THE TIME-PETRI-NET AND THE RECOVERABILITY OF PROCESSES

Philip M. Merlin

Technical Report #48

May 1974

Department of Information and Computer Science
University of California, Irvine

This work was supported in part by the National Science Foundation under grant GJ1042 - "The Distributed Computer System"

ACKNOWLEDGEMENT

I would like to expess 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 occurrance of failures. The results of [1] were improved in [2]. [2] presented an exhaustive study of recoverability when a failure of type "loss of token" occurs. The general structure of a process, in order to be recoverable from that kind of failures, was given. [2] also shows a way of designing Petri-nets (PN), and specially, recoverable Petri-nets.

The processes studied in those papers 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.

The present paper, elaborates the results of [2], and analyzes the practical limitations of recoverable processes. Since in the Petri-net model these limitations are found to be practically unacceptable, a new model (the Time Petri net - TPN) is defined. This model is based in the Petri-net model, but include some knowledge about the execution times of the events. The PN model is found to be a particular case of the TPN model.

The present paper studies the property of recoverability of processes under the occurrance of a failure of type "loss of token" using the TPN model. Section 4 shows that for any given TM that can be implemented by a PN, a TPN can be designed so that it executes the given TM and is recoverable from a given failure of kind "loss of token". Several practical examples are explored.

The contents of this work is a natural continuation of [1] and [2]. This paper assumes that the reader is familiar with the concepts presented in [1] and [2]. The same definitions and notations are used.

2. PROPERTIES OF RECOVERABLE TM's

In [2] the structure of the TMs that can be implemented as recoverable processes (under loss of tokens) were studied. In this section, it is shown that these TMs 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 processes 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 that not belong to kind 1.

Note that, as in [2], we will deal only with processes that have finite TM.

2.1 PROPERTIES OF PROCESSES OF KIND 1.

The recoverable processes of this kind have the following properties:

2.1.1 THEOREM

If A1 + F is a legal state, then there exists a

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_k^1 (see [2]). In [2] (section 3.1.2.7) it is shown that, in this case, the process is recoverable only if A1 is a terminal state. Since we deal in this section with processes that have no terminal states then the transition t_k exist, and therefore the state A2 + F is legal.

2.1.2 THEOREM

There exists at least one loop, so that all the members in the loop include the condition F

PROOF :

From theorem 2.1.1, each state that includes F has a successor that also includes F. It means that there exists the transitions:

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

If there is not a loop, this means that 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.

2.1.3 THEOREM

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 2.1.2 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 in Q1 occur a failure the PN goes to a state, say S1, given by:

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

In [2] it was shown that if there exist 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 PM can only go to states that have less than k instances of F.

Since the TM is finite and recoverable, and we deal with TMs of "kind 1" after the failure there are not terminal states. This means that the states after the failure include loops of legal states. This loops do not include loops that all of its states belong to [Pi], because otherwise the definition of Q1 is contradicted. On the other hand, if there are not loops of elements of [Pi] then theorem 2.1.1 implies that the loops include only states that not include F. But, in [2] it was shown that if after a failure there exists a path of transitions then there also exist a correspondent path of legal transitions. Each state in this second path have one more instance of F than the correspondent state

in the first path. This means that exist 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 occurrance of the failure in state Q1.

Q.E.D.

2.2 PROPERTIES OF PROCESSES OF KIND 2

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. In this case, when the process is in A1 + F the external world will sense the same conditions as in A1. This means that the external world will assume that the process is ended in A1.

On the other side, if A1 + F is also a terminal state then exist two diferent cases:

1. F is not sensed by the external world.

In this case F is not necessary in the terminal

state,and there is not 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 stay 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.

2.3 PROPERTIES OF BOTH KIND OF PROCESSES

The following properties exist in all the 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 not a path from A1 to A2.

PROOF

Exist 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,..... ...,bm

In this case, b1 fires in A1 bringing the system to a

legal state, say S1. But since

$$IC(b1) < A1 < A2$$

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.

