for this TM is shown in figure 4.8.

Example 3:

in this example, the TM of figure 4.9 is implemented. The states of the set S^2 are:

$$S^2 = [\ \langle F,F,A \rangle\ ,\ \langle F,F,B \rangle\ ,\ \langle F,F,C \rangle\ ]$$

and the transitions between these states are:

$$t4,\ t5,\ t6$$

Transition t4 can be implemented as either t^2 or t^3, since:

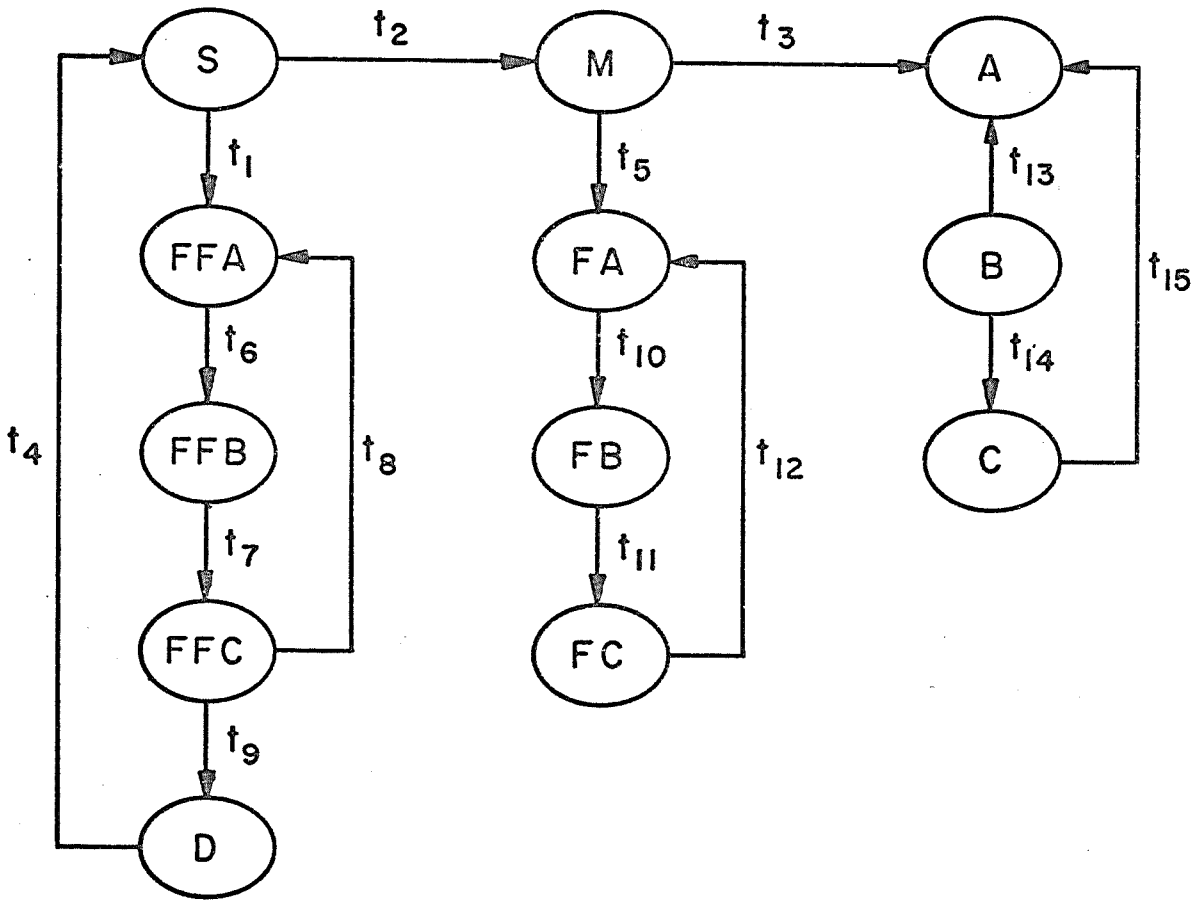


Figure 4.6: A Token Machine (TM)

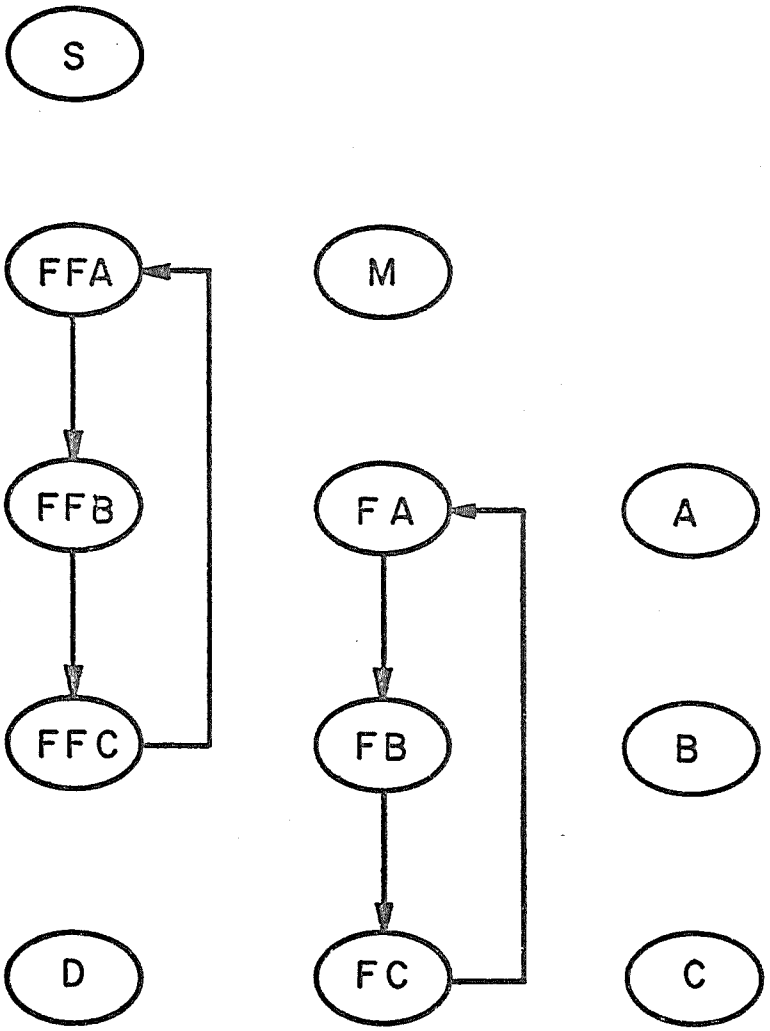


Figure 4.7: Transitions of kind t^3 for the TM of figure 4.6

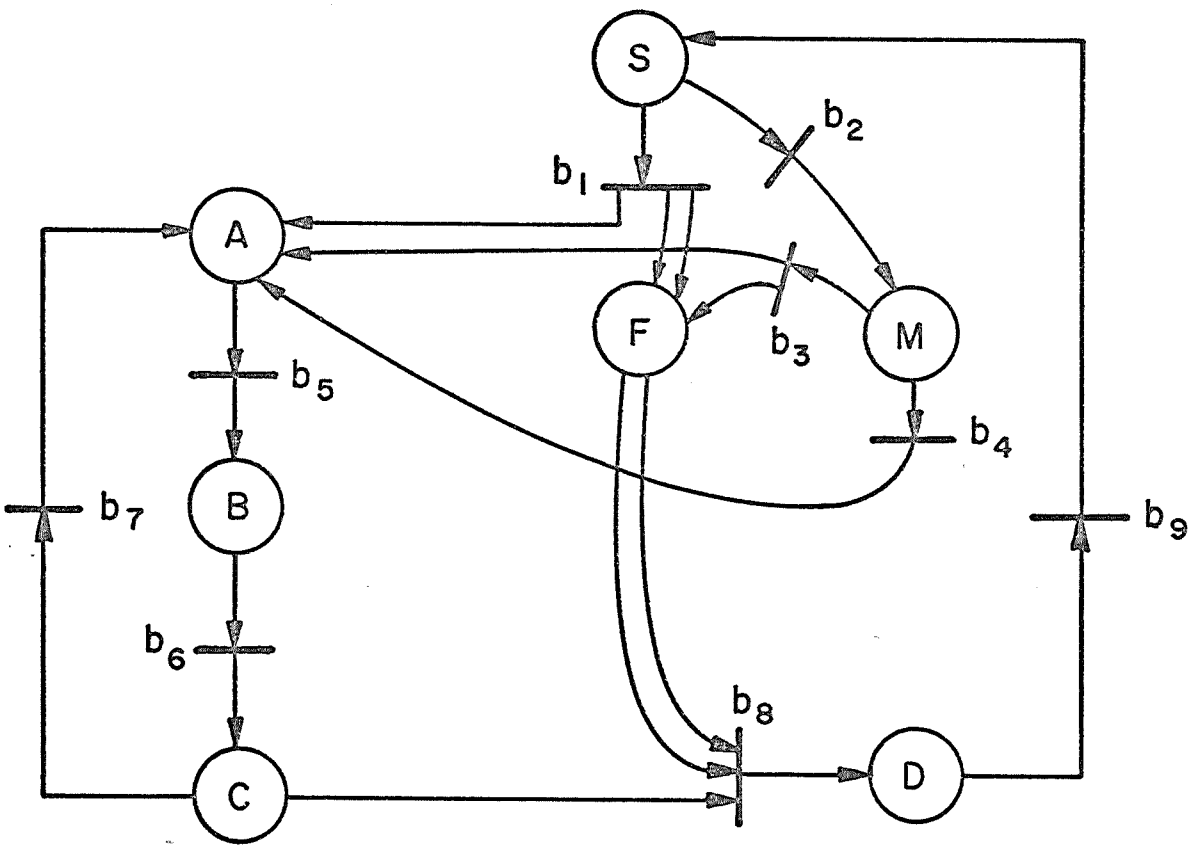


Figure 4.8: A PN for the TM of figure 4.6

$$t4 = \langle F,F,A\rangle \rightarrow \langle F,F,B\rangle$$

and there is no state included in <F,A> (4.3.2 is not satisfied) or state that includes <F,A> (4.3.3 is not satisfied). In the same way, t5, and t6 can also be implemented as t^2 or t^3.

Figure 4.10 shows the transitions t4, t5 and t6. If these three transitions are implemented as transitions t^3, there is a loop of states of kind t^3. But, since there is not a legal state <F,A>, <F,B> or <F,C> then condition 4.3.1(2) is not satisfied. On the other hand, if one of the transitions t4, t5, or t6 is implemented as a transition of type t^2 then there will exists a state S_k^2, without exiting transitions of type t^3, and without a corresponding state $S_k^2 - F$. This situation contradicts condition 4.3.1(1), and then there is no way of implementing the TM of figure 4.9 so that it is recoverable from single failures of kind "loss of token."

If we still desire to implement a recoverable process, the TM have to be changed. Several different modifications can be made in order to get a "similar" TM, but a recoverable one. These modifications changes not only the recoverability properties of the process, but also other properties. One possible modification is the previous example with the TM shown in figure 4.6. Another possible modification is described by the TM of figure 4.11. The corresponding recoverable PN is shown in figure 4.12. The

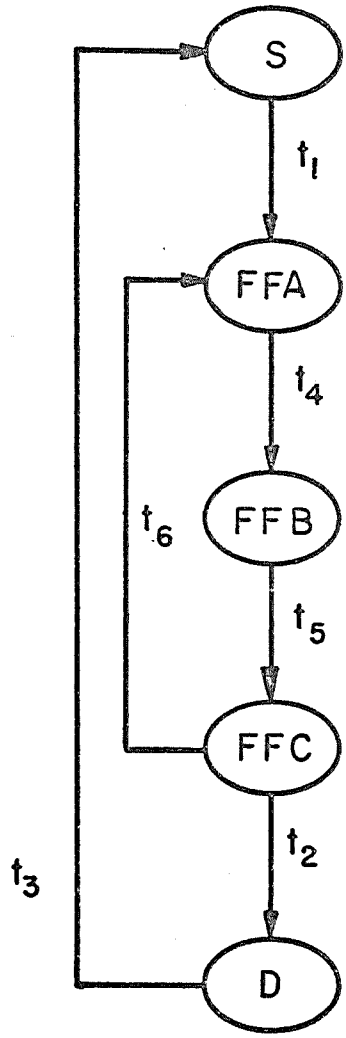


Figure 4.9: A Token Machine (TM)

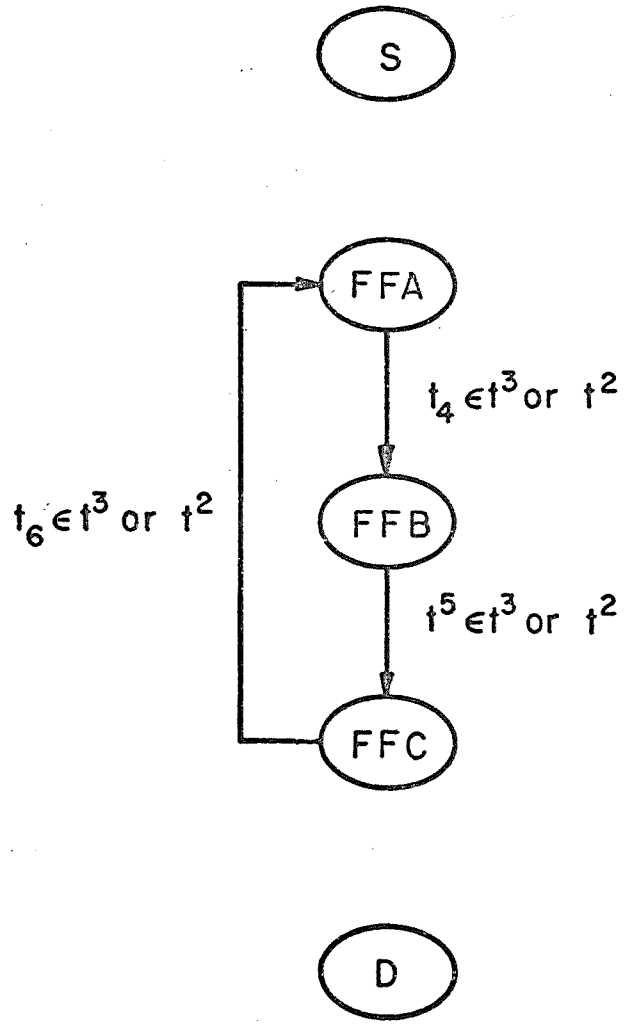


Figure 4.10: Transitions between states S^2 in the TM of figure 4.9

ETM of figure 4.13 confirm the recoverability of the PN.

This chapter described the structure that a TM must have in order to be implementable as a recoverable PN (4.3.1). In 4.3, we showed a way of designing a recoverable PN for any given TM that can be implemented in this way. If the TM is not implementable as a recoverable PN, the presented method points out the reasons that prevent such an implementation.

The next chapter analyzes the practical limitations of recoverable processes under the "loss of token". Since in the Petri-net model these limitations are found to be pragmatically unacceptable, a new model (the Time-Petri-net) is defined and studied in the next chapters. This new model appears to be very useful in the study of recoverability in real processes.

