A STUDY ON RECOVERABILITY OF PROCESSES

Philip M. Merlin

Technical Report #47
April 1974

Department of Information and Computer Science
University of California, Irvine

This work was supported by the National Science Foundation under Grant GJ-1045.

ACKNOWLEDGEMENT

I would like to express my sincere thanks to Professor David J. Farber for his excellent support and guidance.

1. INTRODUCTION

In a previous paper [1], the author proposed a model for the study of the recoverability of processes under the occurrence of failures. The present paper improves the results of the past paper. An exhaustive analysis of the states, and transitions between states (the TM of [1]) of a process is is presented. A way of designing a Petri net, given the possible states and transitions in the system, is presented in section 2.

Section 3 presents a study of the properties of a system when a failure of type "loss of token" occur. The general structure of a process, in order to be recoverable from that kind of failures, is given. This work shows a way of designing recoverable processes.

The processes studied in this paper are characterized by a lack of knowledge about the execution times of its parts. No assumption is made about the times expended by the events when they occur, or the relation between these times.

The contents of this work is a natural continuation of [1]. This paper assumes that the reader is familiar with the concepts presented in [1]. The same definitions and notations are used.

2. THE PETRI NET OF A GIVEN TOKEN MACHINE

[1] shows the importance of the Token Machine (TM), and its generalization the ETM, in the algorithms that test if a process is recoverable under a given failure. This means that, for a better understanding of recoverability, the properties of the TM have to be studied.

The TM is defined from the Petri Net [1]. In this Section, the properties of the TM are studied, and the problem of constructing a Petri net corresponding to a given TM is analysed.

2.1 GENERAL EXPLANATION

Figure 1 shows a TM. The problem is how to build a Petri net so that its TM is the one shown in figure 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:

$$\{S \cup (A \cup B) \cup (C \cup A) \cup (C \cup D) \cup (C \cup B) \cup (B \cup D) \cup (B \cup B) \cup C\} =$$

$$= \{ S, A, B, C, D \}$$

Each state in the TM corresponds to a possible state in the Petri net. Each arc in the TM correspond to a possible transition in the Petri net. This means that for each arc in the TM there is a bar in the Petri net that can carry out this transition. For example, arc 6 in figure 1 represents a transition from state CD to state CB. That is:

$$CD \rightarrow CB$$

Figure 2 shows the two possible ways of implementing this transition in the Petri net.

Arc 5 represent the transition:

$$BD \rightarrow BB$$

Figure 3 shows the possible bars corresponding to this transition.

One of the bars of figure 2, and one of figure 3, must be in the Petri net . But, the same bar is in figure 2(b) and in figure 3(b). In this case,

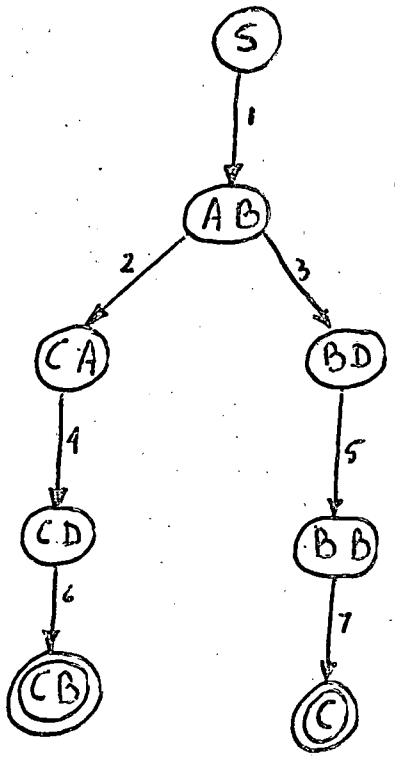
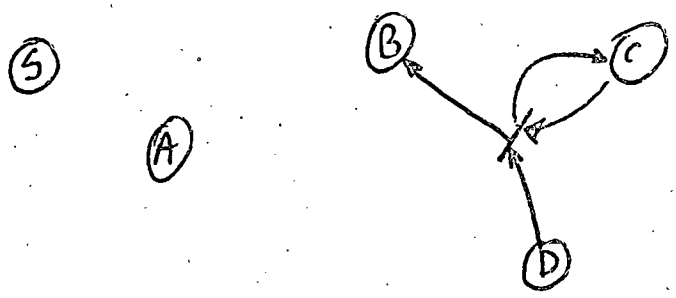
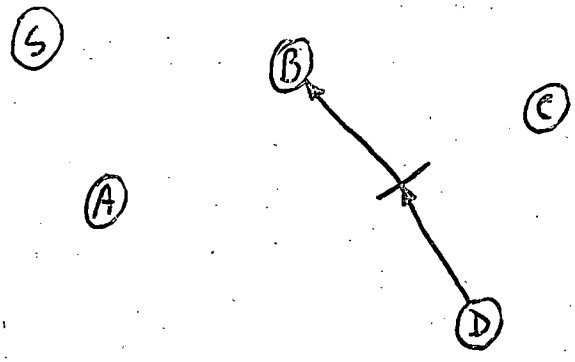


Figure 1 A Token Machine (TM)



(a)



(b)

Figure 2 The Possible Implementations of the Transition CD → CB

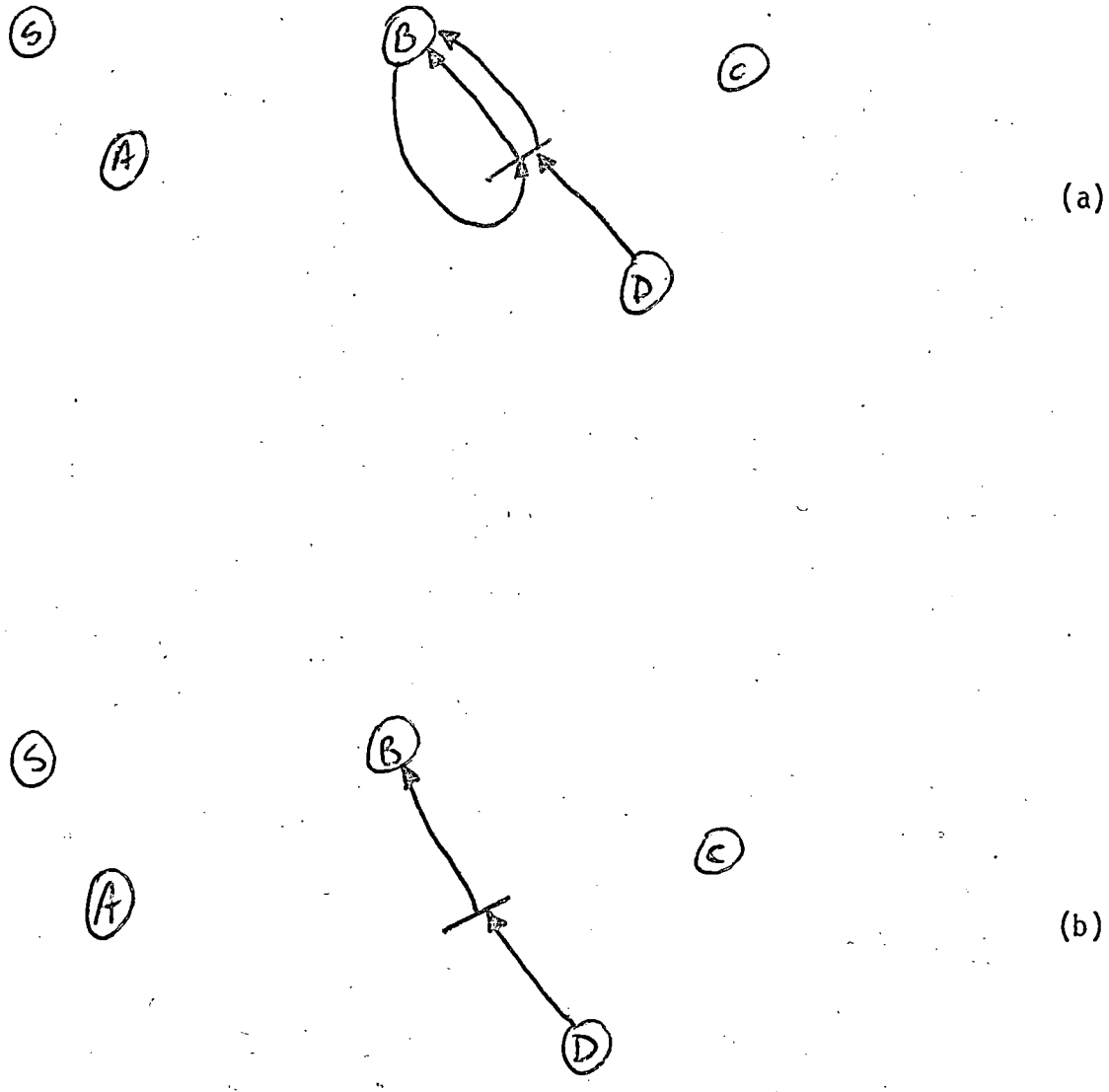


Figure 3 The Possible Implementations of the Transition BD → BB

only one 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 diferent number of bars. The Petri net with the minimal number of bars that implement a given TM is called the Minimal Petri Net (MPN) of the TM.