2.4 DISCUSSION

Theorem 2.3 shows that if A1 + F and A1 are legal states, then there is not a 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 states Ai and Ai + F, then all the recoverable processes have the property of irreversible degradation. The situation is unacceptable specially in the case of processes without terminal states. In this case,the process never terminates, but it degrades in the number of possible states in which it can stay. An example of this case of recoverable process is shown in figure 1. In this example, there are paths from the states FFA, FFB, and FFC to the states FA, FB, FC, A, B, and C, but not in the reverse direction. This process will never terminate. But, after a degradation it will never return to the states D, S, FFA, FFB, and FFC.

Theorem 2.1.3 shows that in recoverable processes of kind 1 there exists a loop of states always having the same number of instances of F. Section 2.2 describes the limitations of the processes of kind 2. The two kind 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.

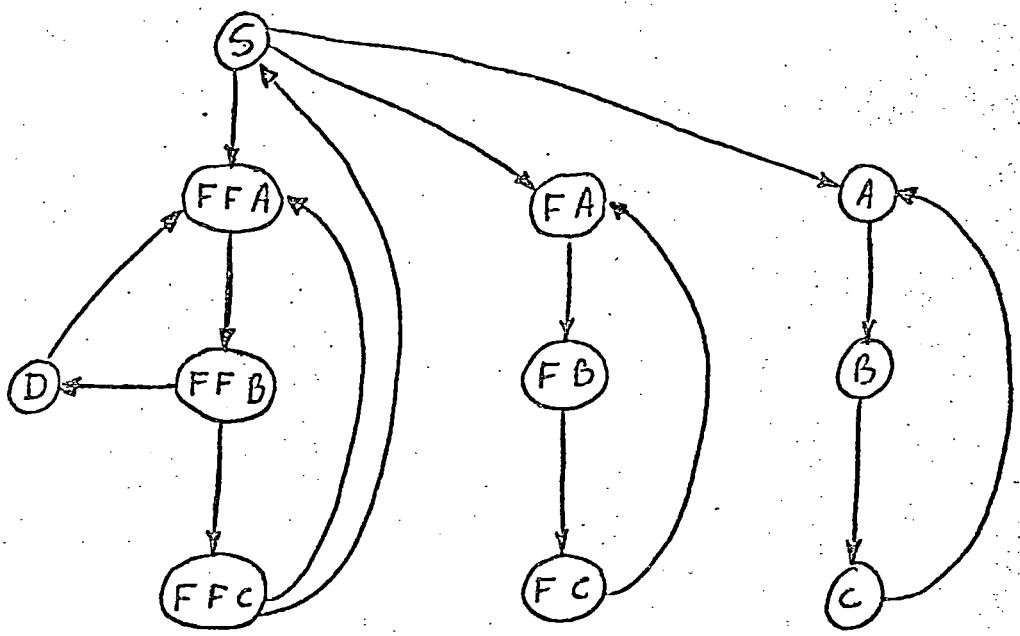


Figure 1: A Recoverable TM

From the analysis in this section, we can conclude that the recoverable processes represented by the model described in [1] and in [2] have a very constrained structure. This limitations to the structure are usually unacceptable in real systems.

But in the examples presented in [1] this limitations has been removed by postulating the existence of the function T (T is defined in [1]). Function T is ,in certain way, 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 following section, the concept of time is introduced into the Petri-net model of processes. Recoverability of processes is studied using this improved model.

3. THE TIME-PETRI-NET (TPN)

3.1 DEFINITIONS

3.1.1 a TPN is defined by a Petri-net (as defined in [2] section 2.2) in which for each bar bi is given a tuple [t*i ; t**i]. For all i there exists:

1. t*i; t**i real numbers
2. $t^*i \geq 0$; $t^{**}i \geq 0$
3. $t^*i < t^{**}i$

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

1. if the conditions IC(bi) holds for a period of time equal or greater than t*i then bi can fire (with the firing algorithm defined in [2]).
2. If the conditions IC(bi) hold for a period of time equal to t**i then bi fires.