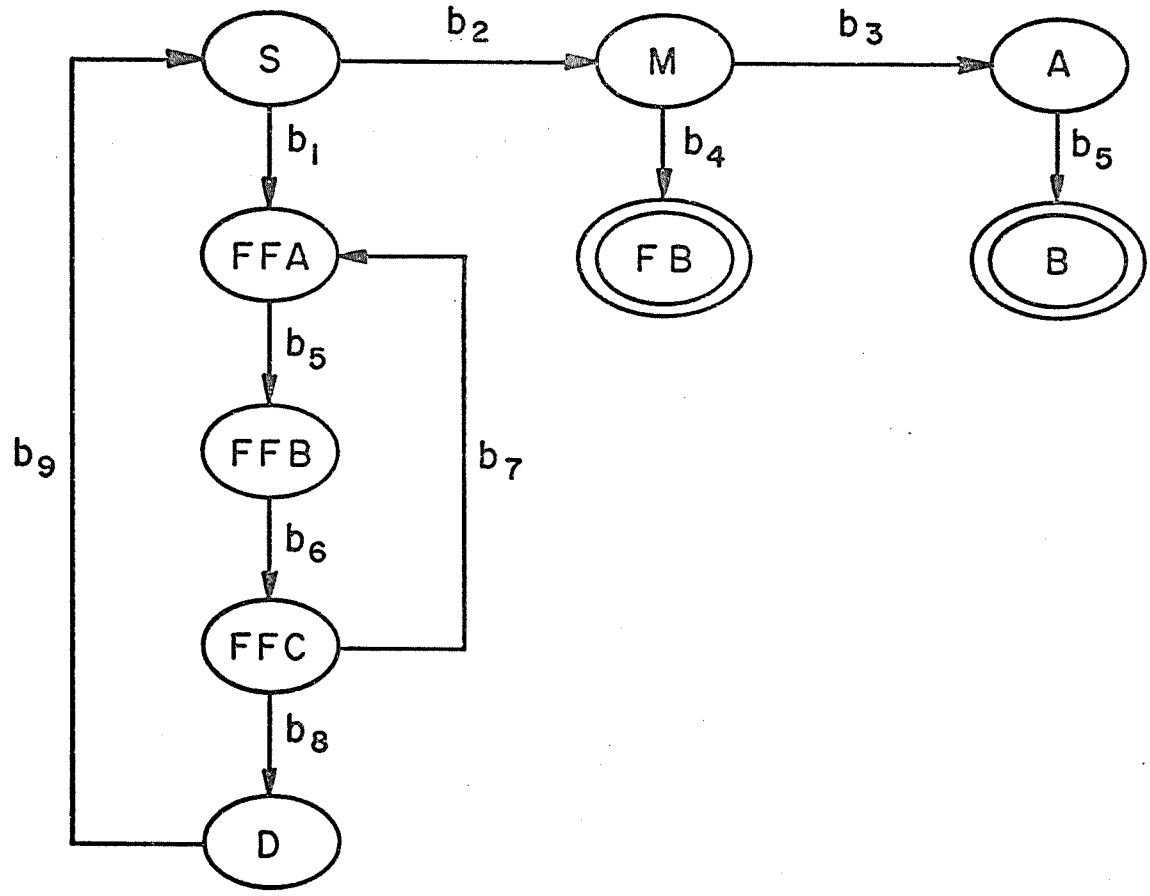


Figure 4.11: A recoverable TM

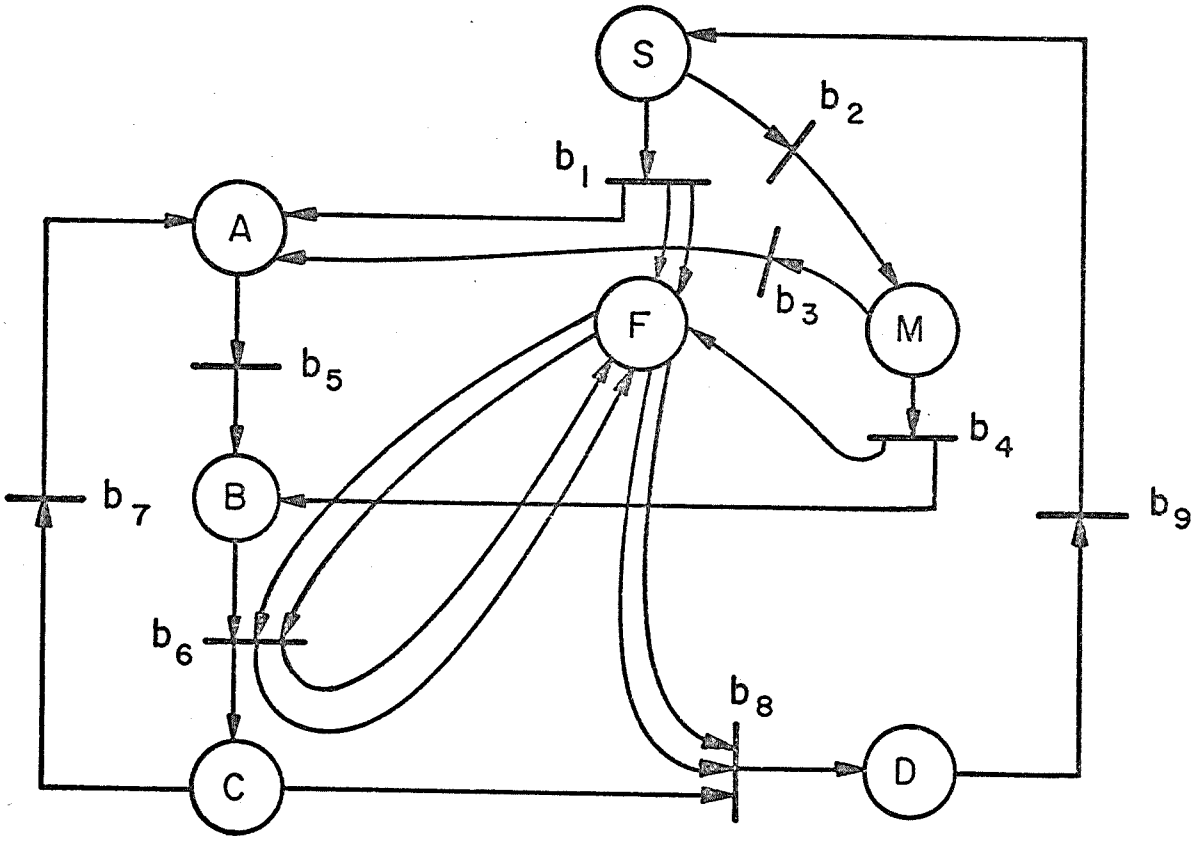


Figure 4.12: A PN for the TM of figure 4.11

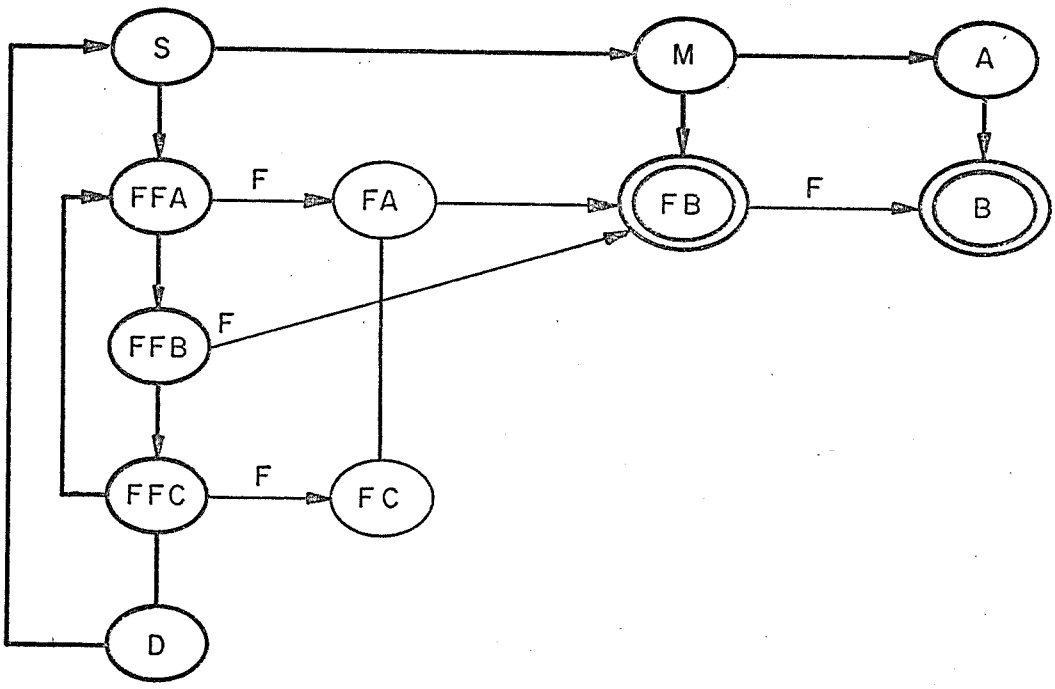


Figure 4.13: The ETM of the PN of figure 4.12

Chapter 5

PRACTICAL LIMITATIONS OF RECOVERABLE TM'S

In chapter 4 the structure of the TMs that can be implemented as recoverable processes (under loss of tokens) were studied. In general, the processes examined in the previous chapters were characterized by a lack of knowledge about the execution times of its parts. No assumption was made about the times expended by the events when they occur, or the relation between these times. In this chapter, it is shown that in the PN model, the recoverable TMs (under "loss of token") have certain properties, usually unacceptable in practical systems. These processes will be divided into different groups and each group will be studied separately.

Processes of Kind 1:

This kind of process is characterized by the following property: If Ai + F is a legal state then Ai is not a legal terminal state.

Processes of Kind 2:

To this group belong all the processes which do not belong to kind 1.

Note that, as in the previous chapters, we will deal

only with processes that have finite TMs.

5.1 Properties of Processes of Kind 1.

The recoverable processes of this kind have the following properties:

5.1.1 Theorem

Let a process be of kind 1 and recoverable. If A1 + F is a legal state, then there exists another state of type S^2, say state A2 + F , and a transition:

$$t_k = A1 + F \rightarrow A2 + F$$

Proof :

Suppose that such a transition does not exist. In this case all the transitions exiting from A1 + F are of type t^1. In 4.3 it is shown that, in this case, the process is recoverable only if A1 is a legal and terminal state. Since this contradicts the definition of process of "kind 1" then the transition t_k exists, and therefore the state A2 + F is legal.

Q.E.D.

5.1.2 Theorem

Let a process be of kind 1 and recoverable; then there exists at least one loop in the TM, so that all the members in the loop include the condition F.

Proof :

From theorem 5.1.1, each state that includes F has a successor that also includes F. It means that there are transitions:

A1 + F -> A2 + F ->..... ->Ai + F->.....

If there is not a loop, then for each i (i as big as we want) there is not a k smaller than i such that:

$$Ai = Ak$$

But, this means that in the TM there is an infinite number of different states. Since in this work we deal only with finite TM then a loop exists.

Q.E.D.

5.1.3 Theorem

Let a process be of kind 1 and recoverable. Then there exists at least one loop, such that all the states that are members of the loop have the same number of instances of F.

Proof :

We define the following notation:

1. [Pi] is the set of all the states in the PN that include at least one instance of F,
2. [Mi] is the set of the states in the PN that belong to directed loops of states of [Pi],
3. [Qi] is the set of all the states in the PN that satisfy:

   (a) Qi is a member of [Mi],

(b) if S1 is a member of [Mi], and exists a path of states of [Pi] from Qi to S1, then Qi has equal or more instances of F than Si has.

Theorem 5.1.2 shows that the set [Mi] is not empty. The set [Qi] is also not empty because the element of [Mi] with maximal number of instances of F always belongs to [Qi].

Suppose that from the set [Qi] we choose an element with minimal instances of F. If this element is denoted as Q1, and if it has k instances of F, then there exist:

$$Q1 = k.F + A$$

(A is a bag that not includes F).

If there occurs a failure in Q1, the PN goes to a state, say S1, given by:

$$S1 = (k - 1).F + A$$

In chapter 4 it was shown that if there exists a path from S1 to any state, say S2, then also exists a path from Q1 to S2 + F. This means that after a failure the PN can only go to states that have less than k instances of F.

Since the TM is finite, and the TM is of kind 1 and recoverable, then after the failure the PN can not reach any terminal state. This means that the states after the failure include loops of legal states. These loops do not include loops that all of its states belong to

[Pi], otherwise the definition of Q1 is contradicted. On the other hand, if there are not loops of elements of [Pi] then theorem 5.1.1 implies that the loops include only states that not include F. But, in chapter 4 (section 4.2) it was shown that if after a failure there exists a path of transitions then there also exists a correspondent path of legal transitions. Each state in this second path has one more instance of F than the correspondent state in the first path. This means that there exists a loop with just one instance of F in its states. This loop corresponds to the loop of states that not include F which exist after the occurrence of the failure in state Q1.

Q.E.D.

5.2 Properties of Processes of Kind 2

By definition, processes of kind 2 are characterized by the existence of a legal terminal state A1 corresponding to a legal state A1 + F.

Usually, part of the conditions of the terminal states of a process are used to notify the external world that the process has finished its execution, and the status in which the process ended. Suppose first that A1 + F is not a terminal state. Then when the process is in A1 + F , the external world will sense the same conditions as in A1. This means that the external world might assume that the

process is ended in A1.

On the other hand, if A1 + F is also a terminal state there exist two cases:

1. F is not sensed by the external world.

   In this case F is not necessary in the terminal state,and there is no reason to implement this state.

2. F is sensed by the external world.

   In this case, after a failure in A1 + F, the process is recoverable since it stays in a legal state (A1). But the error is spread to the external world because the external world senses F in the terminal state, and F has lost the token.

5.3 Properties of Both Kind of Processes

The following properties exist in all processes with finite TM.

Theorem :

In a finite TM, if the states A1 = i.F + Q and A2 = j.F + Q are legal states, and if i<j then there is no path directed from A1 to A2.

Proof :

There exists a k:

$$k > 0$$

so that:

$$j = k + i$$

Suppose that the path from A1 to A2 is implemented by

the successive firing of the bars:

$$b1,b2,\ldots\ldots\ \ldots,bm$$

In this case, b1 fires in A1 bringing the system to a legal state, say S1. But since

$$IC(b1) \leq A1 \leq A2$$

then b1 can fire also in A2 bringing the machine to a legal state S2. S2 is given by:

$$S2 = k.F + S1$$

Now, b2 can fire in S1, but in the same way it can fire in S2. This procedure can be applied again, so that when bm fires it brings the system to state:

$$A2 = j.F + Q$$

thus it can bring the system to a legal state:

$$A3 = (j + k).F + Q = (i + 2.k).F + Q$$

Now, the entire procedure can be applied again to the states A2 and A3. In this case there exists the legal state:

$$A4 = (i + 3.k).F + Q$$

Continuing in the same way, for any positive integer p we can arrive to a legal state:

$$Ap = (i + (p - 1).k).F + Q$$

Since all the Ap are different (they have increasing number of instances of F), the series of states Ap is infinite. In this case the TM is infinite.

Q.E.D.

5.4 Discussion

Theorem 5.3 shows that in finite TMs, if A1 + F and A1 are legal states, then there is no path from A1 to A1 + F. A path in the reverse direction may exist. This means that there exists an irreversible degradation in the process.

Since all the recoverable processes are characterized by the existence of correspondent legal states Ai and Ai + F, then all recoverable processes have the property of irreversible degradation. This limitation is unacceptable especially in the case of processes without terminal states. This is because the process never terminates, but simply degrades in the number of possible states in which it can stay. An example of this case of recoverable process is shown in figure 5.1. In this example, there are paths from the states FFA, FFB, or FFC to the states FA, FB, FC, A, B, and C, but not in the reverse direction. This process will never terminate. And, after a degradation, it will never return to any of the states D, S, FFA, FFB, or FFC.

Theorem 5.1.3 shows that in recoverable processes of kind 1 there exists a loop of states always having the same number of instances of F. Section 5.2 describes the limitations of the processes of kind 2. The two kinds of processes (kind 1 and kind 2) include all the processes with finite TM. This means that each recoverable process has at least the limitations of one of the kinds.

From the analysis just above, we can conclude that the

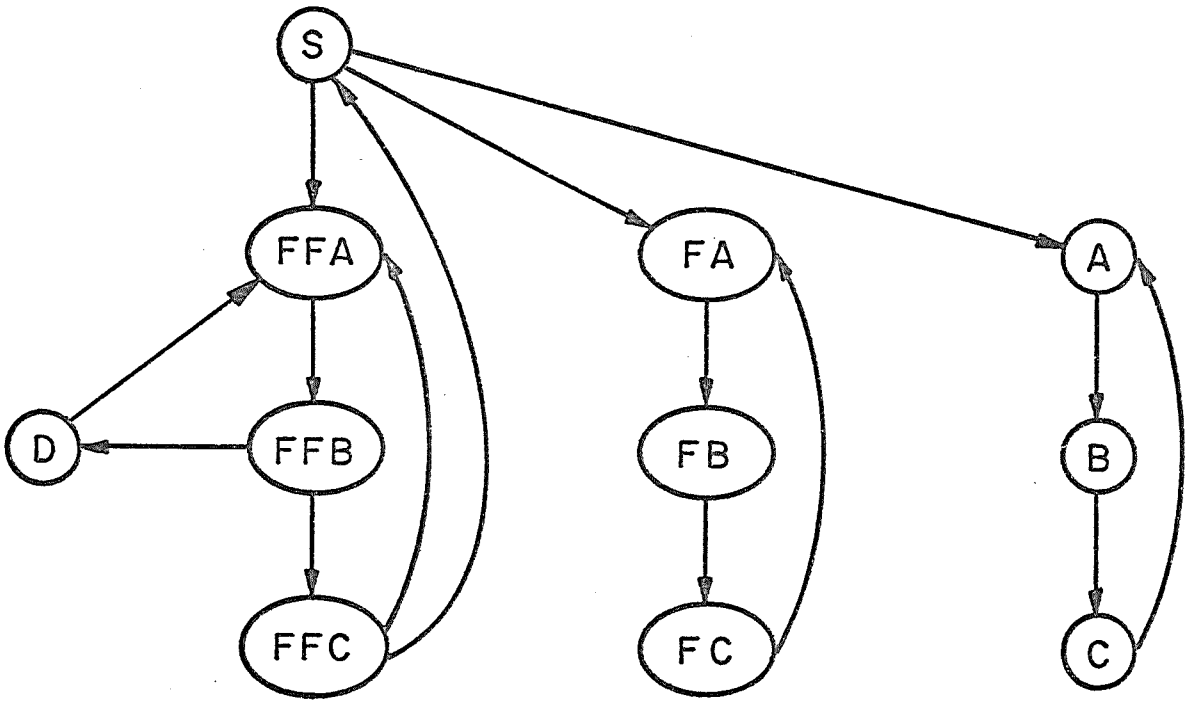


Figure 5.1: A Recoverable TM

recoverable processes represented by the model described in the previous chapters have a very constrained structure. These limitations to the structure are usually unacceptable in real systems. For example, most of the real recoverable processes do not degradate with each failure. After a failure, these processes try to recuperate automatically, and to regain the power that they had before the failure. Also the other limitations described in the previous sections usually do not exist in real systems.

But in the examples presented in chapter 2 these limitations were removed by postulating the existence of the function T ("can fire after a long time"). The function T is, in certain ways, related to the knowledge of some restrictions in the execution times of the different parts of the system. This fact indicates that some knowledge about the times in the system can remove the "bad properties" or "strong limitations" that exist in recoverable processes in the PN model.

In the following section, the concept of time is introduced into the Petri-net model of processes. Recoverability of processes is studied using this new model.

Chapter 6

THE TIME-PETRI-NET (TPN)

In this chapter a new model, the Time-Petri-Net (TPN), is defined and examined. This model contains the PN as a special case, and allows expression of the execution times of its parts.

The TPN is defined by a PN where each bar has two times specified. The first denotes the minimal time that must elapse from the time that all the input conditions of a bar are enabled until this bar can fire. The other time denotes the maximum time that the input conditions can be enabled and the bar does not fire. After this time, the bar must fire. In general, these two times give some measure of minimal and maximal execution times of the bars.

In the next section, the TPN is formally defined. Section 6.2 shows some properties of the TPN, and 6.3 describes a study of recoverability after a "loss of token" in this new model.

6.1 Definitions

6.1.1 a TPN is defined by a Petri-net (as defined in

section 3.2.10) in which for each bar bi there is given a tuple [t*i ; t**i]. For all i:

1. t*i, t**i real numbers
2. t*i, t**i $\geq$ 0
3. t*i < t**i

6.1.2 the firing algorithm in a TPN is defined as following:

1. If the conditions IC(bi) hold for a period of time equal to or greater than t*i then bi <u>can fire</u> (with the firing algorithm defined in 3.2.14).
2. If the conditions IC(bi) hold for a period of time equal to t**i then bi <u>fires.</u>

6.1.3 T^*_A is defined as the minimal time that the conditions of the bag A hold tokens.

6.1.4 T^{**}_A is defined as the maximal time that the conditions of the bag A hold tokens.

6.1.5 $T^*_b(S)$ is defined as the minimal time that the TPN has to stay at state S so that bar b can fire.

6.1.6 $T^{**}_b(S)$ is defined as the maximal time that the system can stay in state S before b fires.

6.1.7 T*(S) is defined as the minimal time that the TPN will stay in state S when it arrives at this state before a transition fires.

6.1.8 T**(S) is defined as the maximal time that the TPN may stay in state S before a transition fires.

6.2 Properties of the TPN

6.2.1 A TPN is a PN if for all bars i:

$$t^*i = 0$$

and $t^{**}i$ = infinite

In this case, a bar can fire at any time that its input conditions hold. This is the definition of the firing algorithm for a PN (section 3.2.14).

6.2.2 If S1 is a legal state in a TPN and in its corresponding PN, and if b1 can fire from S1 in the TPN, it also can fire from S1 in the PN.

This property exists because the firing algorithm in a TPN satisfies the conditions of the firing algorithm in the PN.

6.2.3 The converse of 6.2.2 is not always true. Figure 6.1(a) shows an example of a PN. If A is a legal state then b1 or b2 can fire. Figure 6.1(b) shows a TPN built on this PN. In this case, if A holds a token then b1 has to fire before 5. But b2 can fire only after 6. Thus in this case, b1 will always fire before b2. In this case, b2 never fires.

6.2.4 Applying successively the property 6.2.2, each sequence of legal states that exists in the TPN also exists in the corresponding PN.

6.2.5 Suppose that IC(b1) $\leq$ Si and IC(b2) $\leq$ Si, and Si is a legal state. In a PN, b1 or b2 can fire in Si.

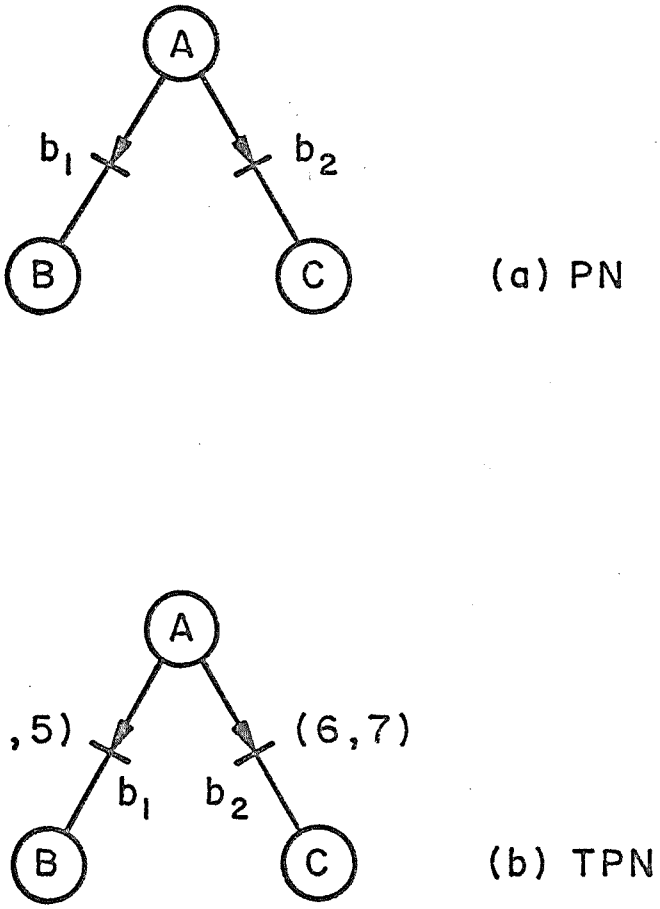


Figure 6.1: (a) a PN; (b) a TPN of the previous PN

But if:

$$T^{*}_{b1}(Si) > T^{**}_{b2}(Si)$$

then b1 never fires in Si. In state Si, b2 will always fire before b1, and the TPN will leave state Si before b1 can fire. (The example of figure 6.1(b) shows this situation).