Arc 2 (figure 1) represent the transition:

$$AB \rightarrow AC$$

Figure 4 shows the possible bars corresponding to this transition.Note that the bar in figure 4(b) can fire each time that B has a token. The states BC, BD, BB are legal states in the TM. In this case, the bar in figure 4(b) can fire also the following transitions:

$$BC \rightarrow CC$$

$$BD \rightarrow CD$$

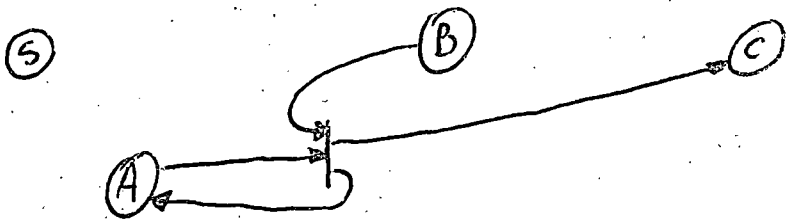
$$BB \rightarrow CB$$

But, since in the TM,there are not arcs corresponding to these transitions, they are not allowed in the Petri net. It means, only the bar of figure 4(a) can be used to implement the arc 2 of figure 1.

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

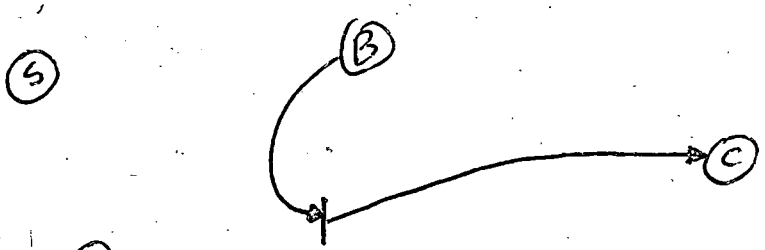
1. the Petri net is not unique.
2. it is necessary to prevent unallowed transitions that can appear as side effects of the implementation of allowed arcs.

Arc 2 (figure 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 conditions. In figure 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 show in figures 2(a) and 3(a). In this case,two bars are needed instead of one.



(a)

D



(b)

D

Figure 4 The Possible Implementations of the Transition AB → AC

In the following Sections the construction of the Petri net of a given TM is formally studied.

2.2 FORMAL DEFINITIONS

2.2.1 A bag is similar to a set but allows multiple instances of the same element. A bag is unordered, as a set is. A bag is denoted with brackets, that is B=[a,b,a] is the bag B of two instances of a and one instance of b.

2.2.2 The Union of a bag (U bag) is the set of elements used to form the bag.

Example: U B = U [a,b,a] = [a,b]

2.2.3 bag_1 is contained in bag_2 ($bag_1 < bag_2$) if there is in bag_2 an instance for each instance of the elements in bag_2.

Example: [a,b,a,b,b,c] < [a,a,a,b,c,c,b,b,d]

2.2.4 Two bags are equal ($bag_1 = bag_2$) if:

$$bag_1 < bag_2$$

and $$bag_2 < bag_1$$

2.2.5 The sum of two bags ($bag_1 + bag_2$) is a bag compose by all the instances in bag_1 and in bag_2.

Example: [a,b,a] + [a,b,c,b] = [a,a,a,b,c,b,b]

2.2.6 The substraction of two bags ($bag_1 - bag_2$) is bag_3, so that:

$$bag_3 + bag_2 = bag_1$$

We do not deal with "negative elements", so that this operation is defined only when $bag_2 < bag_1$.

2.2.7 A transition is an ordered tuple of two bags. It means, ALPHA and BETA are bags, and:

$$t = ALPHA \rightarrow BETA$$

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

1. $LHS(t) = ALPHA$
2. $RHS(t) = BETA$

2.2.8 The Remainer of a transition ($RE(bag_1 \rightarrow bag_2)$) is the bag_3 with the maximal number of instances of the elements so that:

1. $bag_3 < bag_1$; and
2. $bag_3 < bag_2$

2.2.9 The Minimum of a transition ($MIN(t_1)$) is a transition t_2, so that if

$t_1 = bag_1 \rightarrow bag_2$; then:

1. $LHS(t_1) = bag_1 - RE(t_1)$
2. $RHS(t_1) = bag_2 - RE(t_1)$

Note that $LHS(t_1)$ and $RHS(t_2)$ have no elements in common.

By definition, any transition t is:

$$t = RE(t) + LHS(MIN(t)) \rightarrow RE(t) + RHS(MIN(t))$$

EXAMPLE:

arc 5 of figure 1 can be denoted by:

$t_5 = [B,D] \rightarrow [B,B]$; then:

1. $LHS(t_5) = [B,D]$
2. $RHS(t_5) = [B,B]$
3. $RE(t_5) = [B]$
4. $MIN(t_5) = [D] \rightarrow [B]$

2.2.10 A Petri net (PN) is a directed graph defined by the quadruplet $[B,C,A,S_0]$, where:

$B = [b_1,\ldots .,b_m]$ is a finite set of transition bars

$C = [c_1,\ldots .,c_n]$ is a finite set of conditions

B and C are the nodes of the PN

$A = [a_1,\ldots .,a_q]$ is a finite set of directed arcs. Each arc conect either a condition to a bar or a bar to a condition.

S_0 = bag the elements of S_0 are elements from C. That means $U(S_0) < C$; S_0 is the initial distribution of tokens in the conditions C.

2.2.11 The Input Conditions of a bar ($IC(b_i)$) is defined as a bag:

$$IC(b_i) = bag_i$$

The elements of bag_i belong to C, it means:

$$U(bag_i) < C$$

bag_i include an instance of c_j ($j = 1,2,\ldots ,n$) for each arc which conect c_j to b_i .

2.2.12 The Output Conditions of a bar ($OC(b_i)$) is defined as a bag:

$$OC(b_i) = bag_i$$

The elements of bag_i belongs to C, it means:

$$U(bag_i) < C$$

bag_i include an instance of c_j ($j = 1,2,\ldots ,n$) for each arc which conect b_i to c_j .

2.2.13 A Legal State (S_i) of the PN is either the bag S_0 or a bag that is a possible result of the firing algorithm (2.2.14).

2.2.14 The Firing Algorithm

A bar b_i can fire if $IC(b_i) < S_j$ (S_j is a legal state). If b_i fires, a token is removed from each condition in $IC(b_i)$, and a token is placed in each in each condition of $OC(b_i)$. The new distribution of the tokens defines a new legal state S_k.

Following the definitions, the new state is:

$$S_k = S_j - IC(b_i) + OC(b_i)$$

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

$S = [S_1, S_2, \ldots ., \ldots]$ is a set of bags. S_i belongs to S if and only if S_i is a legal state of the PN. S may be finite or infinite, depending on the PN.

$T = [t_1, t_2, \ldots\ldots\ ]$ is a set of transitions of the form $t_k = S_i \rightarrow S_j$. t_k belongs to T if and only if in state S_i a bar can fire bringing the PN to the state S_j.

2.2.16 A live PN is a PN that:

1. for each c_i in C exist at least one element in S in which c_i is included and
2. for each bar b_i in B exist at least one state in S in which b_i can fire.