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

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

3.1.5 $T^*_b(S)$ is defined as the minimal time that the TPN has to stay at state S so that bar b can fire.

3.1.6 $T^{**}_{b}(S)$ is defined as the maximal time that the system can stay in state S before b fires.

3.1.7 $T^{*}(S)$ is defined as the minimal time that the TPN will stay in state S when it arrive to this state.

3.1.8 $T^{**}(S)$ is defined as the maximal time that the TPN can stay in state S.

3.2 PROPERTIES OF THE TPN

3.2.1 A TPN is a PN if for all i:

$$t^{*}i = 0$$

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

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 ([2] section 2.2).

3.2.2 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 includes the conditions of the firing algorithm in the PN.

3.2.3 The opposite of 3.2.2 is not always true. Figure 2(a) shows an example of a PN. If A is a legal state then b1 or b2 can fire. Figure 2(b) shows a

TPN built on the previous PN. In this case, if A holds a token then b1 has to fire before 5. But b2 can fire only after 6. This means that in this case b1 will always fire before b2. In this case, b2 never fires.

3.2.4 Applying successibly the property 3.2.2, each sequence of legal states that exist in the TPN also exists in the corresponding PN.

3.2.5 Suppose that IC(b1) < Si and IC(b2) < Si, and Si is a legal state. In a PN, b1 or b2 can fire in Si. But if there exists:

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

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 2 shows this situation).

3.2.6 From definitions 3.1.1 and 3.1.5 there exists:

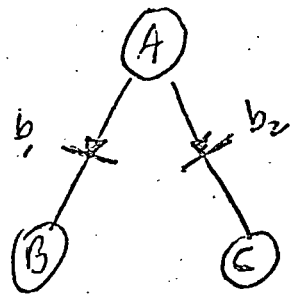
$$t^{*}_{bi}(Sj) \leq t^{*}bi$$

for any i and j.

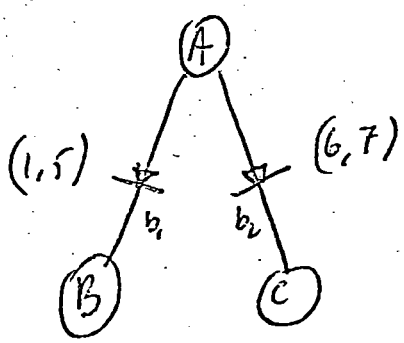
3.2.7 From definitions 3.1.1 and 3.1.6 there exists:

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

for any i and j.



(a) PN



(b) TPN

Figure 2: (a) a PN; (b) a TPN of the previous PN

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

$$IC(bi) < Sj$$

then t**(Sj) is given by:

$$t^{**}(Sj) = \min(t^{**}_{b1}(Sj);\ t^{**}_{b2}(Sj);\ .....t^{**}_{bp}(Sj)) \qquad (3.2.8.1)$$

because the first bar that arrive to its maximal waiting time (t**) has to fire.

Replacing 3.2.7 in 3.2.8.1:

$$t^{**}(Sj) \leq \min(t^{**}b1;\ t^{**}b2;\ .....t^{**}bp) \qquad (3.2.8.2)$$

(3.2.8.2) 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 (3.2.8.1), but in several practical cases the values $t^{**}_{bi}(Sj)$ are dificult to compute.

3.2.9 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:

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

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

$$B < Qi^j$$

Each Qi represents a possible sequence in the TPN. In Qi the bag B holds tokens, and there are not transitions in the sequence such that if bar bk fires in Qi^j then:

$$B \not< Qi^j - IC(bk) \qquad (3.2.9.1)$$

This means that in Qi, there is <u>not</u> a 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:

$$Qi \not< Qj$$

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

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

$$t^{**}B \leq \max(t^{**}(Q1^1)+t^{**}(Q1^2)+\ldots; t^{**}(Q2^1)+\ldots; t^{**}(Qp^1)+\ldots) \qquad (3.2.9.2)$$

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 not transitions that satisfy (3.2.9.1).