6.2.6 From definitions 6.1.1 and 6.1.5:

$$T^{*}_{bi}(Sj) \leq t^{*}bi$$

for any i and j.

6.2.7 From definitions 6.1.1 and 6.1.6:

$$T^{**}_{bi}(Sj) \leq t^{**}bi$$

for any i and j.

6.2.8 Suppose that b1,b2,.....,bp is the set of all bars that satisfy:

$$IC(bi) \leq Sj$$

then T**(Sj) is given by:

$$T^{**}(Sj) = \min(T^{**}_{b1}(Sj);\ T^{**}_{b2}(Sj);.....T^{**}_{bp}(Sj)) \qquad (6.2.8.1)$$

because the first bar that arrives at its maximal waiting time (t**) must fire.

Replacing 6.2.7 in 6.2.8.1:

$$T^{**}(Sj) \leq \min(t^{**}b1;\ t^{**}b2;.....t^{**}bp) \qquad (6.2.8.2)$$

(6.2.8.2) gives an upper bound to the value of the

maximal time that the TPN can be in state Sj. This upper bound is not the minimal, but it is easy to compute since the values t**bi are given in the definition of the TPN. The exact value of T**(Sj) is given by (6.2.8.1), but there are practical cases in which the values $T^{**}_{bi}(Sj)$ are dificult to compute.

6.2.9 Q(B) is defined as the set:

$$Q(B) = [Q1;\ Q2;\ .....;Qp]$$

each element of the set is an ordered, finite or infinite,sequence of bags.

Qi is given by:

$$Qi = [Qi^1;\ Qi^2;.....;Qi^j;..]$$

Each element of Qi is a legal state in the TPN that satisfies:

$$B \leq Qi^j$$

Each Qi represents a possible sequence in the TPN. In Qi the bag B holds tokens, and there are no transitions in the sequence such that if bar bk fires in Qi^j then:

$$B \not\leq Qi^j - IC(bk) \qquad (6.2.9.1)$$

This means that in Qi, there is <u>no</u> transition such that during its execution, B does not hold tokens. The sequences Qi are chosen so that they are of maximal length. Thus if Qi and Qj are members of

Q(B) then:

$$Qi \not\leq Qj$$

The set Q(B) includes all the possible sequences that satisfy the previous constraints.

Using this definition, the maximal time that B can hold tokens satisfies:

$$T^{**}_B \leq \max(T^{**}(Q1^1)+T^{**}(Q1^2)+..;T^{**}(Q2^1)+..;T^{**}(Qp^1)+..) \quad (6.2.9.2)$$

Note that T^{**}_B may be infinite if one of the Qi have an infinite number of elements. This happens if there exists a loop of states such that all of them include B and in the loop there are no transitions that satisfy (6.2.9.1).

6.3 Recoverability of TPN After a Loss of Token

In this section we show how processes that are not recoverable in the PN model can be transformed into recoverable processes using the TPN model.

Suppose that a process, that is not recoverable after the loss of a token in F, is given by its TM. Our goal is to build a TPN so that its possible states and transitions are equal to those in the given TM. If the TM is implemented by a PN, then 4.3 shows that the process is not recoverable after a "loss of token" only if either:

1. there exist directed loops of states Ai + F

connected by transitions of type t^3, and there is not a corresponding legal state Ai, or

2. there are states Ai + F , without exiting transitions of type t^3, and Ai is not a legal state.

Suppose that in the TM, if there exist loops of states Ai + F and there are no corresponding legal states Ai, then <u>not all</u> of the transitions of the loop are <u>only</u> implementable as of type t^3 (see 4.3.2). In this case, at least one of the transitions in the loop can be effected as of type t^2. This means that the TM can be implemented by a PN with an ETM that has no loops of illegal states. If the TM is implemented in this way then the PN is not recoverable only if:

there are states Ai + F , without exiting transitions of type t^3, and Ai is not a legal state.

In this structure, after the occurrence of a failure, the process will terminate in an illegal state.

In order to transform the process to a recoverable one, for each illegal terminal node Ai we have to implement a bar bi that fires in Ai. This bar must execute a transition from Ai to a legal state in the TM, say Si. This means that:

1. IC(bi) $\leq$ Ai, and
2. IC(bi) includes all the instances in Ai that are not in Si.

On the other hand, bi is <u>not</u> allowed to fire in any

legal state. This means that bi does not affect the execution when there is no failure, so that the TM is normally executed. In order to disable the firing of bi during normal execution, t*i has to satisfy:

$$t^*i > T^{**}_{IC(bi)} \qquad (6.3.1)$$

Note that if there exist loops such that all the states in the loop contain IC(bi), then the implementation of the recoverable TPN must be such that $T^{**}_{IC(bi)}$ is not infinite. The example in section 6.4 shows this situation.

The method described in this chapter allows one to build a recoverable TPN for certain TMs that are not implementable by any recoverable PN. The method does not solve this problem in general, but it gives an acceptable solution in many practical cases, as shown in chapter 7.

6.4 Example

Figure 6.2 shows a TM that has to be implemented such that it is recoverable in the case that a loss of token occurs in the condition 5. One possible implementation is the PN shown in figure 6.3. Figure 6.4 shows the ETM corresponding to this implementation. The number on each arc denotes the bar that implements the corresponding transition. In this implementation there are two problems:

1. a loop of the illegal states 24 and 25. This loop can be broken if bar 4 is not allowed to fire in 24. But bar 4 has to fire in 245, 234, 244, and 246.

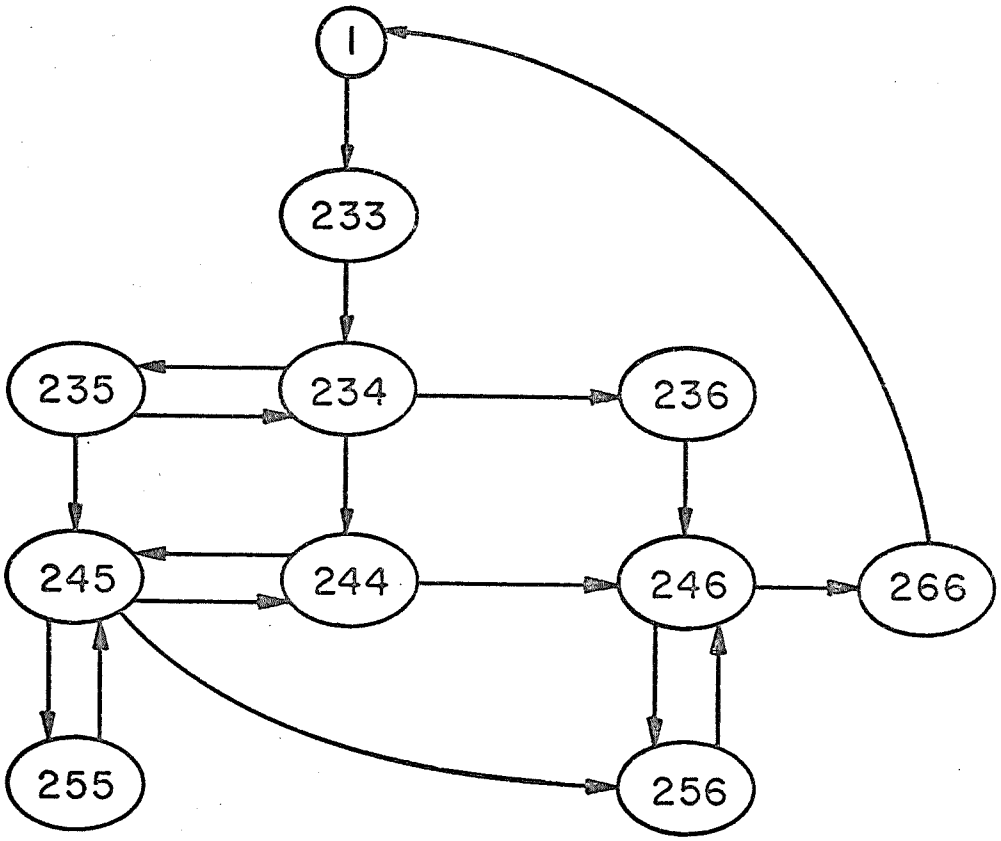


Figure 6.2: A Token Machine (TM)

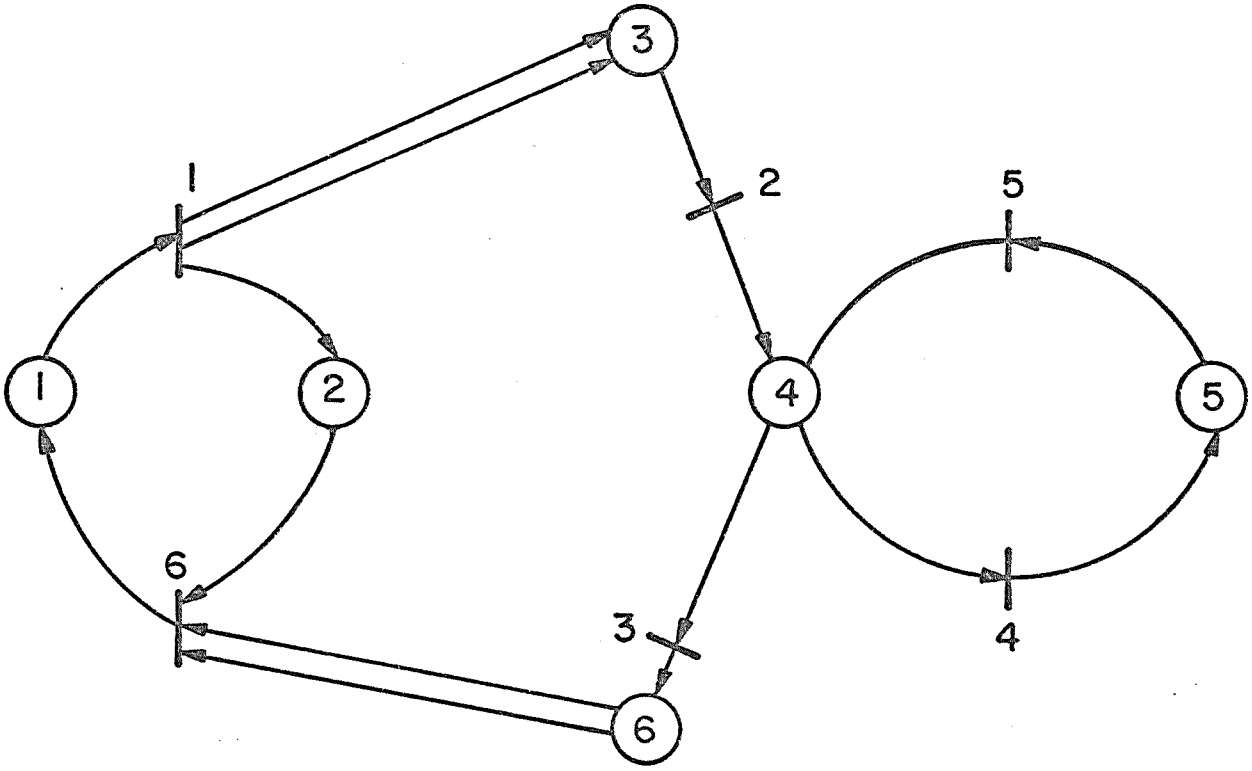


Figure 6.3: A PN for the TM of figure 6.2

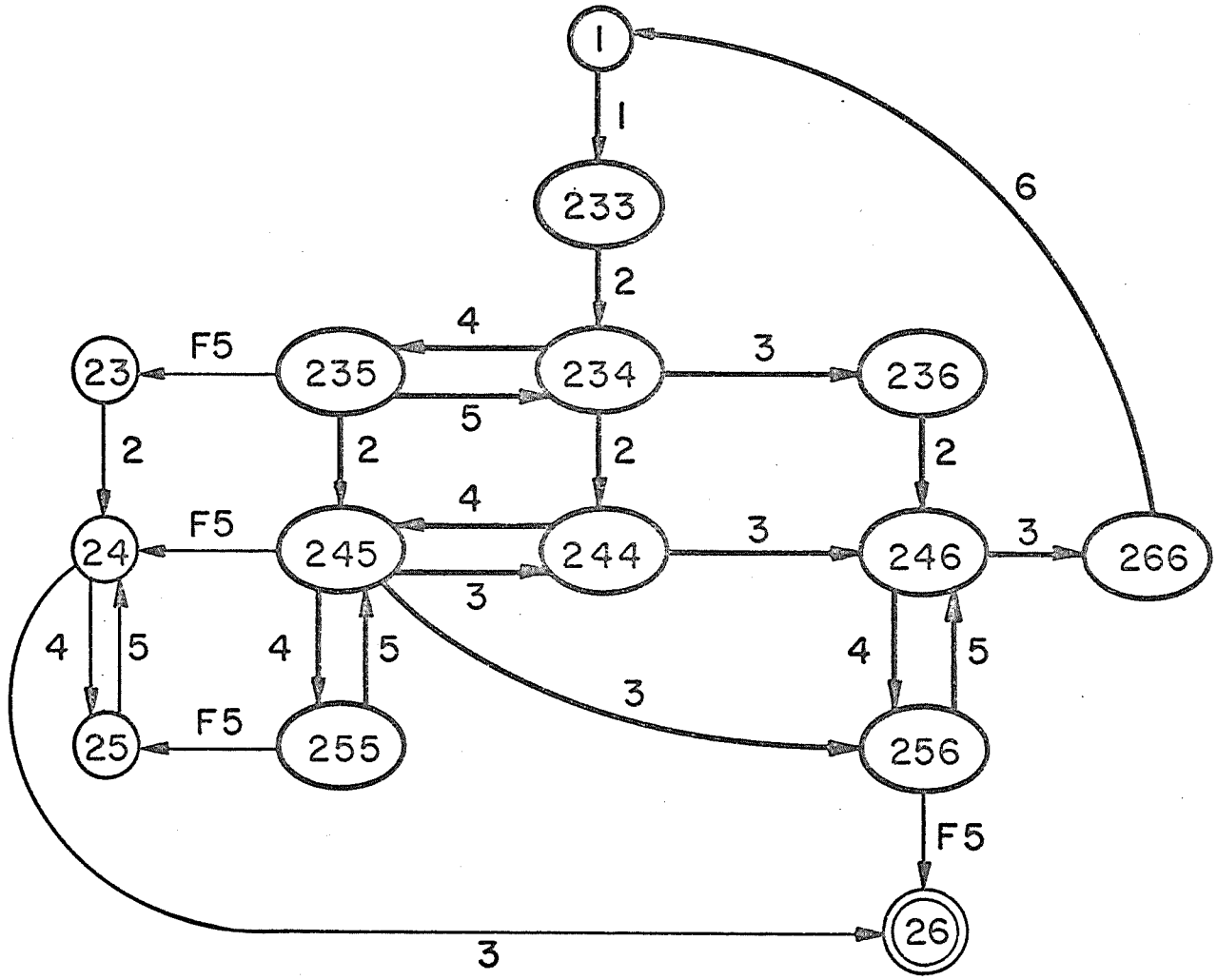


Figure 6.4: The ETM for the PN of figure 6.3

Instead of bar 4 we will implement four different bars:

1. $IC(b4^1) = 45$
2. $IC(b4^2) = 43$
3. $IC(b4^3) = 44$
4. $IC(b4^4) = 46$

These four bars implement the same transitions that bar 4 implements, but they can not fire in state 24.

2. The state 26 is illegal and terminal. But, T^{**}_{26} is infinite because of the loop between the states 246 and 256 connected by the transitions executed by the bars 4 and 5.

But with the new bars in place of 4, the transition from 246 to 256 is executed by 4^4 instead of bar 4, and this problem is also solved. In this case, bar 4^4 removes the token from 6 and places a new token. This means that the maximal existence time of 26 (T^{**}_{26}) is broken when bar 4^4 fires.

Figure 6.5 shows the new implementation of the TM, after bar 4 was split into four different bars. Figure 6.6 shows the ETM for the PN of figure 6.5. This ETM shows that there is only one illegal terminal state, the state 26. This means that we have to implement a bar that fires in 26. The input conditions of this bar are one of the three

following possibilities:

1. IC(7) = 2
2. IC(7) = 6
3. IC(7) = 26

In our example we choose the last possibility. This means that IC(7) = 26.

As shown in (6.3.1), t*7 has to satisfy:

$$t^*7 > T^{**}_{26}$$

The next step is to compute T^{**}_{26}, or at least an upper bound of T^{**}_{26}. In the next steps we will follow the procedure described in 6.2.9.