2.3 FORMAL ANALYSIS

Suppose that a finite TM is given. The problem is to find a PN so that its TM is the given one. We limit our work to live PN's (definition 2.2.16).

Since the PN is live, 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:

$$C = U(\sum_{i=1}^{n} S_i)$$

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

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

The following theorems will help to attain this goal.

2.3.1 THEOREM

If exist a transition $t_1 = S_i \rightarrow S_j$ so that $MIN(t_1) = a \rightarrow b$; and a bar b_e so that:

1. $IC(b_e) = a + x$ (x is a bag)
2. $OC(b_e) = b + x$
3. $x < RE(t_1)$;

then, t_1 is executed by b_1.

PROOF:

By definition 2.2.9:

$$S_i = a + RE(t_1) \qquad (2.3.1.1)$$

$$S_j = b + RE(t_1) \qquad (2.3.1.2)$$

Since is given that $x < RE(t_1)$ then from (2.3.1.1):

$$a + x < S_i$$

and: $$IC(b_e) < S_i$$

By definition of the firing algorithm (2.2.14), in state S_i the bar b_e can fire and the new state is:

$$S_k = S_i - IC(b_e) + OC(b_e)$$

and after replacement of $IC(b_e)$ and $OC(b_e)$ from the conditions of the theorem, and S_i from (2.3.1.1):

$$S_k = a + RE(t_1) - a - x + b + x = b + RE(t_1) \qquad (2.3.1.3)$$

Comparing (2.3.1.2) and (2.3.1.3):

$$S_j = S_k$$

Q.E.D.

2.3.2 THEOREM

If there is one transition $t_1 = S_i \rightarrow S_j$ so that $MIN(t_1) = a \rightarrow b$, there is not another transition t_2 so that $MIN(t_1) = MIN(t_2)$ and $LHS(t_2) = S_i$.

PROOF:

If, $t_1 = S_i \rightarrow S_j$ exists

and: $MIN(t_1) = a \rightarrow b$ then

by definition (2.2.9): $RE(t_1) = S_i - a$

and: $S_j = b + RE(t_1) = S_i + b - a \qquad (2.3.2.1)$

Suppose that there exists: $t_2 = S_i \rightarrow S_k$

and: $MIN(t_1) = a \rightarrow b$ then

By definition (2.2.9): $RE(t_1) = S_i - a$

and $S_k = b - RE(t_2) = S_i + b - a \qquad (2.3.2.2)$

Comparing (2.3.2.1) and (2.3.2.2):

$$S_j = S_k$$

thus: $t_1 = t_2$

This means, t_1 and t_2 are the same transition.

Q.E.D.

2.3.3 THEOREM

If the transition $t_k = S_i \rightarrow S_j$ is executed by the bar b_e then:

(1) $IC(b_e) < S_i$

(2) $S_j = S_i - IC(b_e) + OC(b_e)$

PROOF:

(1) If t_k is executed by b_e, then the firing algorithm (2.2.14) have to be applied to S_i for the bar b_e. It means:

$$IC(b_e) < S_i$$

(2) After b_e fires in state S_i, the new has to be the new state in the firing algorithm:

$$S_j = S_i - IC(b_e) + OC(b_e)$$

2.3.4 THEOREM

If the transition $t_k = S_i \rightarrow S_j$ is executed by the bar b_e then:

(1) $S_i - RE(t_k) < IC(b_e)$

(2) $S_j - RE(t_k) < OC(b_e)$

PROOF:

From theorem (2.3.3):

$$S_j = S_i - IC(b_e) + OC(b_e)$$

or: $$S_j + IC(b_e) = S_i + OC(b_e) \qquad (2.3.4.1)$$

If we denote $LHS(MIN(t_k)) = a$ and $RHS(MIN(t_k)) = b$, then by definition (2.2.9):

$$S_i = a + RE(t_k) \qquad (2.3.4.2)$$

and: $$S_j = b + RE(t_k) \qquad (2.3.4.3)$$

replacing (2.3.4.2) and (2.3.4.3) in (2.3.4.1) yields:

$$b + RE(t_k) + IC(b_e) = a + RE(t_k) + OC(b_e)$$

and this means that:

$$b + IC(b_e) = a + OC(b_e)$$

But, by definition 2.2.9 a and b have no common elements. So that, all the elements of a have to be included in $IC(b_e)$, and all the elements of b in $OC(b_e)$. It means:

$$a < IC(b_e)$$

and: $$b < OC(b_e)$$

and replacing a and b from (2.3.4.2) and (2.3.4.3) then:

$$S_i - RE(t_k) < IC(b_e)$$

and: $$S_j - RE(t_k) < OC(b_e)$$

Q.E.D.

2.3.5 THEOREM

If transitions $t_1 = S_{i1} \rightarrow S_{j1}$ and $t_2 = S_{i2} \rightarrow S_{j2}$ are implemented by the same bar (say b_e), then $MIN(t_1) = MIN(t_2)$.

PROOF:

Suppose that: $MIN(t_1) = a_1 \rightarrow b_1$

and: $MIN(t_2) = a_2 \rightarrow b_2$

If transition t_1 is executed by the bar b_e, then by theorem 2.3.4:

$$S_{i1} - RE(t_1) < IC(b_e)$$

This means, that there exists a bag x_1 such that:

$$IC(b_e) = S_{i1} - RE(t_1) + x_1 \qquad (2.3.5.1)$$

Thus from theorem 2.3.3 and replacement of $IC(b_e)$:

$$OC(b_e) = S_{j1} - S_{i1} + IC(b_e) = S_{j1} - S_{i1} + S_{i1} - RE(t_1) + x_1$$

$$OC(b_e) = S_{j1} - RE(t_1) + x_1 \qquad (2.3.5.2)$$

and by definition 2.2.9:

$$S_{i1} - RE(t_1) = a_1 \qquad (2.3.5.3)$$

$$S_{j1} - RE(t_1) = b_1 \qquad (2.3.5.4)$$

Replacing (2.3.5.3) in (2.3.5.1) and (2.3.5.4) in (2.3.5.2):

$$IC(b_e) = a_1 + x_1$$

and: $$OC(b_e) = b_1 + x_1$$

In the same way for transition t_2:

$$IC(b_e) = a_1 + x_1 = a_2 + x_2 \qquad (2.3.5.5)$$

$$OC(b_e) = b_1 + x_1 = b_2 + x_2 \qquad (2.3.5.6)$$

By the definition of equality, it means:

$$a_1 + x_1 < a_2 + x_2$$

and: $$b_1 + x_1 < b_2 + x_2$$

This means that:

$$x_1 < a_2 + x_2$$

and: $$x_1 < b_2 + x_2$$

But, by the defintion of $MIN(t_2)$ (definition 2.2.9) a_2 and b_2 have no common elements. This means that no elements of x_1 are included in both, a_2 and b_2. Thus, x_1 is included in x_2, or in other words:

$$x_1 < x_2$$

But, from equations (2.3.5.5) and (2.3.5.6) in the same way:

$$x_2 < x_1$$

so that, from the definition of equality (2.2.4):

$$x_2 = x_1$$

and from (2.3.5.5) and (2.3.5.6) then:

$$a_1 = a_2$$

and: $$b_1 = b_2$$ Q.E.D.

2.3.6 DISCUSSION

From theorem 2.3.5, it turns out that transitions with different "minimal transition" must be implemented by different bars. This means that the set of transitions can be divided into different groups, each group having the same minimal transition. Each group is implemented independently of the others.