4. RECOVERABILITY OF TPM 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 [2] shows that the process is not recoverable if exist either:

1. loops of illegal states, or
2. terminal illegal states.

[2] shows that for each loop of illegal states there exists a correspondent loop of legal states that include the condition F. In [2], it is also shown that the designer can choose an implementation such that if there exists a legal transition:

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

and A1 is not a legal state then there does not exist the transition:

$$t_p = A1 \rightarrow A2$$

when the PN arrives to A1 after a failure. Therefore one of the transitions of the legal loop can be implemented so that

there is not a correspondent loop of illegal states.

Suppose that the given TM is implemented by a PN such that there are not loops of illegal states. 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 has to execute a transition from Ai to a legal state in the TM, say Si. This means that:

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

On the other hand, bi is not allowed to fire in any legal state. This means that bi does not affect the execution when there is not failure, so that the TM is normally executed. In order to disable the firing of bi during normal execution, t*i has to satisfy:

$$t^*i > t^{**}IC(bi)$$

Note that if there exists loops such that the states in the loop include IC(bi), then the implementation has to be such that t**IC(bi) is not infinite. The following example shows this situation.

4.1 EXAMPLE

Figure 3 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 4. Figure 5 shows the ETM corresponding to this implementation. The number in each arc denotes the bar that implements the corresponding transition. In this implementation there exists 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. 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