The possible sequences of states that include 26, and that satisfy the constraints explained in 6.2.9 are:

Q1 = [236; 246; 266]

Q2 = [256; 246; 266]

From 6.2.8:

1. $T^{**}(236) \leq t^{**}2$
2. $T^{**}(246) \leq \min(t^{**}3 ; t^{**}4^4)$
3. $T^{**}(266) \leq t^{**}6$
4. $T^{**}(256) \leq t^{**}5$

and using (6.2.9.2):

$$T^{**}_{26} \leq \max(t^{**}2+\min(t^{**}3;t^{**}4^4)+t^{**}6 \; ; \; t^{**}5+\min(t^{**}3;t^{**}4^4)+t^{**}6)$$

Thus, if:

$$t^*7 > \max(t^{**}2+\min(t^{**}3;t^{**}4^4)+t^{**}6 \; ; \; t^{**}5+\min(t^{**}3;t^{**}4^4)+t^{**}6) \qquad (6.4.1)$$

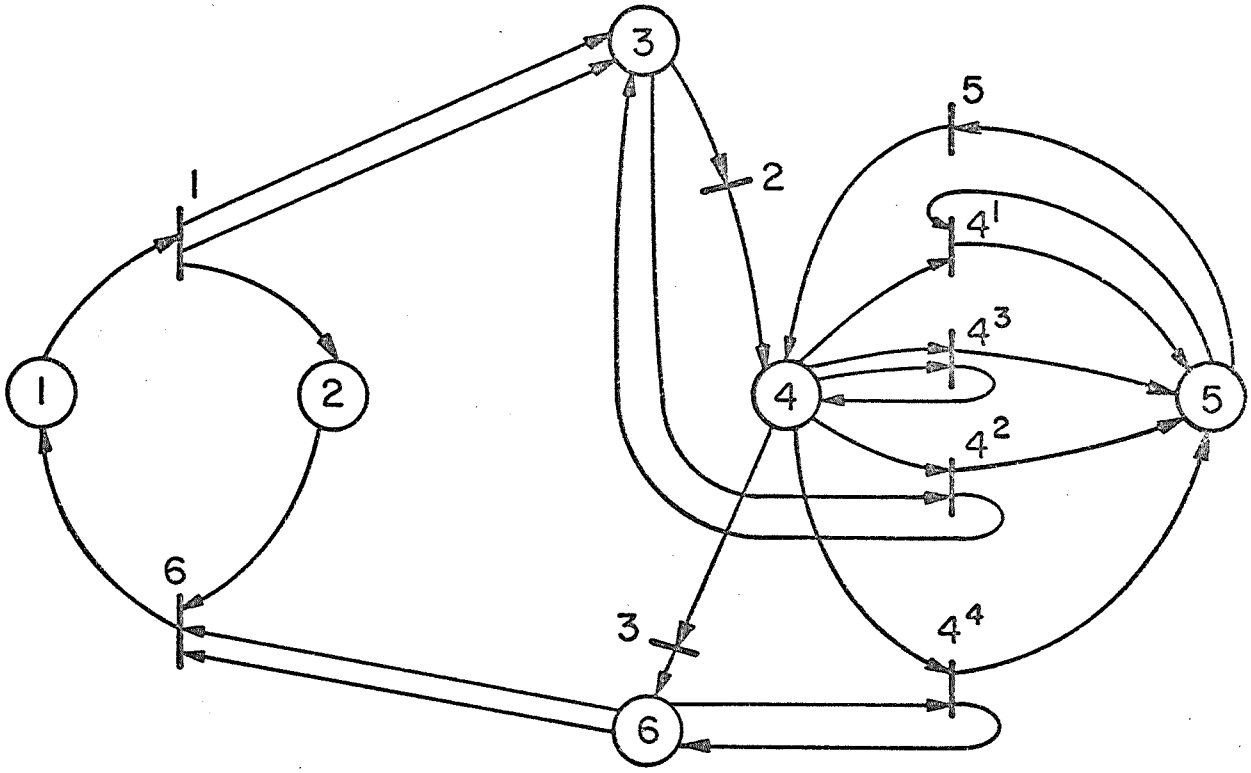


Figure 6.5: A PN for the TM of figure 6.2

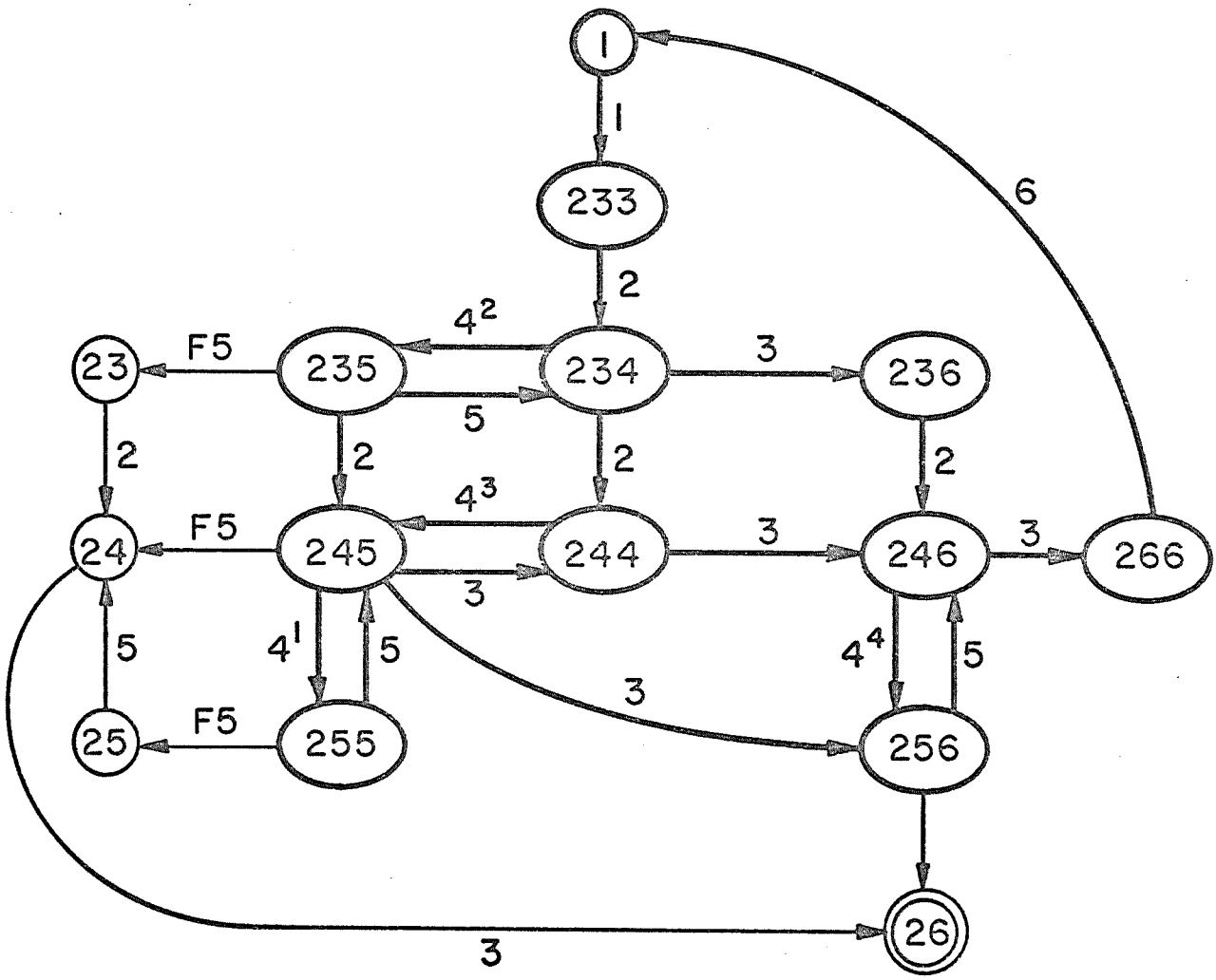


Figure 6.6: The ETM for the PN of figure 6.5

then:

$$t^*7 > T^{**}_{26}$$

And if:

$$OC(7) = Si$$

where Si is one of the legal states, then the process is recoverable. In our example we choose:

$$OC(7) = 1$$

Figure 6.7 shows the TPN that implements the recoverable process of the given TM. We assume that the values of $t^{**}2$, $t^{**}6$, $t^{**}5$ and either $t^{**}3$ or $t^{**}4$[4] are finite, and that t^*7 is chosen so that (6.4.1) is satisfied. The TPN of figure 6.7 implements the TM of figure 6.2 and is recoverable in case of loss of token in condition 5. After a failure, the system will arrive in state 26. After the process is in state 26 for a time equal to t^*7, then bar 7 may fire and the TPN will return to legal state 1. The return to a legal state in finite time can be insured by setting a <u>finite</u> $t^{**}7$.

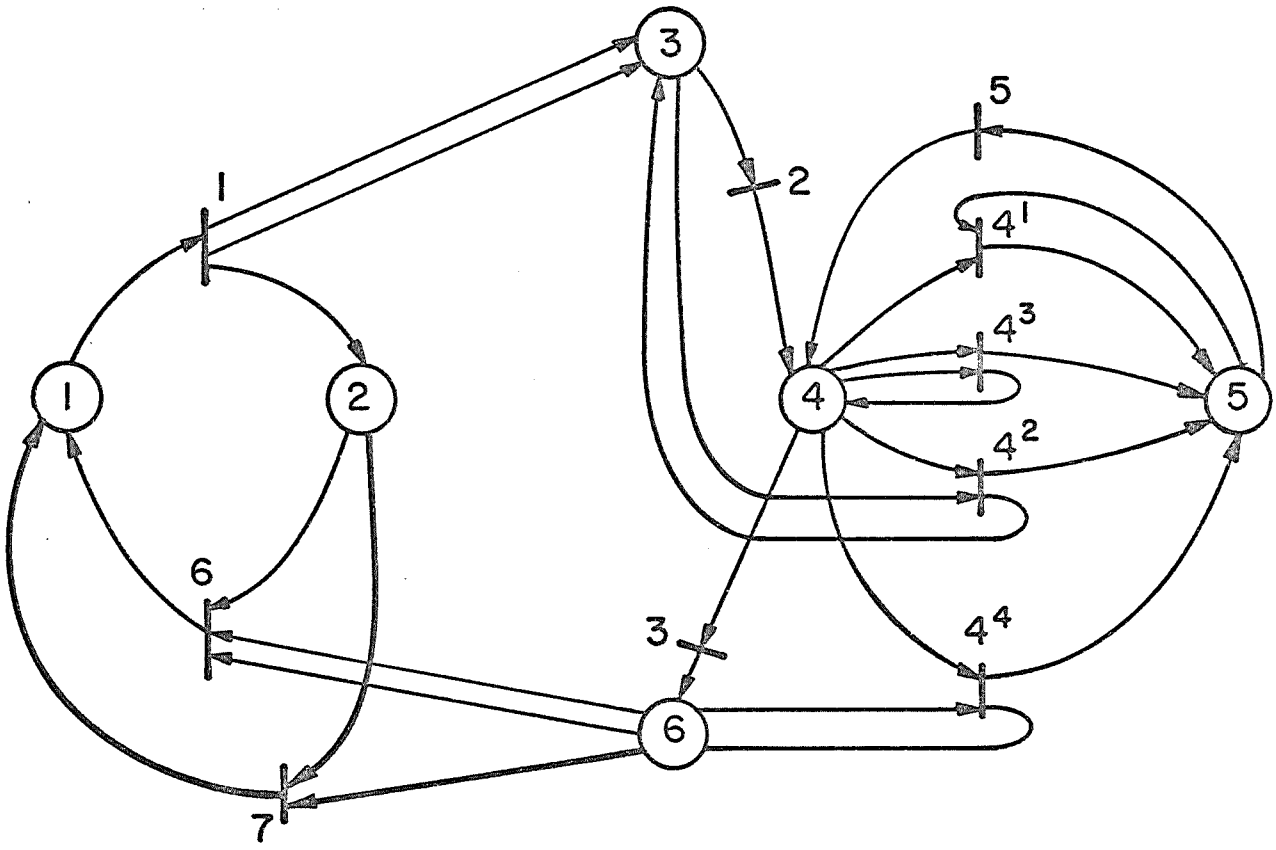


Figure 6.7: A recoverable TPN for the TM of figure 6.2

Chapter 7

A PRACTICAL EXAMPLE -
Recoverability of a Communication Protocol

The study of the communication protocols in this dissertation is motivated by practical reasons. During the last few years, many computer networks have been designed and implemented. Since the probability of failures in the communication links is relatively high, the implementation of recoverable protocol processes is of considerable importance.

The presentation in this section is based, in part, on the study presented in [POST74] and on the examples given in chapter 2. The new model, the TPN, is used. The examples given here are a simplified model of the IMP-IMP protocols used in the ARPANET.

7.1 Example 1

In this section the protocol of figure 7.1 is studied. This protocol was presented in chapter 2. We suppose that a possible failure is the loss of the message M. That is, a token in M can disappear. The dotted line from E to A

represents the preparation of a new message by the sender. The dotted line from D to B represents the receiving process.

The ETM of the PN of figure 7.1, assuming initial state AB, is given in figure 7.2. The ETM shows that there exist two illegal states, WB and WD. In this ETM, there is no loop of illegal states, but there exists an illegal and terminal state, WB. Chapter 6 shows that in order to transform such a process to a recoverable process, there has to be a bar that fires in state WB. If this bar is called 7, then there exists the following possibilities:

1. IC(7) = WB
2. IC(7) = B
3. IC(7) = W

In the first possibility, bar 7 is dependent on both the sender and the receiver. In real systems this structure is difficult to implement because of the physical distance between sender and receiver. In our example, we choose the third possibility. In this case bar 7 is dependent only on the state of the sender. In case of a failure, the sender will again send a transmition of the lost message. This means that:

$$IC(7) = W$$

and:

$$OC(7) = MW$$

so that in case of a failure the system will return to state

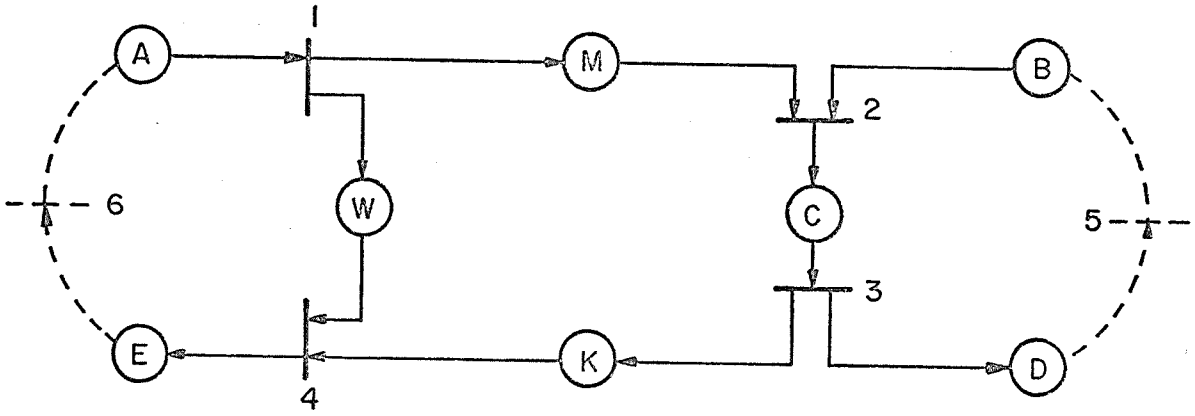


Figure 7.1: A PN of a protocol process

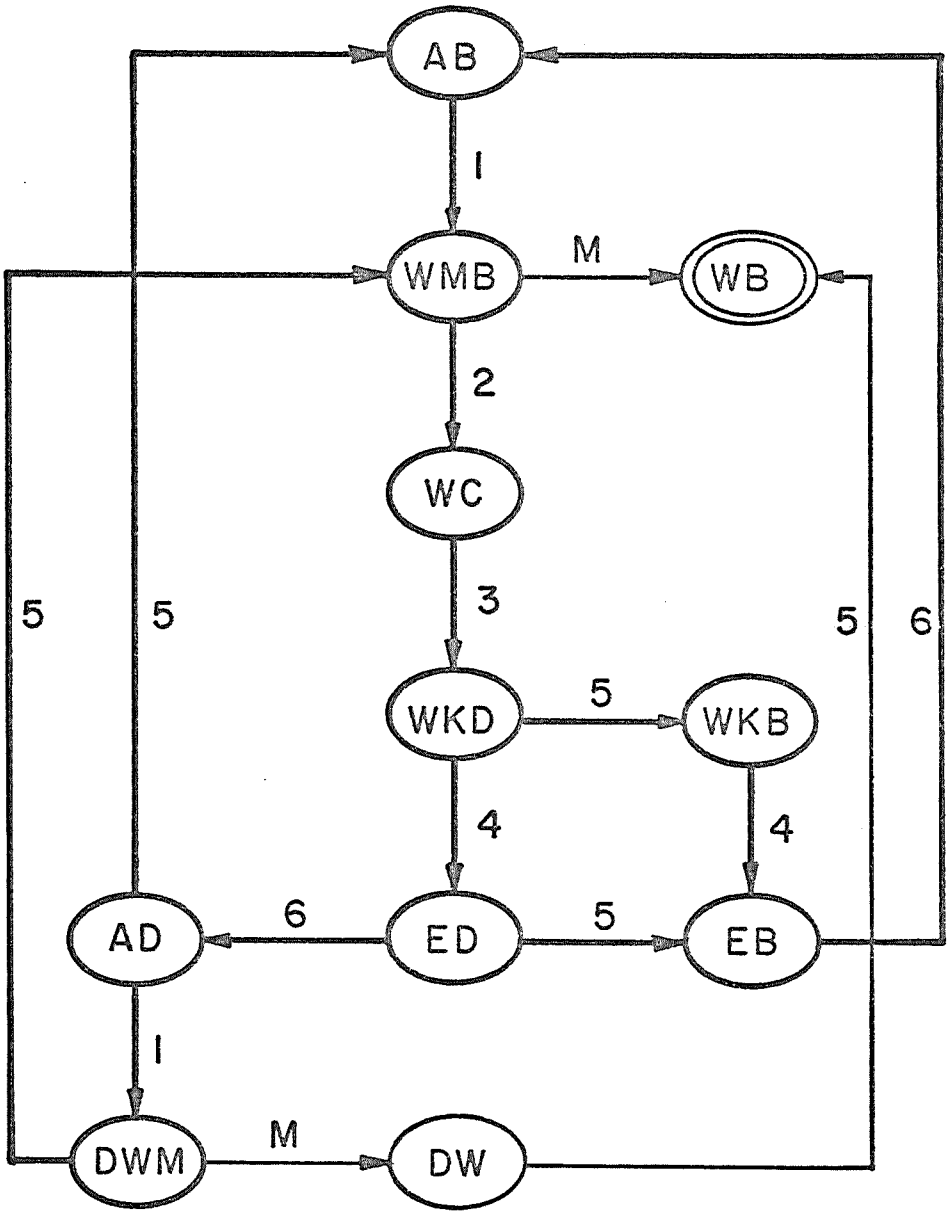


Figure 7.2: ETM for the PN of figure 7.1

WMB.

On the other hand, bar 7 is to fire only if a failure has occurred. In other words:

$$t^*7 > T^{**}_W \qquad (7.1.1)$$

From figures 7.1 and 7.2 it is possible to show that if:

1. $T^*_4(WKD) < T^{**}_5(WKD)$ (bar 4 can fire in WKD)
2. $T^*_6(DE) < T^{**}_5(DE)$ (bar 6 can fire in DE)

and 3. $T^*_1(AD) < T^{**}_5(AD)$ (bar 1 can fire in AD)

(7.1.2)

then the PN can arrive at state WMD, and then:

$$T^{**}_5(WMD) = t^{**}5 - (t^*4 + t^*6 + t^*1)$$

and:

$$T^{**}_W = t^{**}5 - (t^*4 + t^*6 + t^*1) + t^{**}2 + t^{**}3 + t^{**}4 \qquad (7.1.3)$$

Note that after the PN arrives at state WKD, the maximal time that W can hold is the same if first bar 4 fires, or that first bar 5 fires and then 4 fires. This time is equal to the maximal time until bar 4 fires, in this case, $t^{**}4$.

If t^*7 is set such that:

$$t^*7 > t^{**}5 - (t^*4 + t^*6 + t^*1) + t^{**}2 + t^{**}3 + t^{**}4$$

then the PN is recoverable (t^*7 satisfies (7.1.1) in which T^{**}_W is replaced with (7.1.3)). If one of the conditions (7.1.2) is not satisfied, the PN will never arrive at state WMD. Then the case is that:

$$T^{**}_W = t^{**}2 + t^{**}3 + t^{**}4$$

and t^*7 is to satisfy:

$$t^*7 > t^{**}2 + t^{**}3 + t^{**}4$$

The recoverable TPN and its corresponding ETM are shown in figures 7.3 and 7.4 respectively. This TPN is recoverable from failures of type "loss of token" in M. Note that if (7.1.1) is not satisfied then the ETM is infinite and the process is not recoverable. In many practical systems, the t*7 that satisfies (7.1.1) can be very large. In these cases, the protocol of the next example can be used.

7.2 Example 2

Suppose that each message carries a sequence number. If these numbers are from the set of integers [1,2,...,n] then the messages are sent sequentially in the order:

1;2;..n;1;2;...n;1;2....

In the PN that represents this protocol, there exist different conditions Mi ; (i = 1,2,...n). Each Mi corresponds to the message carrying the sequence number i.