Suppose the members of a group are:

$$t_{k_p} = S_{i_p} \rightarrow S_{j_p} \quad ; (p=1,2,\ldots .r) \text{ then}$$

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

$$MIN(t_{k_p}) = a \rightarrow b \quad ; (p=1,2,\ldots .r)$$

Theorem 2.3.1 shows that each member of this group can be implemented by a bar b_p, thus:

1. $IC(b_p) = a + x_p$ (2.3.6.1)
2. $OC(b_p) = b + x_p$ (2.3.6.2)
3. $x_p < RE(t_{k_p})$ (2.3.6.3)

Theorem 2.3.4 shows that this is the only way to implement them. As shown above, there is a certain freedom in the election of the x_p. This means that the members of each group can be implemented in different ways, leading to different PN's. The same x_p can be chosen for different transitions (in the same group). In this case, the same bar executes several transitions. In general, the number of bars that implement the group is equal to the number of different x_p's chosen for this group. A trivial case is:

$$x_p = \emptyset \quad ; (p=1,2,\ldots..r)$$

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 x_p is chosen for equations (2.3.6.1), (2.3.6.2) and (2.3.6.3), by definition of the firing algorithm, bar b_p can fire in all the states S_i that:

$$IC(b_p) < S_i$$

By theorem 2.3.5, all the transitions executed by b_p have the same minimum:

$$a \rightarrow b$$

and by theorem 2.3.2, in each state S_i can be only one transition with minimal: $a \rightarrow b$. Thus, when the machine is in state S_{i_p}, then b_p will execute only legal transitions, and when the machine is in the state:

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

if b_p fires, it will execute an illegal transition. So, in order to prevent illegal transitions, x_p has to satisfy the condition:

$$IC(b_p) = a + x_p \not< S_k \quad ; \text{ for all } S_k \neq S_{i_p} \qquad (2.3.6.4)$$

(Note: $x \not< y$ means x is not included in y or, alternatively, it is not true that $x < y$)

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 $[x_q]$, subset of the bag $[x_p]$, so that:

$$\forall_p \exists_q \quad x_q < RE(t_{k_p}) \qquad (2.3.6.5)$$

and

$$\forall_q \forall_k \quad (k \neq i_p) \Rightarrow a + x_q \not< S_k \qquad (2.3.6.6)$$

Note that it is enough to check (2.3.6.6) for the states S_k that include a. If S_k does not include a, (2.3.6.6) is satisfied independently on x_q.

In general, the following cases may exist:

1. Only one set $[x_q]$ that satisfies (2.3.6.5) and (2.3.6.6). In this case, there exists only one PN corresponding to the given TM.
2. Several sets $[x_q]$ that satisfy (2.3.6.5) and (2.3.6.6). In this case,

there exists several PN's corresponding to the given TM.

3. There are no $[x_q]$ that satisfy also (2.3.6.5) and (2.3.6.6). In this case, there is not a PM corresponding to the given TM. Note that there not exits a set $[x_q]$ that satisfy (2.3.6.5) and (2.3.6.6) if and only if there is a state S_k ,$(k \neq i_p)$ included in a state S_{i_p} .
The two following theorems proof this statement.

2.3.7 THEOREM

$$\text{If:} \quad \forall_p \forall_k \quad (k \neq i_p) \Rightarrow S_{i_p} \not< S_k \qquad (2.3.7.1)$$

then exist at least one set $[x_q]$ that satisfy (2.3.6.5) and (2.3.6.6)

PROOF:

Suppose that the elements of $[x_q]$ are chosen such that:

$$x_q = x_p = RE(t_{k_p}) \qquad (2.3.7.2)$$

THEN:

(1) by definition, the elements of x_q satisfy (2.3.6.5)

(2) but, by (2.3.7.2):

$$q = p$$

and:

$$a + x_q = a + x_p = a + RE(t_{k_p}) = S_{i_p}$$

and replacing S_{i_p} in (2.3.7.1) then the result is (2.3.6.6). Q.E.D.

Q.E.D.

2.3.8 THEOREM

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

$$S_{i_p} < S_k$$

then, there does not exist a set $[x_q]$ such that the conditions (2.3.6.5) and (2.3.6.6) are satisfied.

PROOF:

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

$$S_{i_{pl}} < S_{kl} \qquad (2.3.8.1)$$

Suppose that there exist a set $[x_q]$ such that (2.3.6.5) and (2.3.6.6) are satisfied. Applying (2.3.6.6) for kl and for pl the result is:

$$\forall q \quad (kl \neq i_{pl}) \Rightarrow a + x_q \not< S_{kl}$$

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

$$\forall q \quad a + x_q \not< S_{kl}$$

If, for all q, S_{kl} does not include $a + x_q$, then since $S_{i_{pl}}$ is included in S_{kl} (2.3.8.1) there exists:

$$\forall q \quad a + x_q \not< S_{i_{pl}}$$

This means that:

$$\forall q \quad x_q \not< S_{i_{pl}} - a \qquad (2.3.8.2)$$

Since by definition 2.2.9:

$$S_{i_{pl}} - a = RE(t_{k_{pl}}) \qquad (2.3.8.3)$$

Replacing (2.3.8.3) in (2.3.8.2), there exist that:

$$\forall q \quad x_q \not< RE(t_{k_{pl}}) \qquad (2.3.8.4)$$

But (2.3.8.4) contradicts (2.3.6.5). Therefore, set $[x_q]$ does not exist. Q.E.D.

2.4 EXAMPLES

The previous theorems (2.3.7 and 2.3.8) gives the necessary and sufficient conditions so 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 in the properties wanted in the PN (minimal number of bars, some special configuration, etc.), there exists several algorithms to choose the elements of the sets $[x_q]$ (when such sets exist). One possibility, as shown in theorem 2.3.7, is to choose $x_q = x_p = RE(t_{k_p})$. But, this solution gives a large number of bars. In the following examples, some possibilities are shown.

(1) First, the implementation of the TM of figure 1 is shown. The groups of transitions are shown in table 1.

TABLE 1

group	mimimal transition	transition	RE(t)
1	S → AB	S → AB	∅
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	∅

For group 1:

$$IC(b^1) = [S] + x$$

since no states outside the group include S, x=∅ is chosen. Then:

$$IC(b^1) = [S]$$

$$OC(b^1) = [A,B]$$

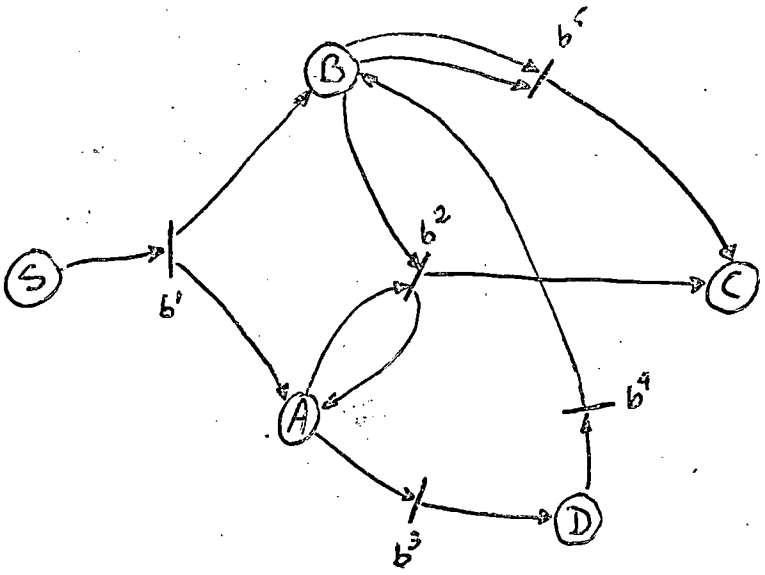


Figure 5 A PN that Implement the TM of Figure 1

For group 2:

$$IC(b^2) = [B] + x$$

The states outside the group that include B are: BD, BB, BC.

If x = [A] , conditions (2.3.6.5) and (2.3.6.6) are satisfied. Thus:

$$IC(b^2) = [A,B]$$

$$OC(b^2) = [A,C]$$

For group 3:

$$IC(b^3) = [A] + x$$

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

$$IC(b^3) = [A]$$

$$OC(b^3) = [D]$$

Also for groups 4 and 5 , x = $\emptyset$ is chosen in the same way.

The corresponding PN is shown in figure 5.

(2) In the following example we will try implementing a PN that has the TM shown in figure 6. The groups of transitions are given in table 2.