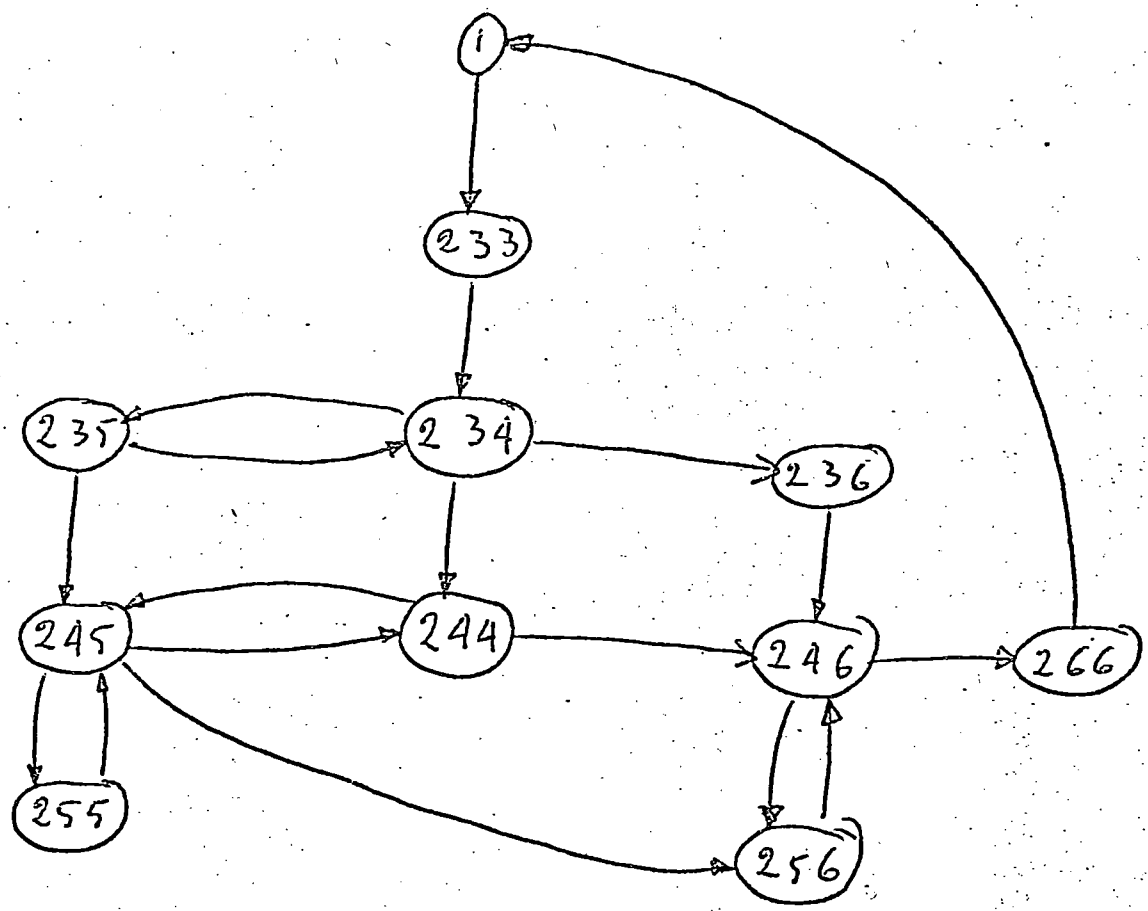


Figure 3: A Token Machine (TM)