In the same way, for each i (i = 1,2,...n) there exist the conditions:

Ai = ready to send message i

Bi = ready to receive message i

Ki = acknowledge to message i is sent

Wi = waiting for acknowledge to message i

Ei = acknowledge to message i was received

Ci = message i was received

Di = prepare for receiving next message

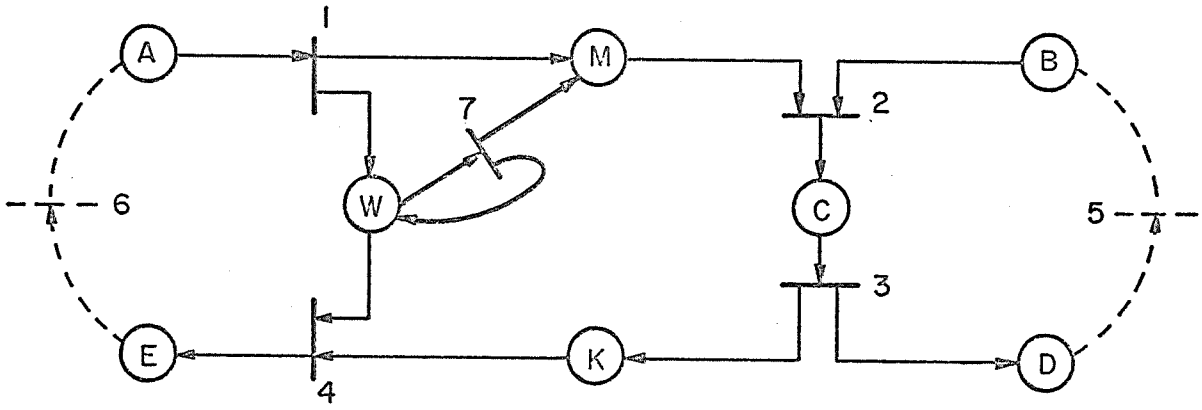


Figure 7.3: Recoverable TPN for the TM of figure 7.1

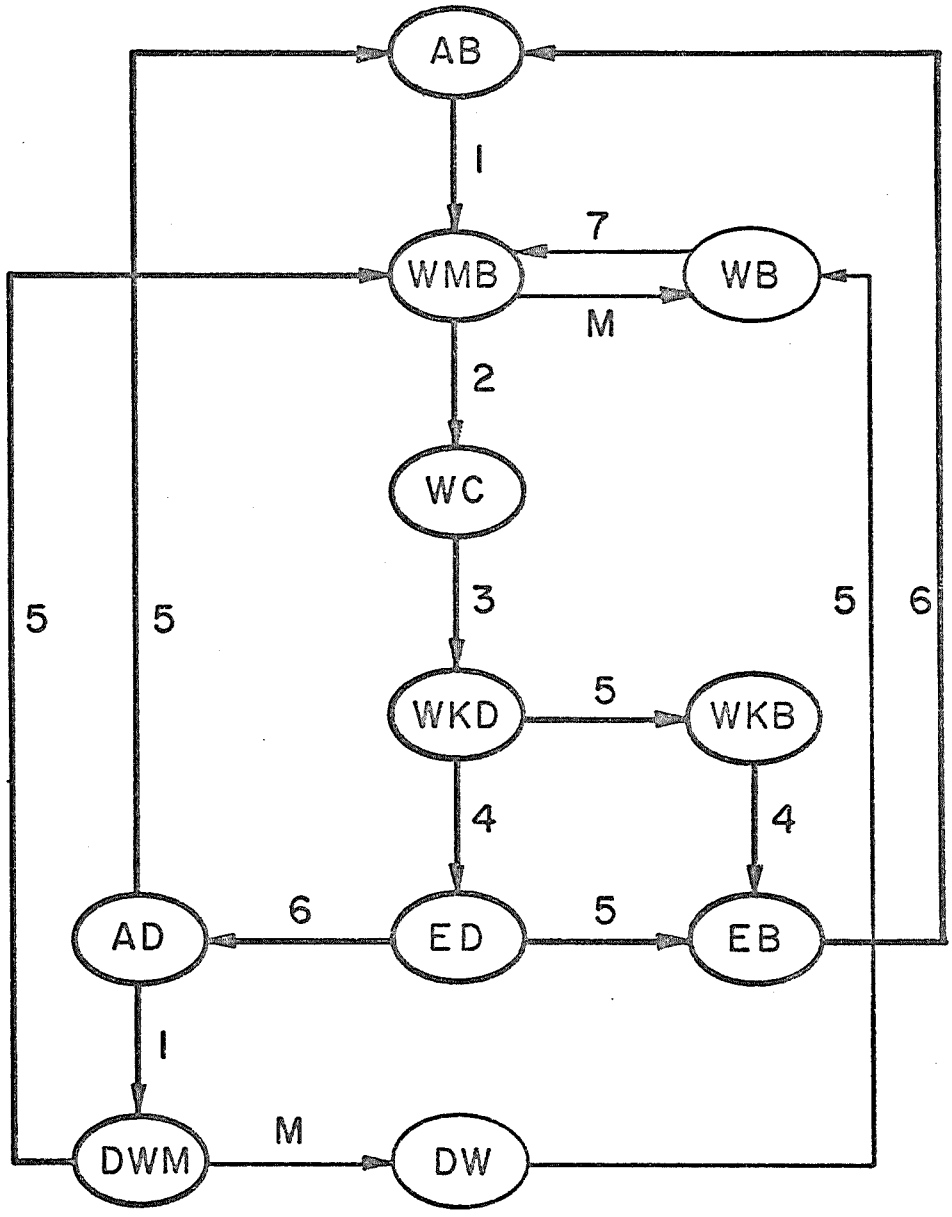


Figure 7.4: ETM for the TPN of figure 7.3

These correspond to the conditions A, B, K, W, E, C, and D in the PN of the previous example.

For simplicity, in the present example, we assume that n=2 (the same approach is applicable for any n). Figure 7.5 shows the PN for this case. This PN is similar to two instances of the PN shown in figure 7.1. The only difference is the dotted lines which represent the sender and receiver processes. In the present case, these processes are responsible for the correct sequencing of the messages.

Figure 7.6 shows the corresponding ETM for the case that a failure can occur in M1 or M2, assuming initial state A1B1.

This ETM is similar to two instances of the ETM shown in figure 7.2. In order to convert the PN of figure 7.5 to a recoverable PN, we use an approach similar to that described in the previous example. In this case, two bars are added, bars 17 and 27. In the same way as in the previous example, there exist:

1. IC(17) = W1
2. OC(17) = W1M1
3. $t^{*}17 > T^{**}_{W1}$ (7.2.1)

   and if:

   1. $T^{*}_{14}(W1K1D1) < T^{**}_{15}(W1K1D1)$
   2. $T^{*}_{16}(E1D1) < T^{**}_{15}(E1D1)$
   3. $T^{*}_{21}(A2D1) < T^{**}_{15}(A2D1)$

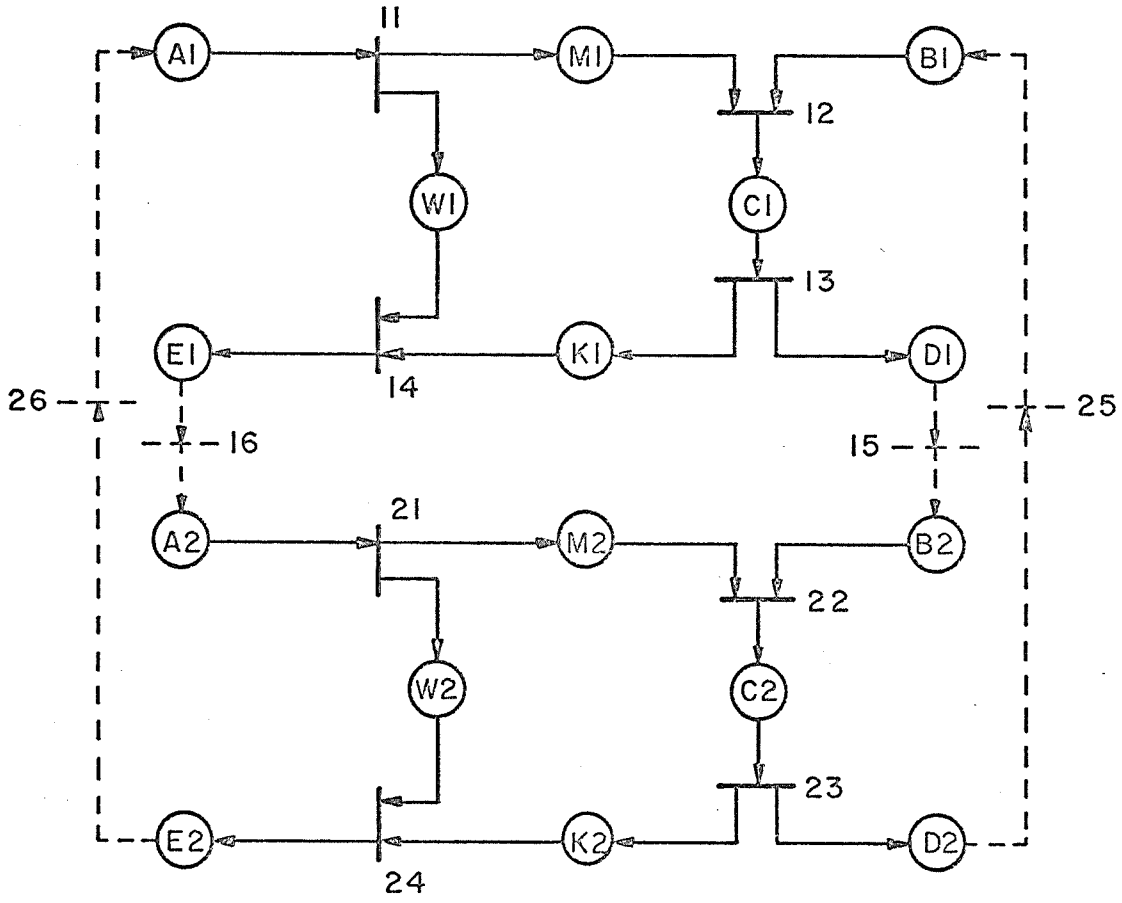


Figure 7.5: PN of a protocol process

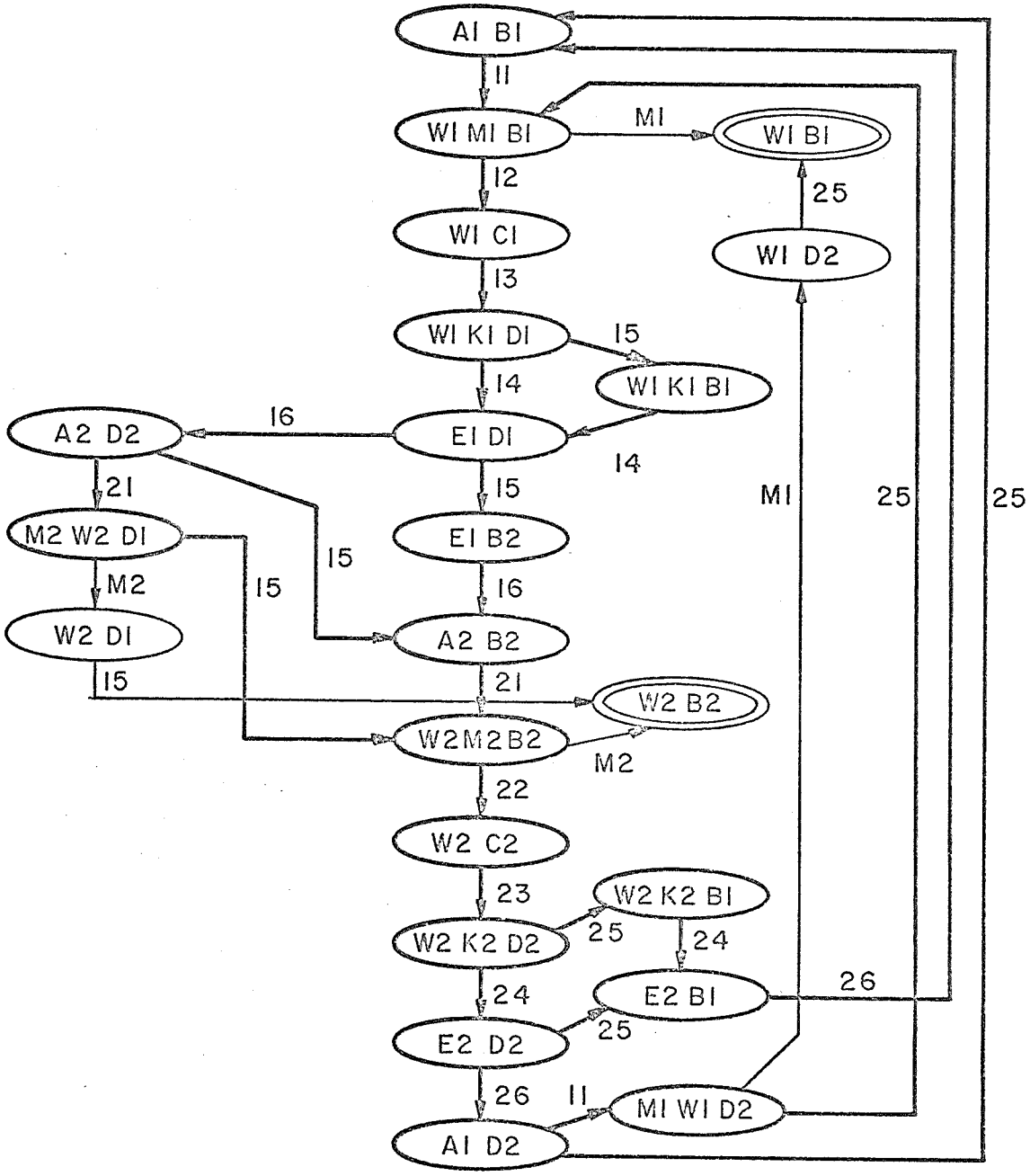


Figure 7.6: ETM for the PN of figure 7.5

then:

$T^{**}_{W1} = t^{**}15-(t^{*}14+t^{*}16+t^{*}21)+t^{**}12+t^{**}13+t^{**}14$

but if one of the conditions above is not satisfied then:

$T^{**}_{W1} = t^{**}12 + t^{**}13 + t^{**}14$

4. IC(27) = W2
5. OC(27) = W2M2
6. $t^{*}27 > T^{**}_{W2}$ (7.2.2)

and if:

1. $T^{*}_{24}(W2K2D2) < T^{**}_{25}(W2K2D2)$
2. $T^{*}_{26}(E2D2) < T^{**}_{25}(E2D2)$
3. $T^{*}_{11}(A1D2) < T^{**}_{25}(A1D2)$

then:

$T^{**}_{W2} = t^{**}25-(t^{*}24+t^{*}26+t^{*}11)+t^{**}22+t^{**}23+t^{**}24$

but if one of the conditions above is not satisfied then:

$T^{**}_{W2} = t^{**}22 + t^{**}23 + t^{**}24$

This TPN is shown in figure 7.7 and it is recoverable.

But, what happens if (7.2.1) or (7.2.2) are not satisfied?. In this case, bar 17 or bar 27 can fire before it is certain that the TPN is in an illegal state. Then the bars 17 or 27 might also fire in legal states. In order to simplify the following explanations for the case that (7.2.1) or (7.2.2) are not satisfied, we assume that:

1. $T^{*}_{16}(E1D1) > T^{**}_{15}(E1D1)$

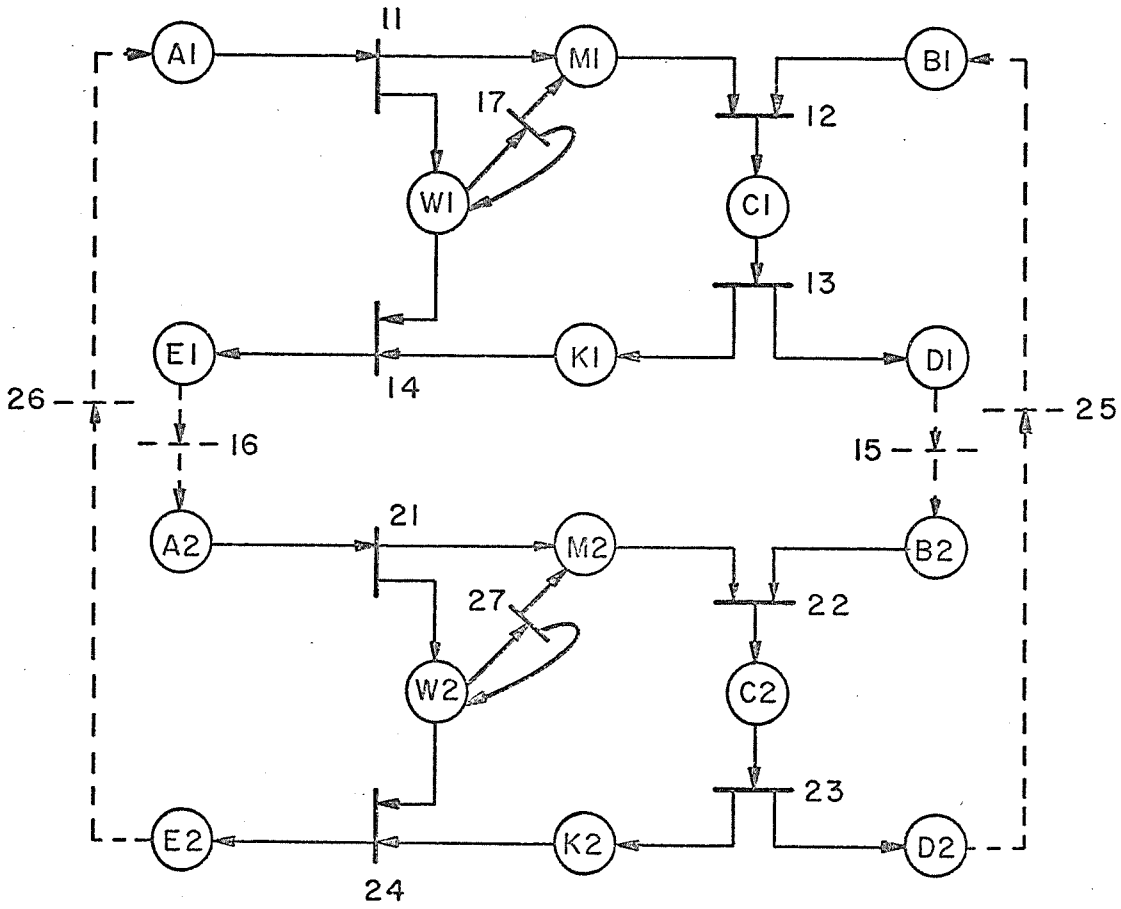


Figure 7.7: Recoverable TPN for the PN of figure 7.5

this means that bar 16 can not fire in E1D1 and, in the TM, the branch that includes the states A2D1 and M2W2D1 is canceled (see figure 7.6). In this case:

$$T^{**}_{W1} = t^{**}12 + t^{**}13 + t^{**}14$$

2. $T^{*}_{26}(E2D2) > T^{**}_{25}(E2D2)$

in the same way as in the previous case, bar 26 can not fire in state E2D2 and, in the TM, the branch that include the states A1D2 and M1W1D2 is canceled (see figure 7.6). In this case:

$$T^{**}_{W2} = t^{**}22 + t^{**}23 + t^{**}24$$

(Note that the two previous assumptions imply that the receiver is ready to receive before the sender is ready to send. In other words, bar 15 fires before 16 and 25 before 26.)

3. t*17 <u>does not</u> satisfy (7.2.1), but it is <u>big enough</u> to prevent the firing of bar 17 in legal states W1M1B1 and W1C1. In other words, bar 17 can fire only in illegal states or in legal states W1K1D1 and W1K1B2. In this case:

$$t^{**}12+t^{**}13+t^{**}14 > t^{*}17 > t^{**}12+t^{**}13 \qquad (7.2.3)$$

4. t*27 <u>does not</u> satisfy (7.2.2), but it is <u>big enough</u> to prevent the firing of bar 27 in legal states W2M2B2 and W2C2. In other words, bar 27 can fire only in illegal states or in legal states W2K2D2 and W2K2B1.

This means that:

$$t^{**}22+t^{**}23+t^{**}24 > t^{*}27 > t^{**}22+t^{**}23 \qquad (7.2.4)$$

These assumptions simplify the presentation of the example, but they do not reduce its generality. The same approach is applicable when the assumptions are not made, and only the results will be different, but not in principle.