TABLE 2

Group	minimal	transition	RE(t)
1	A → B	A → B	$\emptyset$
2	A → C	A → C	$\emptyset$
3	B → CD	B → CD	$\emptyset$
4	C → BE	C → BE	$\emptyset$
5	C → E	CD → ED	D
6	B → D	BE → DE	E
7	DE → F	DE → F	$\emptyset$

For group 6: $IC(b^6) = [B] + x$

and: $x < E$

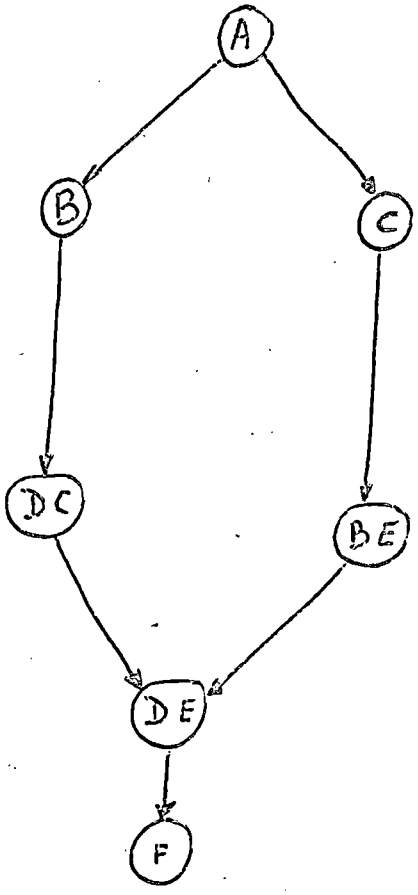


Figure 6 A Token Machine (TM)

It means, x is or Ø or E. But B is a legal state, outside the group 6. Theorem 2.3.8 shows that this case is impossible to implement.since group 6 is implemented by:

$$IC(b^6) = [B]$$

or:

$$IC(b^6) = [B,E]$$

(these are the only possibilities), it also will implement a transition from B to D, and this is a transition that does not exist in the TM.

3. THE TM AND THE LOSS OF A TOKEN

In this section, the structure of the TM's which are recoverable from a failure of type "loss of a token" is studied. The results of this section yields a better understanding of the concept "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 section 2.

This work deals only with failures of type "loss of a token", as defined in [1]. The same ideas are applicable to other kinds of failures.

In [1], the definition of "legal states" of a process is given as the states of the correspondent TM. "Illegal states" are defined as the states, different of the legal states, in which the PN can be after the occurrence of a failure. In [1], it is also pointed out that a PN with finite TM is recoverable from a given failure if:

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

3.1 The states after a 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.

3.1.1 Definitions and properties

We suppose that F is the name of the failing condition. This means, a

token in F may disapear. The set S of all the states in the TM are divided in two groups:

1. $S^1 = [S_i \mid S_i \not< F]$
2. $S^2 = [S_i \mid S_i < F]$

By definition, S^1 and S^2 have no common elements, and:

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

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

In order to simplify the notation, the two following definitions are introduced:

1. A_i is defined as:

   $$A_i = S_i^2 - F$$

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

   $$S_i^2 = A_i + F$$

2. The function #(a,bagl) is defined as the number of instances of the elements a in the bag bagl.

The transitions exiting from the states S_i^2 can be divided into the three following groups:

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

2. $t_k^2 = S_i^2 \rightarrow S_j^2$; and if t_k^2 is executed by the bar b_e then

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

3. $t_k^3 = S_i^2 \rightarrow S_j^2$; and there exists a bar b_e that execute t_k^3 such that

$$\#(F,S_i^2) > \#(F,IC(b_e))$$

For t_k^2 and t_k^3, the relation between the number of instances of F in the initial state and in the input conditions of the bar that executes the transition is given explicitly by the definitions above. As shown in the following, for t_k^1, this relation is given implicitly by its definition.

By definition, F is not included in S_j^1 . By definitions 2.2.8 and 2.2.9, if:

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

then:

$$\#(F,S_i^2) = \#(F,LHS(MIN(t_k^1)))$$

and if t_k^1 is implement by bar b_e, from theorem 2.3.4(1) and definition 2.2.14:

$$LHS(MIN(t_k^1)) < IC(b_e) < S_i^2$$

so that:

$$\#(F,S_i^2) = \#(F,IC(b_e)) \qquad (3.1.1.1)$$

3.1.2 The possible states after a failure

Suppose that the TM is in state $S_i^2 = A_{.i} + F$. If while in this state a failure occurs, then TM will go to state A_i. In the following steps, we assume that the PN is in state A_i , and , the bars that can fire in this situation will be studied. When the machine is in state $A_i + F$ then the set of all the bars are divided in four exclusive groups:

1. b^1; bars that fire transitions of type t_k^1,
2. b^2; bars that fire transitions of type t_k^2,
3. b^3; bars that fire transitions of type t_k^3 and
4. b^4 ; bars that can not fire

Each group is

3.1.2.1 THEOREM

Bars of type b^1 can not fire in state A_i.

PROOF:

Suppose that b_e^1 is a bar of type b^1. (3.1.1.1) shows that:

$$\#(F,A_i + F) = \#(F,IC(b_e^1))$$

It means, in A_i there are less instances of F than in $IC(b_e^1)$. In other words:

$$IC(b_e^1) \not\prec A_i$$

and this contradict the firing algorithm in 2.2.14, so that b_e^1 can not fire in A_i and thus not for any b^1

3.1.2.2 THEOREM

Bars of type b^2 can not fire in state A_i.

PROOF:

Suppose that b_e^2 is a bar of type b^2. Following the definition of transitions of type t_k^2 (section 3.1.1):

$$\#(F,A_i + F) = \#(F,IC(b_e^2))$$

and then the proof continues as in the previous theorem.

3.1.2.3 THEOREM

Bars of type b^3 can fire in state A_i.

PROOF:

Suppose that b_e^3 is a bar of type b^3. If b_e^3 can fire in $A_i + F$ then:

$$IC(b_e^3) < A_i + F \qquad (3.1.2.3.1)$$

But by definition of transitions of type t_k^3 the number of instances of F in $A_i + F$ is greater than those instances in $IC(b_e^3)$. This means that the number of instances of F in A_i is equal or greater than those instances in $IC(b_e^3)$. But, by (3.1.2.3.1), the instances of the other elements of A_i are also included in $IC(b_e^3)$, thus

$$IC(b_e^3) < A_i$$

and the firing algorithm can be applied.

3.1.2.4 THEOREM

Bars of type b^4 can not fire in state A_i.

PROOF:

Suppose that b_e^4 is a bar of type b^4. By definition, b_e^4 can not fire in $A_i + F$. It means, the firing algorithm can not be applied. The firing algorithm can not be applied only if:

$$IC(b_e^4) \not< A_i + F$$

but:

$$A_i < A_i + F$$

Then:

$$IC(b_e^4) \not< A_i$$

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

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

3.1.2.5 THEOREM

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

$$t_1^3 = A_1 + F \rightarrow A_2 + F$$

$$t_2^3 = A_1 \rightarrow A_2$$

PROOF:

By theorem 2.3.3(2), if b_e^3 implements the transition t_1^3 then:

$$A_2 + F = A_1 + F - IC(b_e^3) + OC(b_e^3)$$

or: $$A_2 = A_1 - IC(b_e^3) + OC(b_e^3) \qquad (3.1.2.5.1)$$

But by theorem 3.1.2.3, b_e^3 can also fire in state A_1. Suppose that in state A_1, b_e^3 executes the transition:

$$A_1 \rightarrow S_j$$

Then, by theorem 2.3.3(2):

$$S_j = A_1 - IC(b_e^3) + OC(b_e^3) \qquad (3.1.2.5.2)$$

Comparing (3.1.2.5.1) and (3.1.2.5.2):

$$S_j = A_1 \quad \text{Q.E.D.}$$

3.1.2.6 THEOREM

If A_1, A_2 and $A_1 + F$ are legal states in a TM, and if there exists a

$$t_i = A_1 \rightarrow A_2$$

then there also exits a:

$$t_k^3 = A_1 + F \rightarrow A_2 + F$$

PROOF:

If: $$t_i = A_1 \rightarrow A_2$$

then there exists a bar b_e, and by theorem 2.3.3:

$$IC(b_e) < A_1 \qquad (3.1.2.6.1)$$

and: $$A_2 = A_1 - IC(b_e) + OC(b_e) \qquad (3.1.2.6.2)$$

Since (3.1.2.6.1) also exist

$$IC(b_e) < A_1 + F$$

This means that the firing algorithm can be applied for bar b_e in state $A_1 + F$:

$$t_j = A_1 + F \rightarrow S_j$$

but, by the firing algorithm:

$$S_j = A_1 + F - IC(b_e) + OC(b_e) \qquad (3.1.2.6.3)$$

Comparing (3.1.2.6.3) and (3.1.2.6.2):

$$S_j = A_2 + F$$

From (3.1.2.6.2):

$$\#(F, IC(b_e)) \leq \#(F, A_1)$$

or: $$\#(F, IC(b_e)) < \#(F, A_1 + F)$$

and by definition, the transition t_j:

$$t_j = A_1 + F \rightarrow A_2 + F$$

is of type t_k^3. Q.E.D.