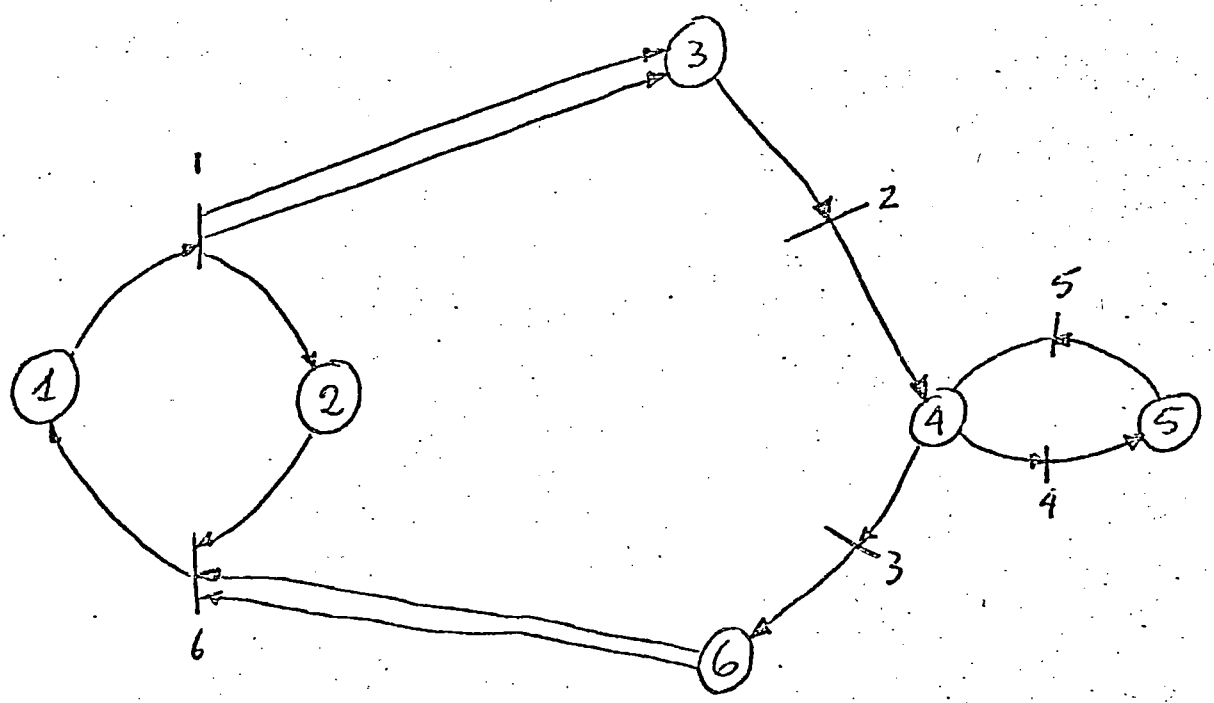


Figure 4: A PN for the TM of figure 3

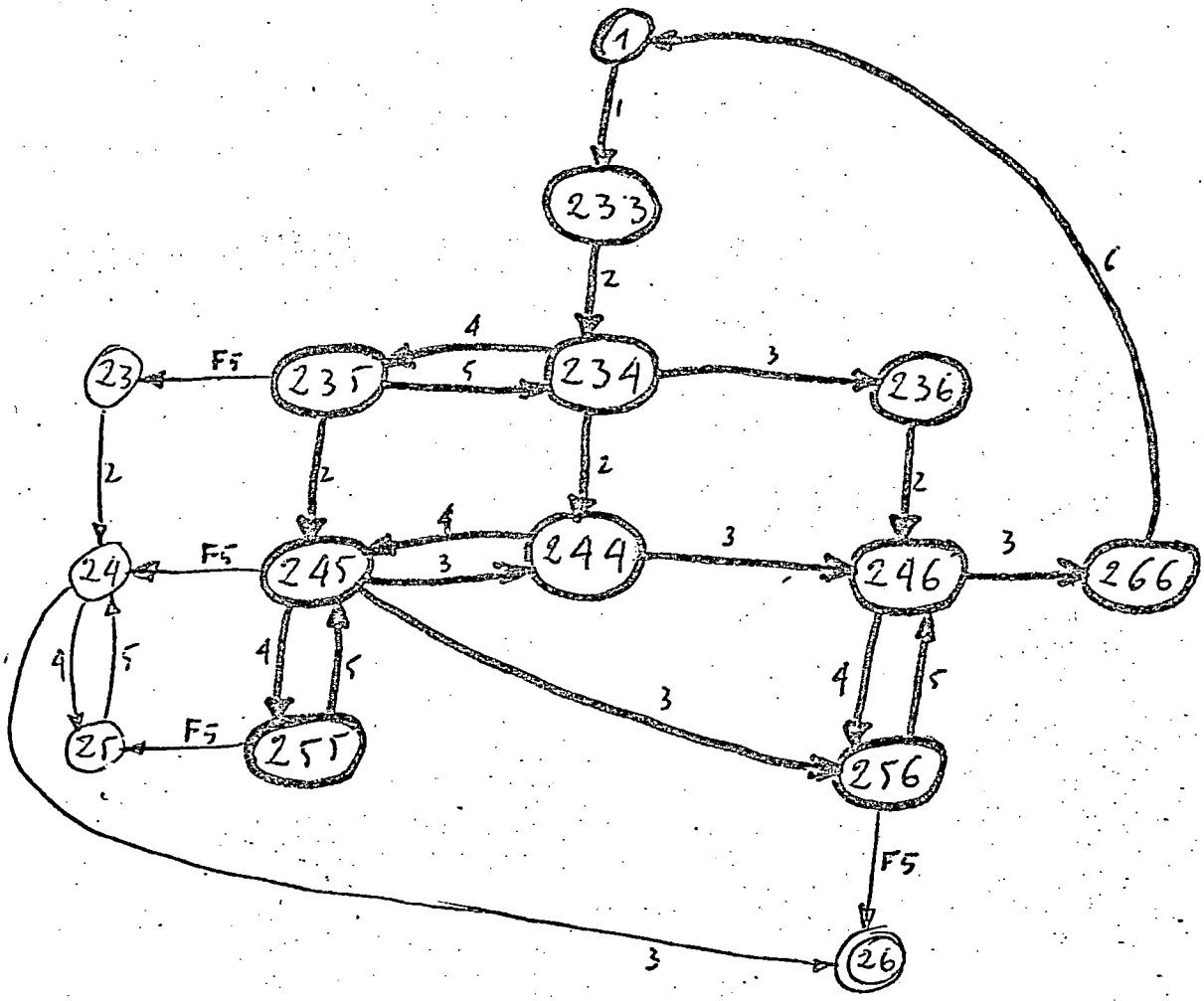


Figure 5: The ETM for the PN of figure 4

and 256 connected by the transitions executed by the bars 4 and 5.