The TM (not the ETM) for the TPN described in figure 7.7, for the case that the assumptions above are satisfied, is shown in figure 7.8. This TM is infinite since the number of instances of M1 and M2 grows infinitely. In this situation it can occur that the execution never returns to "normal execution". By "normal execution" we mean the legal states of figure 7.6. At this point, we can look at the problem in the following way:

> "when bar 17 fires in states W1K1D1 or W1K1B1, or when bar 27 fires in states W2K2D2 or W2K2B2, they introduce a pseudo failure of type generation of extra token".

When bar 17 fires, an extra token is added to M1, and when bar 27 fires, an extra token is added to M2. The states after the occurrence of the "pseudo failure" are called pseudo illegal states. The transitions between these states are called pseudo illegal transitions.

At this point, we want to insure that, after the

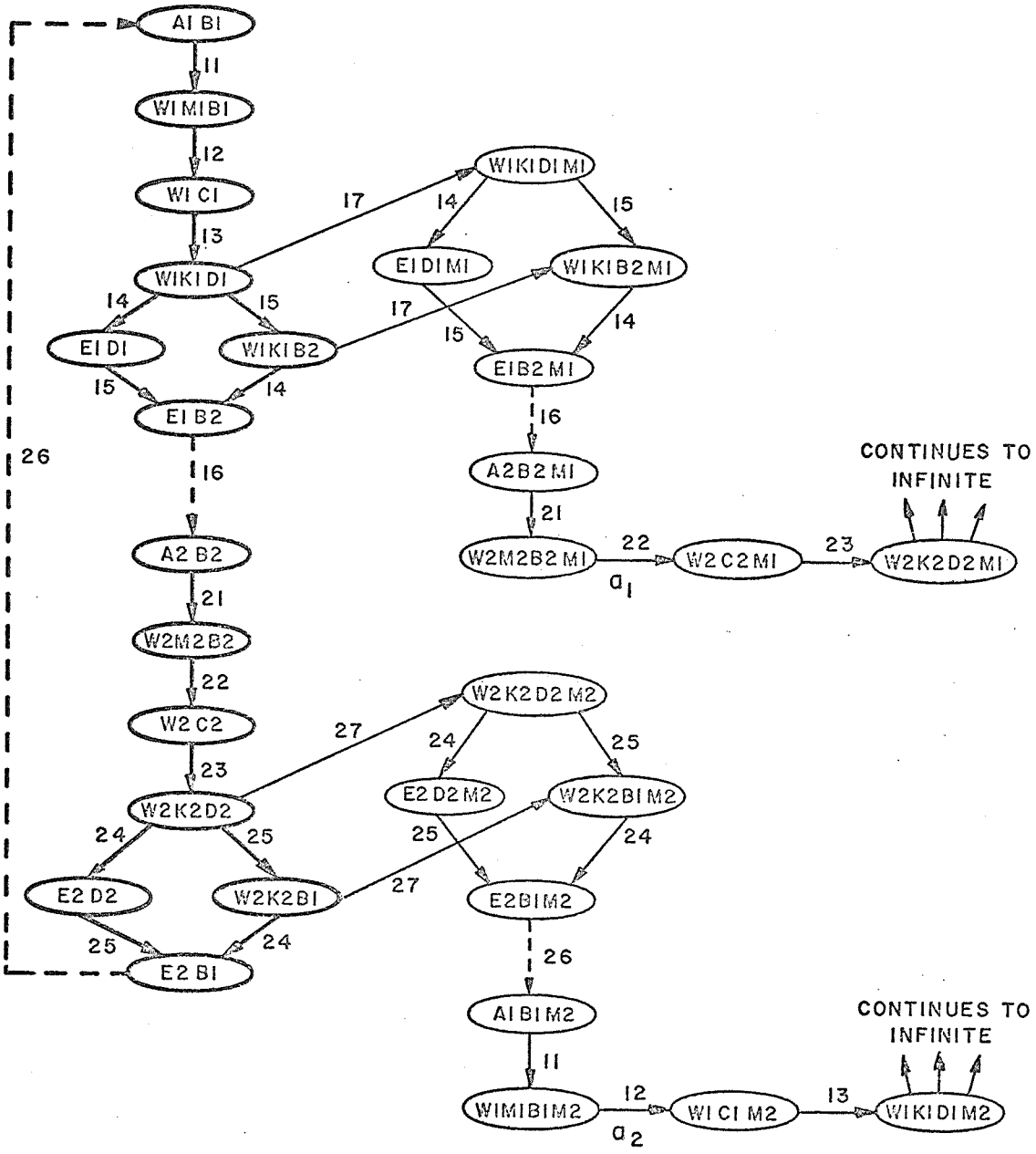


Figure 7.8: TM for the case that (7.2.3) and (7.2.4) are satisfied - pseudo-failures (tokens added before time)

occurrence of a pseudo failure, execution will always return to the legal states. The solution of this problem is the same as in the case that a real failure of type "generation of an illegal token" has occurred.

In order to solve in general this kind of problem, it is necessary to exhaustively analyze the problem of "recoverability under the generation of an illegal token," in a way similar to that done in chapter 4 for the case of "loss of token." But, several particular cases can be easily solved without such an extensive analysis.

Now the solution of our example will be given. At this point, we can not formally determine if our solution is the only possible solution. But, the solution presented here appears to be applicable in many practical cases.

Suppose that a "cut-set" of pseudo illegal arcs is chosen in the TM of figure 7.8. Since, in this case, the cut-set includes only pseudo illegal states, it divides the TM into two parts:

1. part 1 includes all the legal states and some (or none) of the pseudo illegal states,
2. part 2 includes only the pseudo illegal states not included in part 1.

In our example the cut-set of arcs [a1 , a2] as shown in figure 7.8 is chosen.

If bars are added to the PN such that:

1. there exists a path from each pseudo illegal state in

part 1, to a legal state,

2. the additional bars can not fire in legal states,
3. the arcs of the cut-set (a1 and a2 in the example) will never be executed,

then the process is recoverable under the occurrence of a pseudo failure. If the conditions above are satisfied after the occurrence of a pseudo failure, the execution will always return to a legal state.

In order to satisfy these conditions, the bars 18 and 28 are added to the TPN of figure 7.7, such that:

IC(18) = B1M2

OC(18) = B1

IC(28) = B2M1

OC(28) = B2

The new TPN is shown in figure 7.9 and the correspondent TM in figure 7.10. Figure 7.10 shows that conditions 1 and 2 are satisfied. Condition 2 is satisfied because neither IC(18) nor IC(28) (B1M2 or B2M1) are included in any of the legal states. In order to satisfy condition 3, we have to insure that arcs a1 and a2 (figure 18) will never be executed. This means that in state W2M2B2M1 bar 28 will fire before bar 22 can fire, and that in state W1M1B1M2 bar 18 will fire before bar 12 can fire. In other words, using property 6.2.5:

$$T^{**}_{18}(W1M1B1M2) < T^{*}_{12}(W1M1B1M2) \qquad (7.2.5)$$

$$T^{**}_{28}(W2M2B2M1) < T^{*}_{22}(W2M2B2M1) \qquad (7.2.6)$$

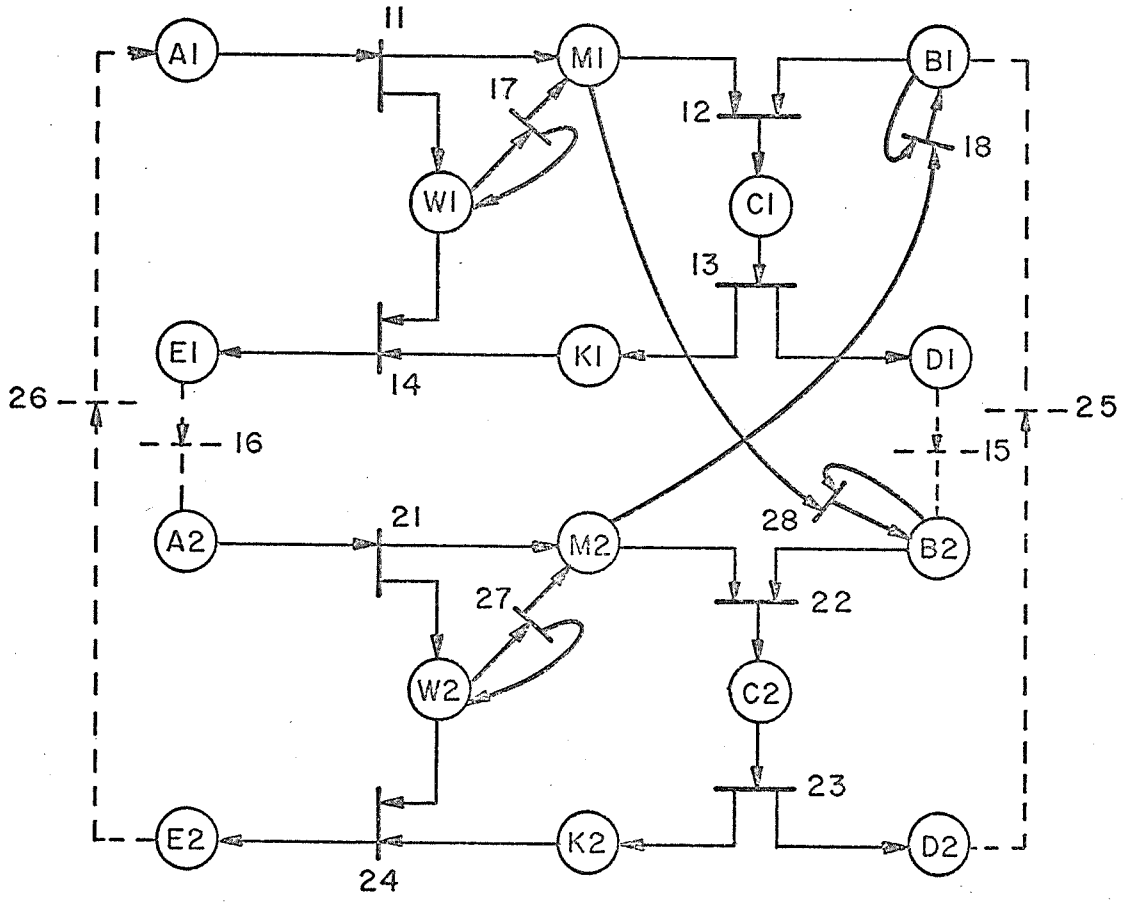


Figure 7.9: A new TPN

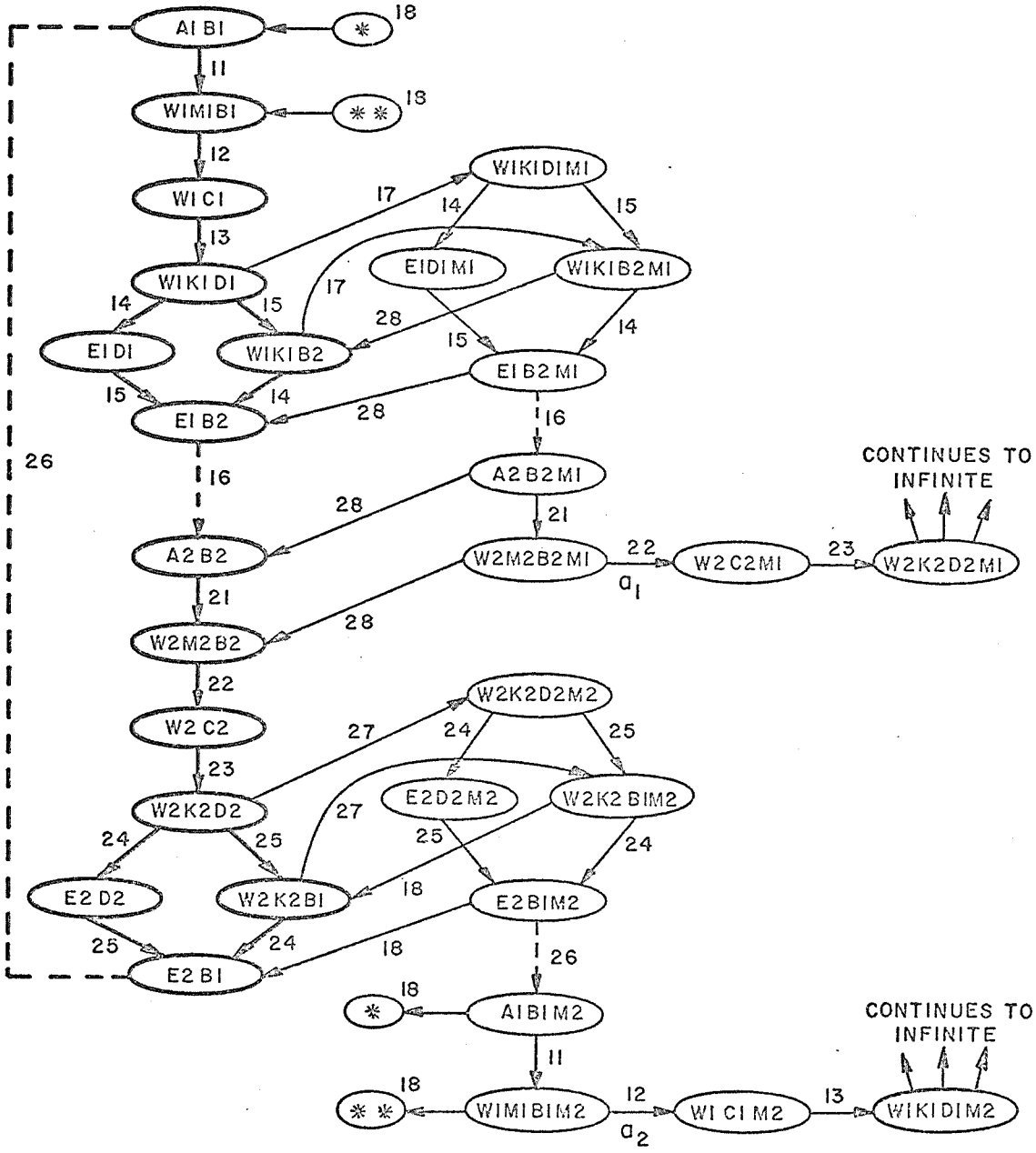


Figure 7.10: TM for the TPN of figure 7.9

But since M1 < IC(12), and the token in M1 is placed when the process enters the state W1M1B1M2 then:

$$T^*_{12}(W1M1B1M2) = t^*12 \qquad (7.2.7)$$

Note that in this case t*12 is the minimal time that can elapse between a token being placed on M1 until this token is removed. This time can be interpreted as the minimal propagation time of the message M1 !!

In the same way there exists:

$$T^*_{22}(W2M2B2M1) = t^*22 \qquad (7.2.8)$$

and t*22 can be interpreted as the minimal propagation time of the message M2.

From figures 7.9 and 7.10 it is possible to show that:

$$T^{**}_{18}(W1M1B1M2) = t^{**}18 - t^*11 - t^*26 \qquad (7.2.9)$$

and $$T^{**}_{28}(W2M2B2M1) = t^{**}28 - t^*21 - t^*16 \qquad (7.2.10)$$

Replacing (7.2.7) and (7.2.9) in (7.2.5) the result is:

$$t^{**}18 - t^*11 - t^*26 < t^*12 \qquad (7.2.11)$$

and replacing (7.2.8) and (7.2.10) in (7.2.6) the result is:

$$t^{**}28 - t^*21 - t^*16 < t^*22 \qquad (7.2.12)$$

The TM of the TPN of figure 7.9, with the constraints given by (7.2.3), (7.2.4), (7.2.11), and (7.2.12) is shown in figure 7.11. This TM includes all (and only) the states, legal and pseudo illegal, that are included in what we called "part 1" of the TM of figure 7.8. But, this is not

the only way to look at the problem. The pseudo illegal states of part 1 are allowed to hold tokens, just as the legal states. This means that these pseudo illegal states might just as well be considered legal states. Thus, all the states of figure 7.11 can be considered legal. These two ways of interpretation are equally convenient.

The TPN of figure 7.9, with the constraints (7.2.3), (7.2.4), (7.2.11) and (7.2.12) was designed so that it is recoverable under failures of kind "loss of tokens" in M1 or M2. The ETM of figure 7.12 shows this property.

The process, as given by the TM of figure 7.11 or the TPN of figure 7.9 (and the constraints in the execution times), has the following interesting properties:

1. The messages are received in the same order that they are sent. This property is shown directly from figure 7.11. States W2M2B2M1 and W1M1B1M2 are the only states in which two messages are simultaneously in the link. But, in W2M2B2M1 the message M1 was sent first (note that the only predecessor of W2M2B2M1 is A2B2M1), and in this case M1 is received first (the only successor of W2M2B2M1 is W2M2B2). In the same way, when the process is in W1M1B1M2 the message M2 was sent first and it will be received first. This limitation on the order of the messages can be removed, in part, if the sequence number of each message is chosen from more than two