3.1.2.7 DISCUSSION

The previous theorems show that, after a failure, the PN can execute only transitions implemented by the bars of the group b^3, corresponding to transitions of the group t_k^3.

If: $$t_k^3 = A_1 + F \rightarrow A_2 + F$$

it was shown (theorem 3.1.2.5) that there exits the transition:

$$A_1 \rightarrow A_2$$

Theorem 3.1.2.6 shows that all the transitions from A_1 have a correspondent t_k^3 transition from $A_1 + F$. This means that there is one-to-one correspondence between all the transitions from A_1 and all the transitions of type t_k^3 from $A_1 + F$. The same properties hold for all the states A_i.

After the transition from A_1 the PN is in state A_2. But, the machine can arrive to A_2 also if the PN is in state $A_2 + F$ and then there occurs the failure. This means that the previous theorems are appliable when the PN is in A_2. In other words, from A_2 exits only transitions corresponding to the transitions of type t_k^3 exiting from $A_2 + F$. Thus, after the ocurrence of a failure the PN execute only transitions between the states of the group A_i. This procedure can be

continued until:

1. either it arrives to a A_i, that correspond to a legal state $A_i + F$ that does not have exiting transitions of type t_k^3. In this case, there are no transitions exiting from A_i. A_i is a terminal state.
2. or a loop of states A_{i_k} (k= 1,2,...,r) is executed. This loop corresponds to a loop of transitions of type t_e^3 between the states $A_{i_k} + F$.

3.2 The structure of the recoverable TM's

As shown in reference [1], 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.

The previous section shows that after the occurrence of a failure the illegal states are A_i, corresponding to legal states $A_i + F$. Since this work deals only with finite TM's, the number of legal states is finite. It means the number of illegal states (the number of different A_i) 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 have to be legal. This implies that if $A_i + F$ is a legal state, and there are no transitions of type t_k^3 exiting from $A_i + F$, then, A_i has to be a legal state.

Condition (3) for recoverability points out that loops of only illegal states are not allowed. If, in the TM, there exists loops of states $A_i + F$, connected by transitions of type t_k^3, then, after a failure, there exists corresponding loops of states A_i. In order to satisfy condition 3, at least one of the A_i states of each loop has to be a legal state. But, if a state is legal, all the successors of this state are legal

(definition 2.2.13 and 2.2.14). Thus, all the loops of states A_i, connected by transitions of type t_k^3 have to be legal and the successors of the states A_i in the loop are also 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). But from the properties of the TM it is possible, for certain cases, to distinguish between them directly from the TM. The following properties are derived from the theorms and definitions in the previous sections:

1. If A_1, A_2 and $A_1 + F$ are legal states, and if exist $t_i = A_1 \rightarrow A_2$, then there exists a $t_j = A_i + F \rightarrow A_2 + F$, and t_j is of type t_k^3.
   (this property is proved in theorem 3.1.2.6)
2. If there exists a transition $t_j = A_1 + F \rightarrow A_2 + F$, and if A_1 is also a legal state, and the transition $A_1 \rightarrow A_2$ does not exist, then, t_j is a transition of type t_k^2.
   (this property can be proved from the theorems of section 2 and definition of transitions of type t_k^2.)
3. If there exists a transition $t_j = A_1 + F \rightarrow A_2 + F$, and if A_1 is not a legal state, then transition t_j can be implemented as either of type t_k^2 or t_k^3.
   (this property can be proved from the theorems of section 2 and definitions of transitions of type t_k^2 and t_k^3.)

The previous properties show that the decision of whether a transition is t_k^2 or t_k^3 may be made, in part, by the designer of the system.

3.2.1 RECAPITULATION

A finite TM is recoverable under a loss of a token from condition F if and only if:

1. if $A_i + F$ is a legal state without exiting transitions of type t_k^3, then A_i is a legal state in the TM.
2. if $A_{i_j} + F$ (j=1,2...,r) are legal states connected in a directed loop by transitions of type t_k^3, and exist a path of transition of type t_k^3 from the states A_{i_j} to the states $A_{i_w} + F$ (w=r+1,r+2,....,p), then, the states A_{i_j} (j=1,2,.....p) are legal states.

As shown above, exist a one to one correspondence between all the transitions from the states A_{i_j} and the transitions t_k^3 from the $A_{i_j} + F$ states.

3.2.2 EXAMPLES

Figure 7 shows an example of a TM. The TM has no loops of states including F. From state FC there does not exist transitions to other states that include F. Thus, there does not exist exit transitions from FC of type t_k^3. In other words the TM is recoverable if C is a terminal state. Figure 8 shows the corresponding PN. The PN was constructed using the tools developed in section 2. In order to demonstrate the existence of recoverability, figure 9 shows the corresponding ETM (the ETM is defined in reference [1]).

Figure 10 shows an example of a TM with loops of transitions of type t_k^3. The TM is recoverable. The corresponding PN is shown in figure 11.

Figure 12 shows other example of recoverable TM. Figure 13 shows the corresponding PN, and figure 14 shows the ETM.

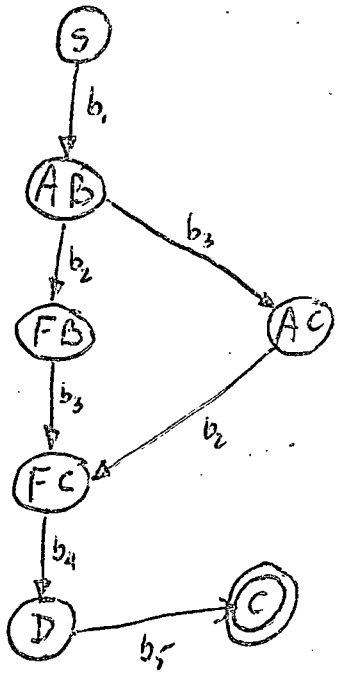


Figure 7 A Recoverable TM

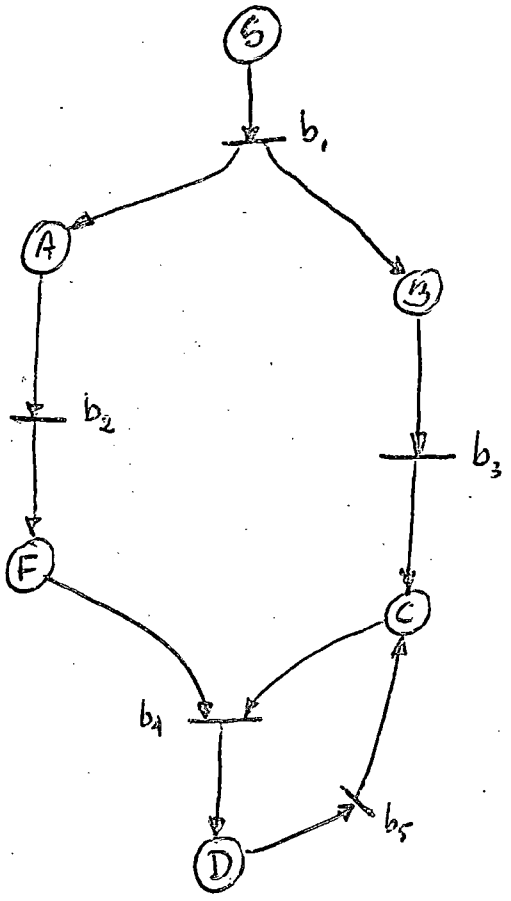


Figure 8 A PN for the TM of Figure 7

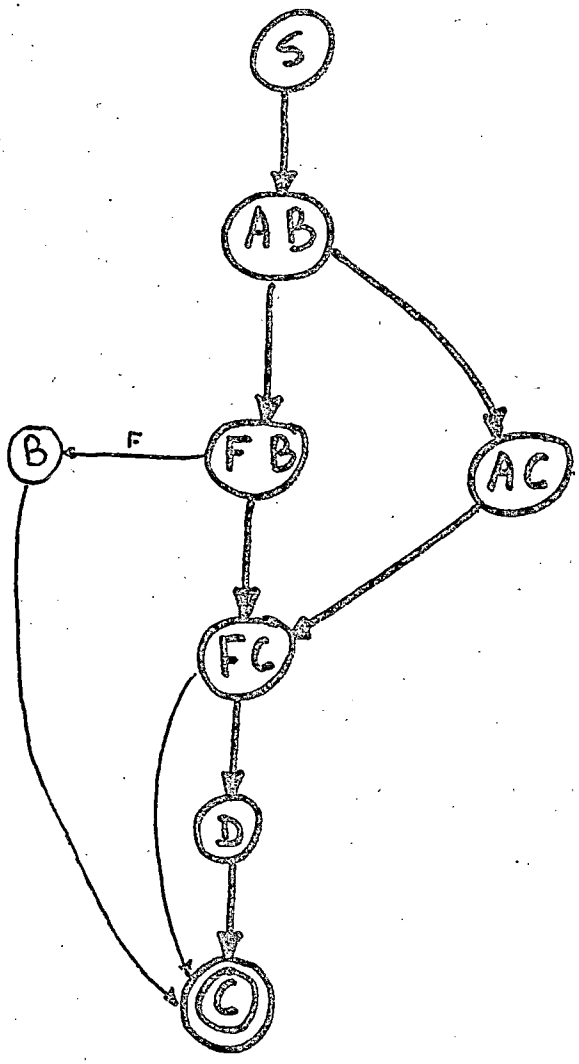


Figure 9 The ETM for the PN of Figure 8

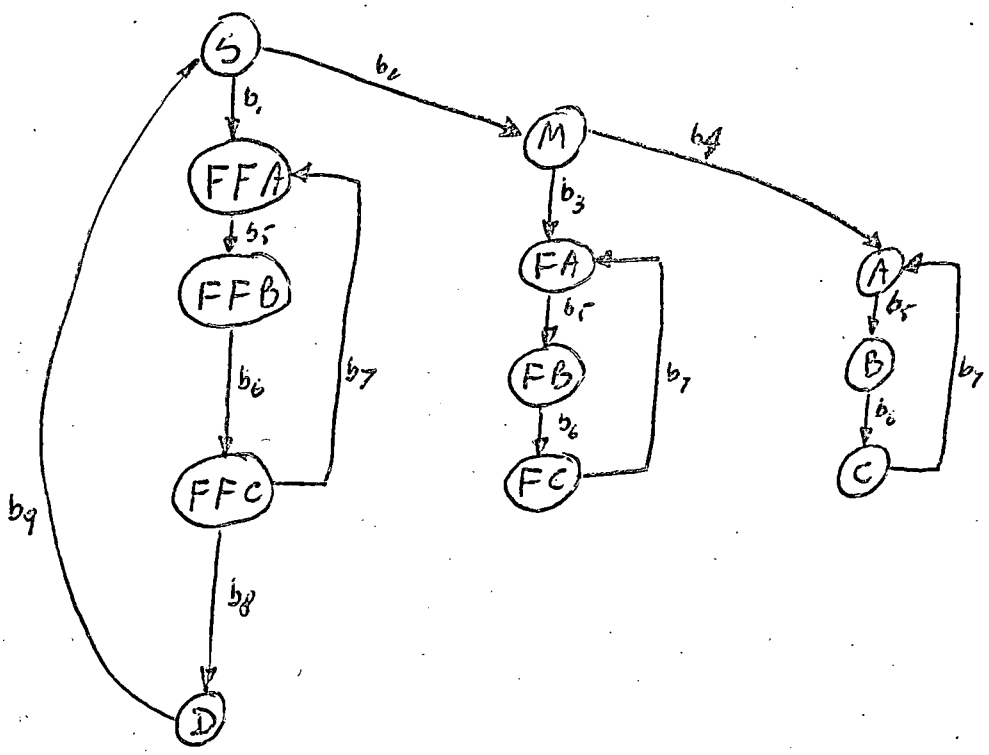


Figure 10 A Recoverable TM

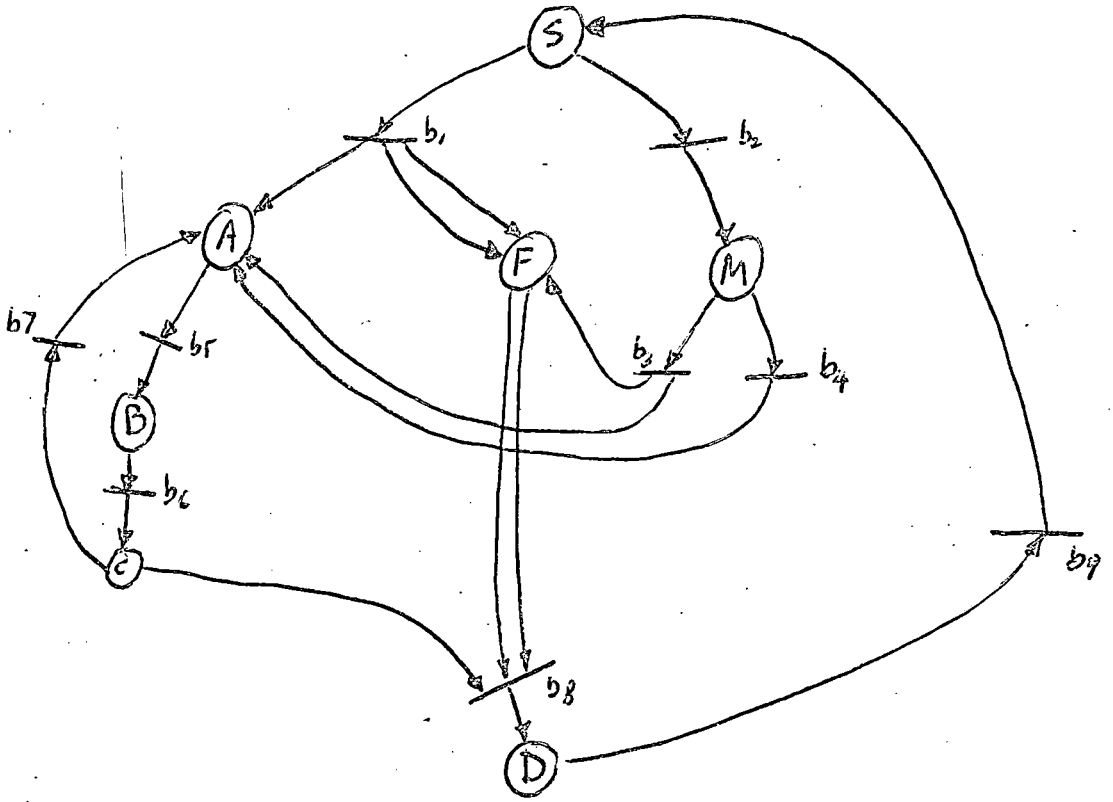


Figure 11 A PN for the TM of Figure 10

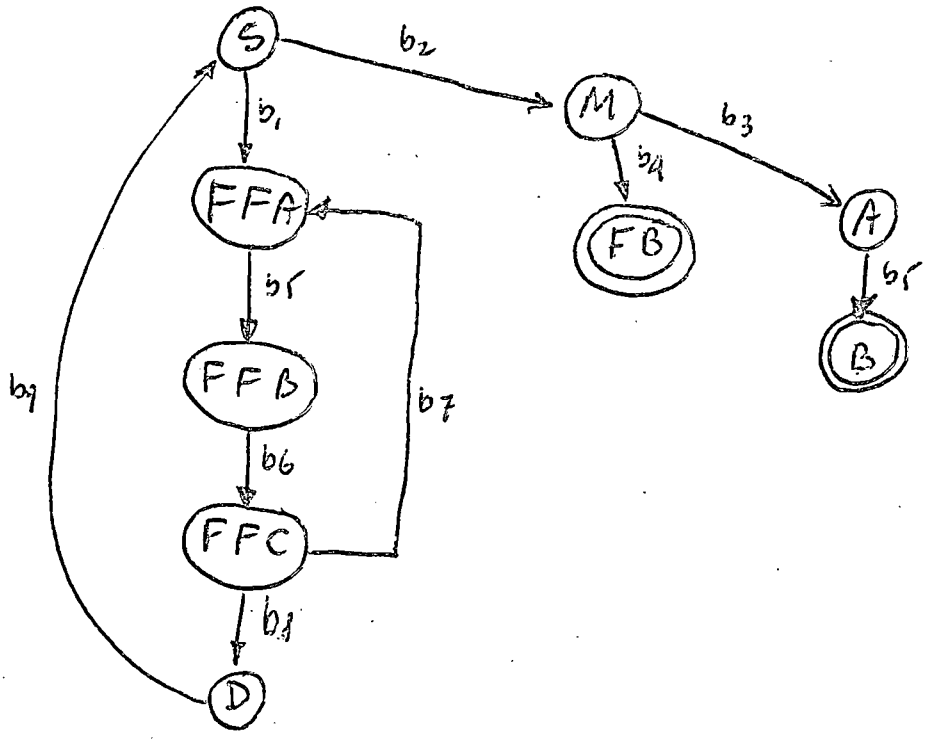


Figure 12 A Recoverable TM

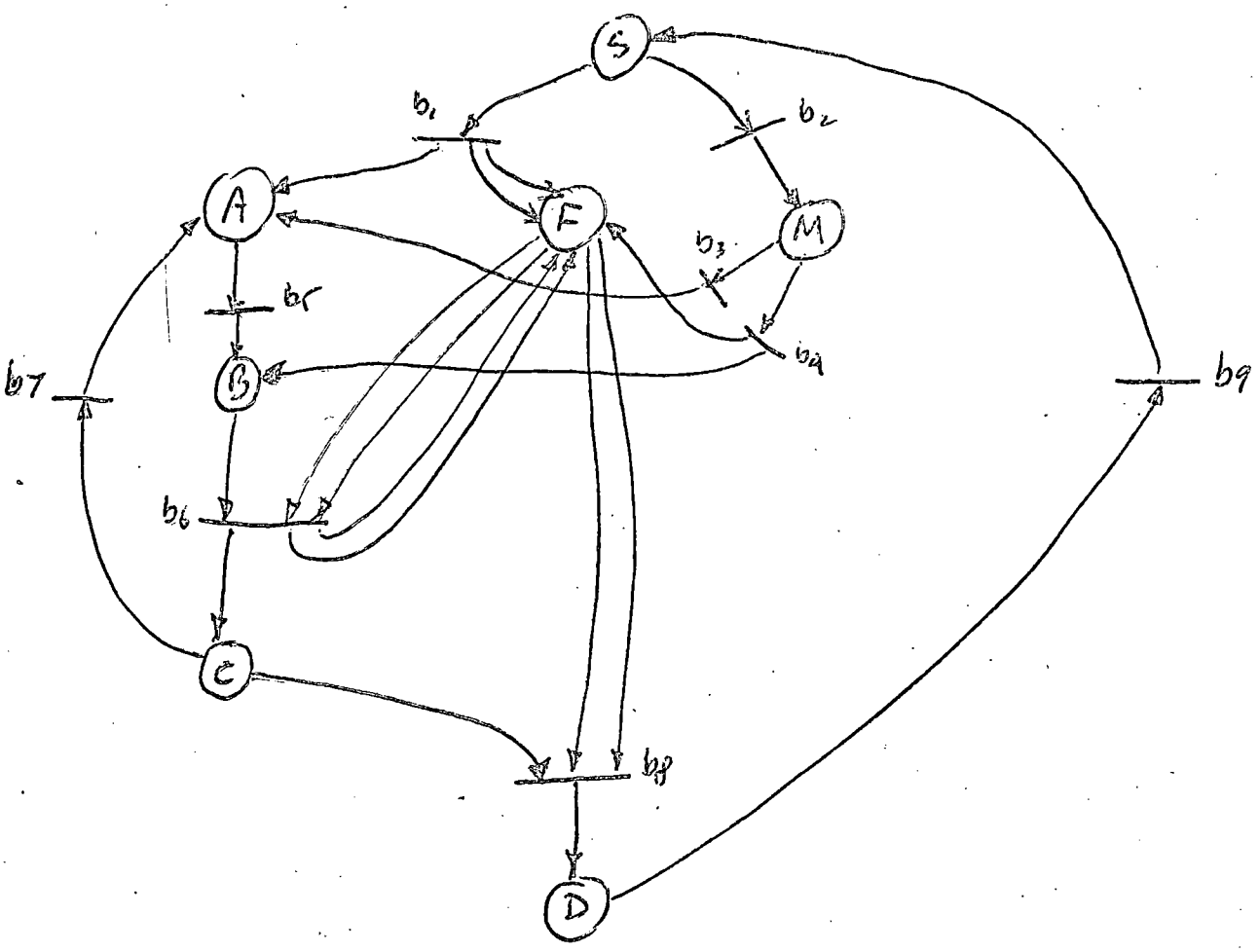