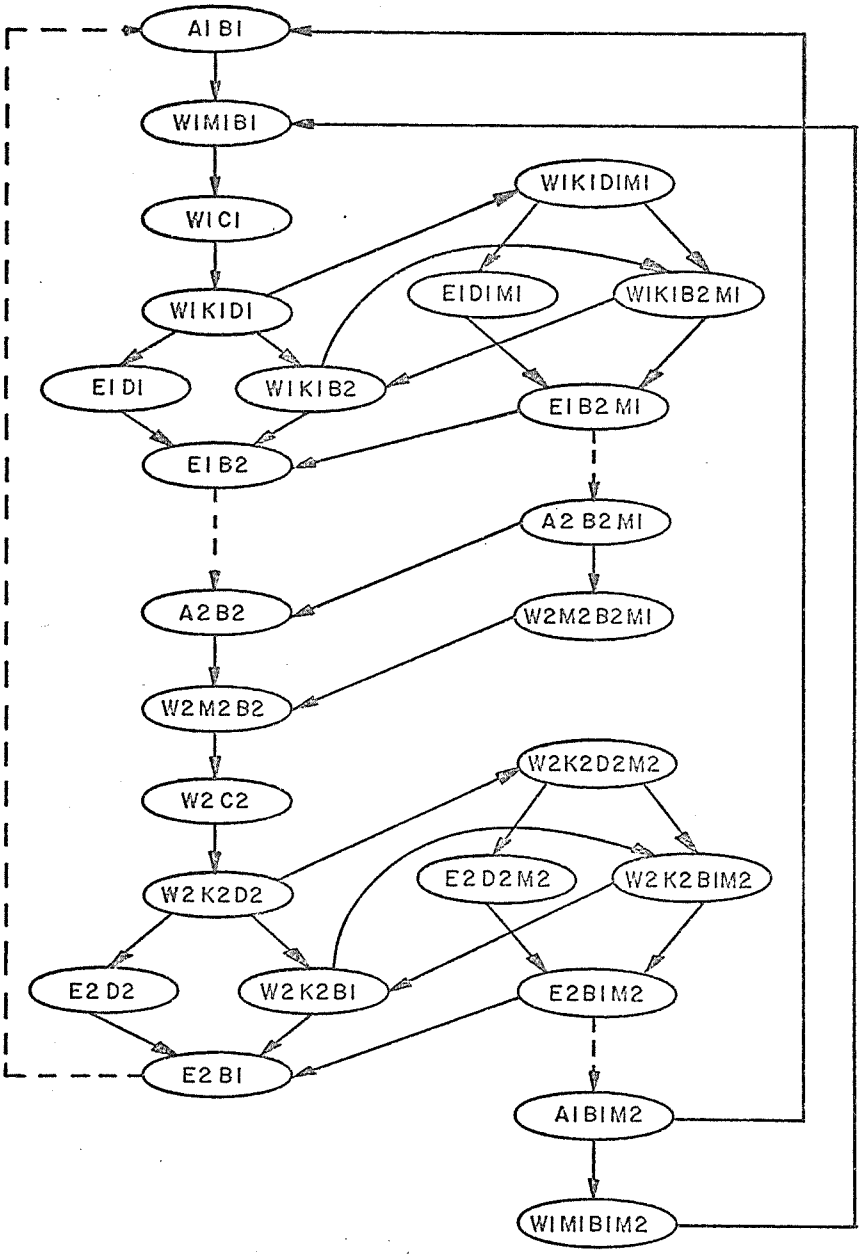


Figure 7.11: TM for the TPN of figure 7.9

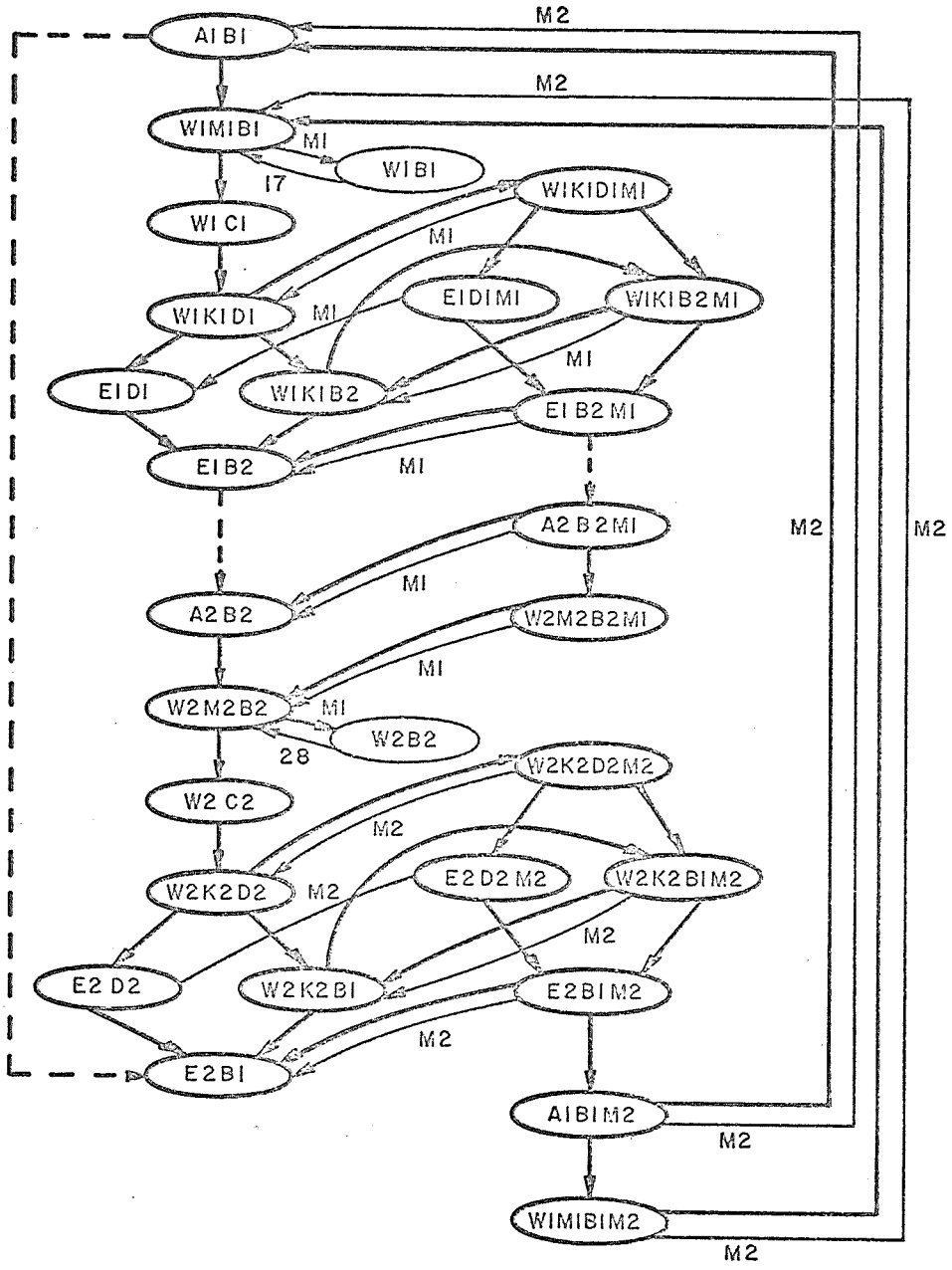


Figure 7.12: ETM for the TPN of figure 7.9

possibilities [POST74].

2. Inequalities (7.2.11) and (7.2.12) can be rewritten as:

$$t^{**}18 < t^{*}12 + t^{*}11 + t^{*}26 \qquad (7.2.11a)$$

$$t^{**}28 < t^{*}22 + t^{*}21 + t^{*}16 \qquad (7.2.12a)$$

(7.2.11a) shows that the maximal time that it takes to receive an illegal message (t**18) has to be smaller than the minimal time it takes to prepare a new message (t*26), to send it (t*11) and to receive it (t*12), (see figure 7.9). The same relation exists in (7.2.12a).

3. As shown before, t*12 represents the minimal propagation time of the message M1. In a certain way, t**18 represents the maximal propagation time of M2. But since in practice M1 and M2 propagate in the same channel then (t**18 - t*12) denotes the variance in the propagation time of the messages. But, from (7.2.11a):

$$t^{**}18 - t^{*}12 < t^{*}11 + t^{*}26 \qquad (7.2.11b)$$

Thus, the minimal preparation time of a message (t*26) plus the minimal sending time (t*11) has to be greater than the variance of the propagation time. This means that in a recoverable process of this kind a higher uncertainty in the propagation time leads to the reduction of the frequency of the

messages. The same conclusion can be derived from (7.2.12a).

Chapter 8

CONCLUSIONS

8.1 Summary

In this dissertation, the problem of recoverability of processes has been modeled and formally defined using Petri-nets. The particular case of failures of type "loss of tokens" has been exhaustively explored. A way of designing processes that are recoverable from this kind of failure was given. This method of design is based on the properties of recoverable TMs and on a procedure for designing a PN that implements a given TM. This last procedure can be useful not only for the design of recoverable processes, but in general for designing PN's with properties which are, sometimes, better reflected in the TM than in the PN itself.

In the case that no assuptions have been made about the execution times of the different parts of the PN, the recoverable processes under a failure of type "loss of token" are very limited in their possible structure. These limitations are usually unacceptable in practical (real) processes. Because of these limitations, some knowledge

about the execution times was introduced in the PN's, and a new model, the TPN, was defined. For any given TM, that has a correspondent PN with an ETM in which there are no loops of illegal states, a TPN can be designed so that it executes the given TM and is recoverable from a given failure of type "loss of token."

But, in this recoverable TPN it is necessary to accept constraints on the execution times of its parts. If these constraints can not be accepted, they can be partially relaxed by introducing a "pseudo failure" of type "generation of token." In this case, the recovery from the "pseudo failure" has to be insured.

The approach used in this work for the study of failures of type "loss of tokens" can be applied in order to explore other types of failures.

Other authors ([LARS74]) have written about the importance of "the problem of including some measure of service times at the modules." Since the TPN includes this measure of service time, this model can be useful not only in the exploration of recoverability, but in order to model and explore other properties of processes.

The approach presented in this work does not differentiate between the hardware components and the software parts of the processes. The approach is uniform and in practice each part can be implemented by any kind of elements.

8.2 Limitations of the Described Methodology

This methodology of recoverable system design has the following limitations:

1. TM is not always the best design tool. For some relatively simple PN the corresponding TM is, some times, very large.
2. I have not developed a way of evaluating and comparing the possible results.
3. The methodology presented in this dissertation carries all the limitations of Petri nets. Among these limitations are the difficulty of representing data and the difficulty of modeling large systems.

8.3 Suggestions for Further Exploration

This work points out several areas needing further research. Among these areas are:

1. The formal analysis of recoverability under the occurrence of other kinds of failures. Among these, "generation of illegal tokens," etc.
2. Further general research on the TPN model.
3. The formal analysis of other properties of processes,such as:
    (a) "fail-soft,"
    (b) "fault-tolerant,"
    (c) "best-effort"
4. Research on the propagation of failures among

processes in an hierarchical structure.

REFERENCES

ARMS66 Armstrong D.B., "On Finding a Nearly Minimal Set of Fault Detection Tests for Combinational Logic Nets"; IEEE Transactions on Electronic Computers, vol. EC.15, No. 1; February, 1966.

BAER68 Baer J.l., Graph Models of Computation in Computer Systems, Ph.D Dissertation, Report No. 68.46. Department of Engineering, University of California, Los Angeles, 1968 (UCLA-10P14-51).

BOVE68 Bovet D, Memory Allocation in Computer Systems, Ph.D. Dissertation, Report No. 68-17, School of Engineering and Applied Science, University of California, Los Angeles, 1968.

CERF71 Cerf V., E.B. Fernandez, K.P. Gostelow, and S.A. Volansky, "formal Control-Flow Properties of a Graph Model of Computation", UCLA Technical Report, No. ENG-7178, University of California, Los Angeles, December 1971 (UCLA-10P14-105).

CERF72 Cerf V.G. Multiprocessors, Semaphores, and a Graph Model of Computation, Ph.D. Dissertation, ENG-7223, Computer Science Department, University of California, Los Angeles, April 1972 (UCLA-10P14-110).

DENN71 Dennis J.B. and E.C Van Horn, "Programming Semantic of Multiprogrammed Computations", Communications of the ACM 6-3, March 1966.

DENN71 Denning P.G., "Third Generation Computer Systems", Computer Surveys, Vol 3, No. 4, December 1971.

EICH65 Eichelberger E.B., "Hazard Detection in Combinational and Sequential Switching Circuits", IBM Journal, March 1965.

ESTR63 Estrin G. and R. Turn, "Automatic Assignment of Computations in a Variable Structure Computer System", IEEE Transactions on Computers, EC-12, December 1963.

FALK74 Falk H., "data Communications", IEEE Spectrum, January 1974.

FARB72 Farber D.J, and F.R. Heinrich, "The Structure of a Distributed Computer System - The Distributed File System", Proc. International Conference on Computer Communications, October 1972

FARB73 Farber D.J, J. Feldman, F.R. Heinrich, M.D. Hopwood, K.C. Larson, D.C. Loomis, and L.A. Rowe, "The Distributed Computing System", Proc. Seventh Annual IEEE Computer Society International Conference, February 1973

FERN72 Fernandez E.B., Activity Transformation on Graph Models of Parallel Computations, Ph.D. Dissertation, ENG-7287, Computer Science Department, University of California, Los Angeles, October 1972 (UCLA-10P-14-116).

GOST71 Gostelow K.P., Flow of Control, Resource Allocation, and the Proper Termination of Programs, Ph.D. Dissertation, ENG-7179, Computer Science Department, University of California, Los Angeles, December 1971 (UCLA-10P-14-106).

GOST74 Gostelow K.P. and T.J. van Weert, "Processes and Networks", Stichting Academisch Rekencentrum Amsterdam, Postbus 7161, Amsterdam, The Netherlands, January 14, 1974. (Author's present address: University of California, Irvine 92664)

HOLT68 Holt A.W., H. Saint, R.M. Shapiro and S. Warshall, "Final Report for the Information System Theory Project", Rome Air Development Center (Appied Data Research Inc.) Contract #AF30(602)-4211, 450 Seventh Ave., New York, New York 10001, 1968.

HOLT69 Holt A.W. and F. Commoner, "Events and Conditions", Applied Data Research Inc., 450 Seventh Ave., New York, New york 10001, 1969.

HORN66 Van Horn E.C., "Computer Design of Asynchronously Reproducible Multiprocessing", Project MAC, Report MAC-TR-39, MIT, 1966.

KARP69 Karp R.M. and R.E. Miller, "Parallel Program Schemata", Journal of Computer and System Science, 3(4), May 1969.

KNUT69 Knuth D.E., The Art of Computer Programming, Vol. 2, Addison-Wesley, Melo Park, California, 1969.

KOHA71 Kohavi Z. and D.A. Spires, "Designing Sets of Fault-Detection Tests for Combinational Logic Circuits", IEEE Transactions on Computers , Vol. C-20, No. 12, December 1971.

KOHA72 Kohavi I. and Z. Kohavi, "Detection of Multiple Faults in Combinational Logic Networks", IEEE Transactions on Computers, Vol. C-21, No. 6, June 1972.

LARS74 Larson K.C. Computation Graphs, Ph.D. Dissertation, Department of Information and Computer Science, University of California, Irvine, (in preparation).

MART66 Martin D.F., The Automatic Assignment and Sequencing of Computations on Parallel Processor Systems, Ph.D Dissertation, Report No.66-4, Department of Engineering, University of California, Los Angeles, 1966.

METC73 Metcalfe R.M., Packet Communication, Project MAC, Report MAC-TR-114, MIT, December 1973.

PATI70 Patil S.S. Coordination of Asynchronous events, Sc.D. Thesis, Project MAC, Report MAC-TR-72, MIT, 1970.

PETR62 Petri C.A. Communication with Automata, Thesis, Dermstadt Institute of Technology, Bonn, Germany, 1962.

POST74 Postel J.C., A Graph Model Analysis of Computer Communications Protocols, Ph.D. Dissertation, Computer Science Department, University of California, Los Angeles, 1974.

RODR69 Rodriguez J. A Graph Model for Parallel Computation, Sc.D. Thesis, Department of Electrical Engineering, MIT, September 1967. (Also MAC-TR-56, September 1969).

ROTH66 Roth J.P., "Diagnosis of Automata Failures: A Calculus and a Method", IBM Journal, July 1966.

ROTH67 Roth J.P., W.G. Bouricius and P.R. Schneider, "Programmed Algorithms to Compute Tests to Detect and Distinguish Between Failures in Logic Circuits", IEEE Transactions on Electronic Computers, Vol. EC-16, No. 5, October 1967.

ROWE73 Rowe L.A., Hopwood M.D. and Farber D.J., "Software Methods for Achieving Fail-Soft Behavior in the Distributed Computing System", Record 1973 IEEE Symposium on Computer Software Reliability, New York City, May 1973.

RULI71 Rulifson J.F., "QA4 Programming Concepts", Artifitial Intelligence Group TN-60, Stanford Research Institute, Menlo Park, California, August 1971.

RUSS69 Russell E.C. Automatic Program Analysis, Ph.D. Dissertation, Report No. 69-12, School of Engineering and Applied Science, University of California, Los Angeles, March 1969 (UCLA-10P14-72)

SALT66 Saltzer J.H. "Trafic Control in a Multiplexed Computer System", Project MAC, Report MAC-tr-30, 1966.

SELL68 Sellers F.F. Jr., M.Y. Hsiao and L.W. Bearnson, "Analyzing Errors With the Boolean Difference", IEEE Transactions on Computers, Vol. C-17, No.7, July 1968.

SLUT68 Slutz D.R. The Flow-Graph Schemata Model of Parallel Computation, Ph.D. Dissertation, MAC-TR-53, MIT, Sept. 1968.

VOLA70 Volansky S.A., Graph Model Analysis and Implementation of Computational Sequences, Ph.D. Dissertation, 70-48, Computer Science Department, University of California, Los Angeles, 1970
(UCLA-10P14-93)

YAV71 Yav S.S. and Y.S. Tang, "An Efficient Algorithm for Generating Complete Test Sets For Combinational Logic Circuits", IEEE Transactions on Computers, Vol. C-20, No. 11, November 1971.

YOEL64 Yoeli M. and S. Rinon, "Application of Ternary Algebra to the Study of Static Hazards", Journal of the Association for Computer Machinery, Vol. 11, No. 1, January 1964.

Appendix A

THEOREMS OF CHAPTER 3

In this appendix, the statements of chapter 3 are formally proved. The five first theorems show properties of the TM and especially its relation to the corresponding PN. Theorem A.6 and A.7 gives necessary and sufficient conditions such that, given the states of a TM and a group of all the TM transitions with the same "minimal transition," this group can be implemented by a PN. Note that a TM can be effected by a PN if and only if each of its groups with common "minimal transition" can be implemented.

A.1 Theorem

If there exists a TM transition tr = Si -> Sj

such that:

MIN(tr) = a -> b

and a bar bf such that:

1. IC(bf) = a + x (x is a bag)
2. OC(bf) = b + x and:
3. $x \leq RE(tr)$

then, the TM transition tr is effected by bf in PN.

Proof:

By definition 3.2.9:

$$Si = a + RE(tr) \qquad (A.1.1)$$

$$Sj = b + RE(tr) \qquad (A.1.2)$$

Since it is given that $x \leq RE(tr)$, from (A.1.1)

$$a + x \leq Si.$$

and: $IC(bf) \leq Si$

By definition of the firing algorithm (3.2.14), in state Si the bar bf can fire and the new state is:

$$Sk = Si - IC(bf) + OC(bf)$$

and after replacement of IC(bf) and OC(bf) from the conditions of the theorem, and Si from (A.1.1):

$$Sk = [a+RE(tr)]-[a+x]+[b+x] = b+RE(tr) \qquad (A.1.3)$$

Comparing (A.1.2) and (A.1.3):

$$Sj = Sk$$

and thus the transition tr in TM is effected by the firing of bar bf in PN. Q.E.D

A.2 Theorem

If there is a TM transition t1 = Si -> Sj

such that:

$$MIN(t1) = a \rightarrow b,$$

then, there is not another TM transition t2 such that:

$$MIN(t1) = MIN(t2)$$

and

$$LHS(t2) = Si$$

In other words, if there is a TM transition exiting from a legal state, there is not another transition with the same "minimal transition" exiting from the same state.

Proof:

If transition t1 = Si -> Sj

exist in the TM, and:

MIN(t1) = a -> b

then by definition (3.2.9):

RE(t1) = Si - a

and: Sj = b + RE(t1) = Si + b - a (A.2.1)

Suppose that there exist:

t2 = Si -> Sk

and: MIN(t2) = a -> b

then, by definition (3.2.9):

RE(t2) = Si - a

and: Sk = b + RE(t2) = Si + b - a (A.2.2)

Comparing (A.2.1) and (A.2.2):

Sj = Sk

thus: t1 = t2

Since the collection T of all the transitions in the TM (see 3.2.15) is a set (no multiple instances of the same element),and t1 is equal to t2, then t1 and t2 are the same transition.

Q.E.D.

A.3 Theorem

If the TM transition tk = Si -> Sj is effected by the firing of bar bf then:

(1) $IC(bf) \leq Si$

(2) $Sj = Si - IC(bf) + OC(bf)$

Proof:

(1) If tk is effected by bf, then the firing algorithm (3.2.14) must be applied to Si for the bar bf. I.e.

$$IC(bf) \leq Si$$

(2) After bf fires in state Si, the next state, Sj, has to be the new state produced by the firing algorithm:

$$Sj = Si - IC(bf) + OC(bf)$$

Q.E.D.

A.4 Theorem

If the transition tr = Si -> Sj

with: MIN(tr) = a -> b

is effected by the bar bf then:

1. $IC(bf) = a + x$ (x is a bag)
2. $OC(bf) = b + x$ and:
3. $x \leq RE(tr)$

Proof:

From theorem A.3:

$$Sj = Si - IC(bf) + OC(bf)$$

or:

$$Sj + IC(bf) = Si + OC(bf) \qquad (A.4.1)$$

If we denote LHS(MIN(tk)) = a and RHS(MIN(tk)) = b, then by definition (3.2.9):

$$Si = a + RE(tr) \qquad (A.4.2)$$

and: $$Sj = b + RE(tr) \qquad (A.4.3)$$

replacing (A.4.2) and (A.4.3) in (A.4.1) yields:

$$b + RE(tr) + IC(bf) = a + RE(tr) + OC(bf)$$

This means that:

$$b + IC(bf) = a + OC(bf) \qquad (A.4.4)$$

But, by definition 3.2.9, a and b have no common elements. So, all the elements of "a" have to be included in IC(bf), and all the elements of b must be included in OC(bf). I.E.,

$$a \leq IC(bf) \qquad (A.4.5)$$

and: $$b \leq OC(bf)$$

From (A.4.5) there exists a bag x such that:

$$IC(bf) = a + x \qquad (A.4.6)$$

and part (1) of the theorem is proved.