Figure 13 A PN for the TM of Figure 12

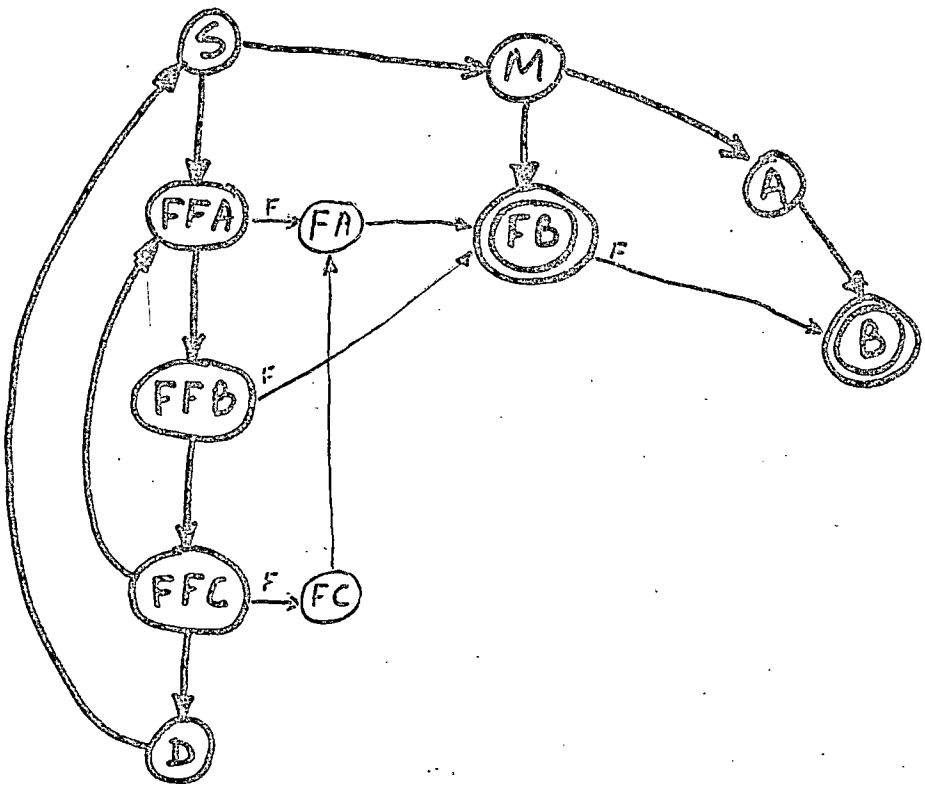


Figure 14 The ETM of the PN of Figure 13

Compare figures 10 and 12. The left loop in both TM's are equal. But, the transitions are t_k^3 or t_k^2 depending in the other states (states A_i) of the system.

References

[1] Merlin, P.M. Recoverability of Processes. Technical Report #44; Department of Information and Computer Science; University of California; Irvine; 92664; February 1974.