Replacing (A.4.6) in (A.4.4) yields:

$$b + a + x = a + OC(bf)$$

or: $$OC(bf) = b + x$$

and part (2) of the theorem is proved.

From theorem A.3(1):

$$IC(bf) \leq Si$$

Replacing (A.4.6) and (A.4.2) yields:

$$a + x \leq a + RE(tr)$$

or: $$x \leq RE(tr)$$

and part (3) of the theorem is proved.

Q.E.D

Note that theorems A.1 and A.4 point out a set of necessary and sufficient conditions that a bar must satisfy in order to carry out a given TM transition.

A.5 Theorem

If transitions:

$$t1 = Si_1 \rightarrow Sj_1$$

and: $$t2 = Si_2 \rightarrow Sj_2$$

are effected by the same bar (say bf),

then: $MIN(t1) = MIN(t2)$

Proof:

Suppose that:

$$MIN(t1) = a1 \rightarrow b1$$

and: $$MIN(t2) = a2 \rightarrow b2$$

If transition t1 is effected by the bar bf, then by theorem A.4:

$$IC(bf) = a1 + x1$$

and: $$OC(bf) = b1 + x1$$

In the same way, for transition t2:

$$IC(bf) = a1 + x1 = a2 + x2 \qquad (A.5.1)$$

and: $$OC(bf) = b1 + x1 = b2 + x2 \qquad (A.5.2)$$

By the definition of equality (see 3.2.5):

$$a1 + x1 \leq a2 + x2$$

and: $$b1 + x1 \leq b2 + x2$$

This means that:

$$x1 \leq a2 + x2$$

and: $$x1 \leq b2 + x2$$

But, by the definition of MIN(t2) (definition 3.2.9), a2 and b2 have no common elements. Therefore, no elements of x1 are included in both a2 and b2. Thus, x1 is included in x2, or in other words:

$$x1 \leq x2 \qquad (A.5.3)$$

But, from equations (A.5.1) and (A.5.2) in the same way:

$$x2 \leq x1 \qquad (A.5.4)$$

so that, as shown in 3.2.5, from (A.5.3) and (A.5.4) there exist:

$$x2 = x1$$

and from (A.5.1) and (A.5.2) then:

$$a1 = a2$$

and $$b1 = b2$$

Q.E.D

A.6 Theorem

Let: $$tk_p = Si_p \rightarrow Sj_p \quad ; \quad (p=1,2,....,n)$$

be a group of all the TM transitions having the same "minimal transition":

$$MIN(tk_p) = a \rightarrow b.$$

If: $$\forall p \; \forall k \;\; (k \neq i_p) \Longrightarrow Si_p \not\leq Sk \qquad (A.6.1)$$

then there exists at least one set [xq] that satisfies:

$$\forall p \; \exists q \;\; xq \leq RE(tk_p) \qquad (A.6.2)$$

and: $\forall q \; \forall k \;\; (k \neq i_p) \Longrightarrow a + xq \not\leq Sk$ (A.6.3)

Note that (A.6.2) is (3.3.9) and (A.6.3) is (3.3.10).

Proof:

Suppose that the elements of [xq] are chosen such that:

$$xq = xp = RE(tk_p) \qquad (A.6.4)$$

then:

(1) by definition, the elements of xq satisfy (A.6.2)

(2) but, by (A.6.4)

$$q = p$$

and:

$$a + xq = a + xp = a + RE(tk_p) = Si_p$$

and replacing Si_p in (A.6.1) then the result is (A.6.3)

Q.E.D

A.7 Theorem

Let: $tk_p = Si_p \rightarrow Sj_p \quad ; \quad (p=1,2,....,n)$

be a group of all the TM transitions having the same "minimal transition":

$$MIN(tk_p) = a \rightarrow b$$

If there exist at least one k , $(k \neq i_p)$, such that for some i_p:

$$Si_p \leq Sk$$

then, there does not exist a set [xq] such that the conditions:

$$\forall p \; \exists q \;\; xq \leq RE(tk_p) \qquad (A.7.1)$$

and $\forall q \; \forall k \;\; (k \neq i_p) \Longrightarrow a + xq \not\leq Sk$ (A.7.2)

are satisfied.

Note that (A.7.1) is (3.3.9) and (A.7.2) is (3.3.10).

Proof:

Suppose that exist an element of k, say k1, such that:

$$Si_{p1} \leq Sk_1 \qquad (A.7.3)$$

Suppose that there exist a set [xq] such that (A.7.1) and (A.7.2) are satisfied. Applying (A.7.2) for k_1 and for p1 the result is:

$$\forall q \quad (k_1 \neq i_{p1}) \Longrightarrow a + xq \not\leq Sk_1$$

But, since $k_1 \neq i_{p1}$ is "True", then it is also true that

$$\forall q \quad a + xq \not\leq Sk_1$$

If, for all q, Sk_1 does not include $a + xq$, then since Si_{p1} is included in Sk_1 (A.7.3):

$$\forall q \quad a + xq \not\leq Si_{p1}$$

Therefore:

$$\forall q \quad xq \not\leq Si_{p1} - a \qquad (A.7.4)$$

Since by definition 3.2.9:

$$Si_{p1} - a = RE(tk_{p1}) \qquad (A.7.5)$$

Replacing (A.7.5) in (A.7.4):

$$\forall q \quad xq \not\leq RE(tk_{p1}) \qquad (A.7.6)$$

But (A.7.6) contradicts (A.7.1). Therefore, set [xq] does not exist.

Q.E.D.

Appendix B

THEOREMS OF CHAPTER 4

In this appendix, the statements of chapter 4 are proven formally. Theorem B.1 shows a property of the transitions of type t^1 (see 4.1.2) in TM. Theorems B.2, B.3, B.4, and B.5 prove that <u>only</u> bars that fire transitions of type t^3 (t^3 is defined in 4.1.2) in Ai + F can fire in Ai. Theorems B.6 and B.7 prove properties of the transitions of kind t^3 in the TM. Theorems B.8 and B.9 state necessary and sufficient conditions such that a given TM transition can be implemented <u>only</u> as of type t^3. Theorems B.10 and B.11 state necessary and sufficient conditions such that a given TM transition can be implemented <u>only</u> as of type t^2. This appendix uses the notation defined in 3.2 and in 4.1, and the properties of TM's proved in appendix A.

B.1 <u>Theorem</u>

If the transition of type t^1:

$$t^1_k = S^2_i \rightarrow S^1_j$$

is implemented by the bar bf then:

$$\#(F,S_i^2) = \#(F,IC(bf))$$

Proof:

There exists:

$$t_k^1 = S_i^2 \text{ -> } S_j^1$$

Since by definition (4.1.1) S_j^1 does not include the condition F, then by definition of MIN(t) (3.2.9) there exists:

$$\#(F,S_i^2) = \#(F,LHS(MIN(t_k^1))) \qquad (B.1.1)$$

If t_k^1 is implemented by bar bf, from theorem A.4(1) and definition 3.2.14:

$$LHS(MIN(t_k^1)) \leq IC(bf)) \leq S_i^2$$

then: $$\#(F,LHS(MIN(t_k^1))) \leq \#(F,IC(bf)) \leq \#(F,S_i^2)$$

(B.1.2)

Comparing (B.1.1) with (B.1.2), yields:

$$\#(F,S_i^2) = \#(F,IC(bf))$$

Q.E.D.

B.2 Theorem

Bars that, in state Ai + F are of type b^1, can not fire in state Ai.

Proof:

Suppose that b_f^1 is a bar of type b^1. Theorem B.1 shows that:

$$\#(F,Ai+F) = \#(F,IC(b_f^1))$$

This means that in Ai there are less instances of F than in

$IC(b_f^1)$. In other words:

$$IC(b_f^1) \leq Ai$$

and this contradict the firing algorithm in 3.2.14, so that b_f^1 can not fire in Ai and thus not for any b^1.

Q.E.D.

B.3 Theorem

Bars that, in state Ai + F are of type b^2, can not fire in state Ai.

Proof:

Suppose that b_f^2 is a bar of type b^2. Bars of type b^2 fire transitions of type t^2. Following the definition of transitions of type t^2 (section 4.1.2):

$$\#(F,Ai+F) = \#(F,IC(b_f^2))$$

and then the proof continues as in the previous theorem.

B.4 Theorem

Bars that, in state Ai + F are of type b^3, can fire in state Ai.

Proof:

Suppose that b_f^3 is a bar of type b^3. If b_f^3 can fire in Ai + F then:

$$IC(b_f^3) \leq Ai + F \qquad (B.4.1)$$

But by definition of transitions of type t^3 (4.1.2) the number of instances of F in Ai + F is greater than those

instances in $IC(b_f^3)$. This means that the number of instances of F in Ai is <u>equal or greater</u> than those instances in $IC(b_f^3)$. But, by (B.4.1), the instances of the other elements of Ai are also included in $IC(b_f^3)$, thus:

$$IC(b_f^3) \leq Ai$$

and the firing algorithm can be applied.

Q.E.D.

B.5 <u>Theorem</u>

Bars that, in state $Ai + F$ are of type b^4, can <u>not</u> fire in state Ai.

<u>Proof</u>:

Suppose that b_f^4 is a bar of type b^4. By definition, b_f^4 can not fire in $Ai + F$ (4.1.2 and 4.1.3). This means that the firing algorithm can not be applied. The firing algorithm can not be applied only if:

$$IC(b_f^4) \not\leq Ai + F$$

but: $$Ai \leq Ai + F$$

then: $$IC(b_f^4) \not\leq Ai$$

and the firing algorithm also can not be applied for b_f^4 in state Ai and thus not for any b^4.

Q.E.D.

Since b^3 has been shown to be the only type of bar that can fire in Ai, we will now consider some properties of transitions executed by bars b^3.

B.6 Theorem

If bar b_f^3, of type b^3, implements the transition:

$$t_1^3 = A1 + F \rightarrow A2 + F$$

then, b_f^3 also implements the transition:

$$t2 = A1 \rightarrow A2$$

Proof:

By theorem A.3(2), if b_f^3 implements the transition t_1^3 then:

$$A2 + F = A1 + F - IC(b_f^3) + OC(b_f^3)$$

or:
$$A2 = A1 - IC(b_f^3) + OC(b_f^3) \qquad (B.6.1)$$

But by theorem B.4, b_f^3 can also fire in state A1. Suppose that in state A1, b_f^3 executes the transition:

$$A1 \rightarrow Sj$$

Then, by theorem A.3(2):

$$Sj = A1 - IC(b_f^3) + OC(b_f^3) \qquad (B.6.2)$$

Comparing (B.6.1) and (B.6.2):

$$Sj = A2$$

thus bar b_f^3 executes the transition:

$$t2 = A1 \rightarrow A2$$

Q.E.D.

B.7 Theorem

If A1 + F is a legal state in a TM, and if in the TM there exist a transition ti:

$$ti = A1 \rightarrow A2$$

then there also exist a transition of type t^3:

$$t_k^3 = A1 + F \rightarrow A2 + F$$

Proof:

If: $ti = A1 \rightarrow A2$

then there exists a bar bf that effects ti, and by theorem A.3:

$$IC(bf) \leq A1 \qquad (B.7.1)$$

and: $$A2 = A1 - IC(bf) + OC(bf) \qquad (B.7.2)$$

Since (B.7.1) there also exists:

$$IC(bf) \leq A1 + F$$

This means that the firing algorithm can be applied for bar bf in state A1 + F:

$$tj = A1 + F \rightarrow Sj$$

but, by the firing algorithm:

$$Sj = A1 + F - IC(bf) + OC(bf) \qquad (B.7.3)$$

Comparing (B.7.3) and (B.7.2):

$$Sj = A2 + F$$

thus there exist the transition:

$$tj = A1 + F \rightarrow A2 + F$$

From (B.7.1):

$$\#(F,IC(bf)) \leq \#(F,A1)$$

or: $$\#(F,IC(bf)) < \#(F,A1+F)$$

and by definition, the transition tj:

$$tj = A1 + F \rightarrow A2 + F$$

is of type t_k^3.

Q.E.D.

B.8 Theorem

Let there be a TM with a legal transition:

$$tj = A1 + F \rightarrow A2 + F$$

If there exists another legal transition $tk = S3 \rightarrow S4$, such that:

1. $MIN(tk) = MIN(tj)$ (B.8.1)
2. $S3 \leq A1$ (B.8.2)

then all the possible implementations of tj are of type t^3.

Proof:

(Note that in principle, theorem B.7 is a special case of B.8)

If: $tk = S3 \rightarrow S4$

then by theorem A3, any bar bf that effects tk satisfies:

$$IC(bf) \leq S3 \quad \text{(B.8.3)}$$

and: $S4 = S3 - IC(bf) + OC(bf)$ (B.8.4)

Since (B.8.2), also exist that:

$$IC(bf) \leq A1 \quad \text{(B.8.5)}$$

and also: $IC(bf) \leq A1 + F$

This means that the firing algorithm can be applied for bar bf in state $A1 + F$. Since (B.8.1) and theorems A.2 and A.5, bf also effects transition tj. In other words, for any implementation, the bars that effect tk also effect tj. But from (B.8.5):

$$\#(F, IC(bf)) \leq \#(F, A1)$$

or: $\#(F, IC(bf)) < \#(F, A1+F)$

and by definition, tj is of type t^3, for any implementation.

Q.E.D.

B.9 Theorem

Let there be a TM with a legal transition:

$$tj = A1 + F \rightarrow A2 + F$$

If there is not another legal transition $tk = S3 \rightarrow s4$, such that:

1. $MIN(tk) = MIN(tj)$ (B.9.1)
2. $S3 \leq A1$ (B.9.2)

then there exists an implementation in which tj is of type t^2.

Proof:

Suppose that there is no transition $tk \neq tj$ that satisfies (B.9.1). Then by theorem A.5 (and as discussed in 3.3) there is no interaction between the implementation of tj and the implementation of other transitions in the TM. In other words, bars that effect other transitions will not effect tj. In this case, if tj is effected by a bar bf such that:

$$IC(bf) = A1 + F \quad \text{(B.9.3)}$$
$$OC(bf) = A2 + F$$

then bf is the only bar that effects tj, and since (B.9.3) tj is of type t^2.

Suppose now that exists a group of transitions having the same minimal transition that tj has. This group can be denoted by the set [tq]. There exists:

$$\forall p \quad (tp \in [tq]) \iff (MIN(tp) = MIN(tj))$$

Suppose that (B.9.2) is never satisfied. This means that:

$$\forall p \quad (tp \lessdot [tq]) \Longrightarrow LHS(tk) \not\leq A1 \qquad (B.9.4)$$

The group [tq] can be implemented by a set of bars [bq] such that:

1. $IC(bq) = LHS(tq)$ (B.9.5)
2. $OC(bq) = RHS(tq)$

If bf denotes any bar (of the group [bq]) that fires in A1 + F then:

$$IC(bf) \leq A1 + F \qquad (B.9.6)$$

But from (B.9.4) and (B.9.5):

$$IC(bf) \not\leq A1 \qquad (B.9.7)$$

Comparing (B.9.6) and (B.9.7) there exist that:

$$\#(F, IC(bf)) = \#(F, A1+F) \qquad (B.9.8)$$

Since (B.9.8) exists for any bar (of group [bq]) that fire in A1 + F then tj is effected as of type t^2.

Q.E.D.

B.10 Theorem

Let be a TM with a legal transition:

$$tj = A1 + F \rightarrow A2 + F$$

If there exists a legal state, say S3, such that:

1. $S3 \neq A1 + F$ (B.10.1)
2. $A1 \leq S3$ (B.10.2)
3. if transition tk exits S3 then $MIN(tk) \neq MIN(tj)$.

   In other words:

   $\forall q \quad (LHS(tq)=S3) \Longrightarrow (MIN(tq) \neq MIN(tj))$ (B.10.3)

then all the possible implementations of tj are of type t^2.

Proof:

Suppose that there exists a state S3 that satisfies (B.10.1), (B.10.2) and (B.10.3), but transition tj is of type t^3. In this case, there exists a bar bf such that:

$$IC(bf) \leq A1 + F \qquad (B.10.4)$$

and: $$\#(F,IC(bf)) < \#(F,A1+F) \qquad (B.10.5)$$

From (B.10.4) and (B.10.5):

$$IC(bf) \leq A1 \qquad (B.10.6)$$

From (B.10.6) and (B.10.2):

$$IC(bf) \leq S3$$

this means that bf can fire also in state S3. If when bf fires in S3 effects a transition, say t3 = S3 -> S4 , then by theorem A.5:

$$MIN(t3) = MIN(tj)$$

and this contradict (B.10.3). In other words, if (B.10.1), (B.10.2) and (B.10.3) are satisfied, there is no implementation such that tj is of type t^3. Since tj is a transition between two states that include F, then all the implementations of tj are of type t2.

Q.E.D.

B.11 Theorem

Let be a TM with a legal transition:

$$tj = A1 + F \rightarrow A2 + F$$

If there is not a legal state, say S3, such that:

1. $S3 \neq A1 + F$ (B.11.1)

2. $A1 \leq S3$ (B.11.2)

3. $\forall q \quad (LHS(tq)=S3) \Longrightarrow (MIN(tq) \neq MIN(tj))$ (B.11.3)

then there exists an implementation in which tj is of type t^3.

Proof:

We assume that (B.11.1) is always satisfied. This means that in any TM there is at least two different states.

Suppose that there is no legal state, different from A1 + F, that includes A1. In other words:

$$\forall q \quad (Sq \neq A1+F) \Longrightarrow (A1 \not\leq Sq) \qquad (B.11.4)$$

If tj is implemented by bar bf such that:

$$IC(bf) = A1 \qquad (B.11.5)$$

and:

$$OC(bf) = A2 \qquad (B.11.6)$$

then replacing (B.11.5) in (B.11.4):

$$\forall q \quad (Sq \neq A1+F) \Longrightarrow (IC(bf) \not\leq Sq)$$

In this case bf can fire only in state A1 + F . This means that the implementation described by (B.11.5) and (B.11.6) is possible.

But from (B.11.5):

$$\#(F,IC(bf)) = \#(f,A1)$$

or:

$$\#(F,IC(bf)) < \#(F,A1+F)$$

and bf effects tj as transition of type t^3.

Suppose now that there exist legal states that include A1, but they have exiting transitions with the same minimal

transition that tj has. In other words:

$\forall q\ [(Sq \neq A1+F)\ \text{and}\ (A1 \leq Sq)] \Longrightarrow$

$\Longrightarrow [\exists p\ (LHS(tp)=Sq) \Longrightarrow (MIN(tp)=MIN(tj))]$ (B.11.7)

Suppose again that tj is implemented by bf such that:

$IC(bf) = A1$ (B.11.8)

and: $OC(bf) = A2$ (B.11.9)

From (B.11.8) bf can fire in the states that satisfy:

$A1 \leq Sq$

But from (B.11.7), each state in which bf can fire has an exiting transition with the same minimal transition that tj has. Then, from theorems A.2 and A.5, when bf fires it executes legal transitions. This means that the implementation of (B.11.8) and (B.11.9) is legal. But as was shown before, bf implements tj as a transition of type t^3.

Q.E.D.