But after the transition from 246 to 256 is executed by 4^4 instead of bar 4, also this problem is 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 shows the new implementation of the TM, after bar 4 was split into four different bars. Figure 7 shows the ETM for the PN of figure 6. 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 before, t*7 has to satisfy:

$$t*7 > t**26$$

The next step is to compute t**26, or at least an upper

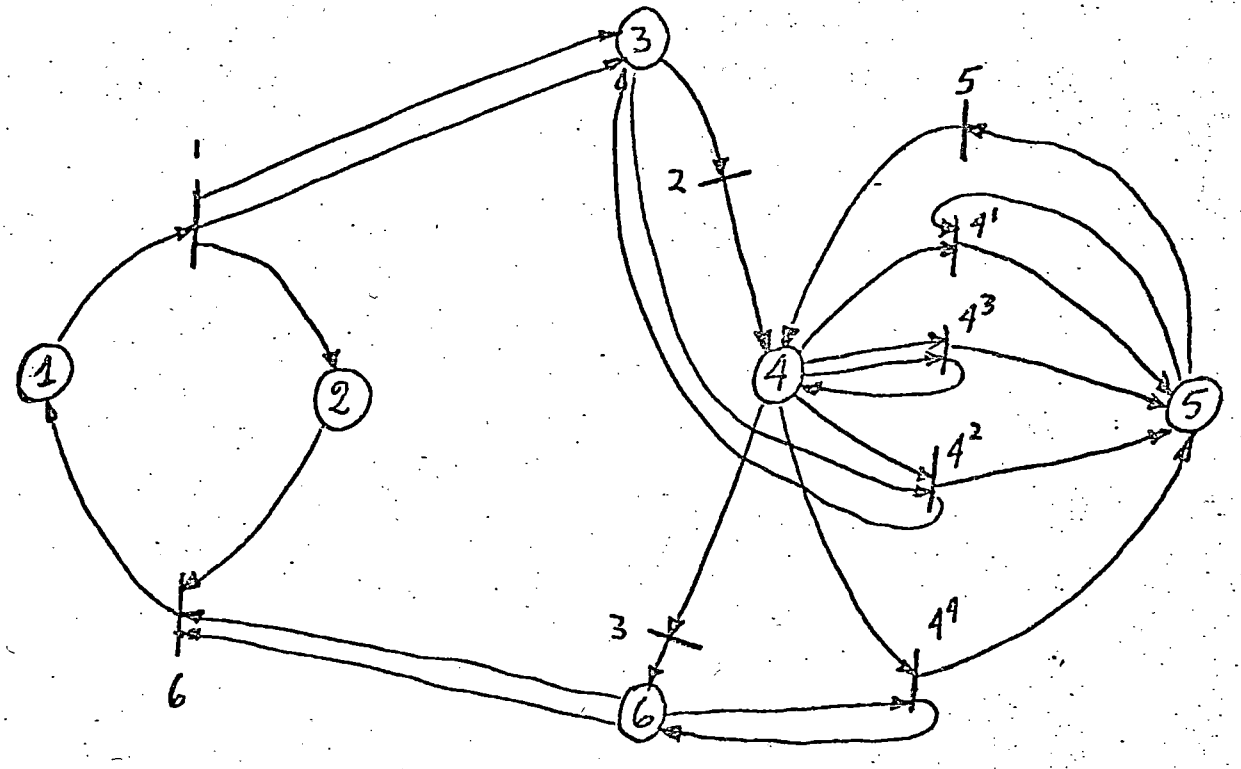


Figure 6: A PN for the TM of figure 3

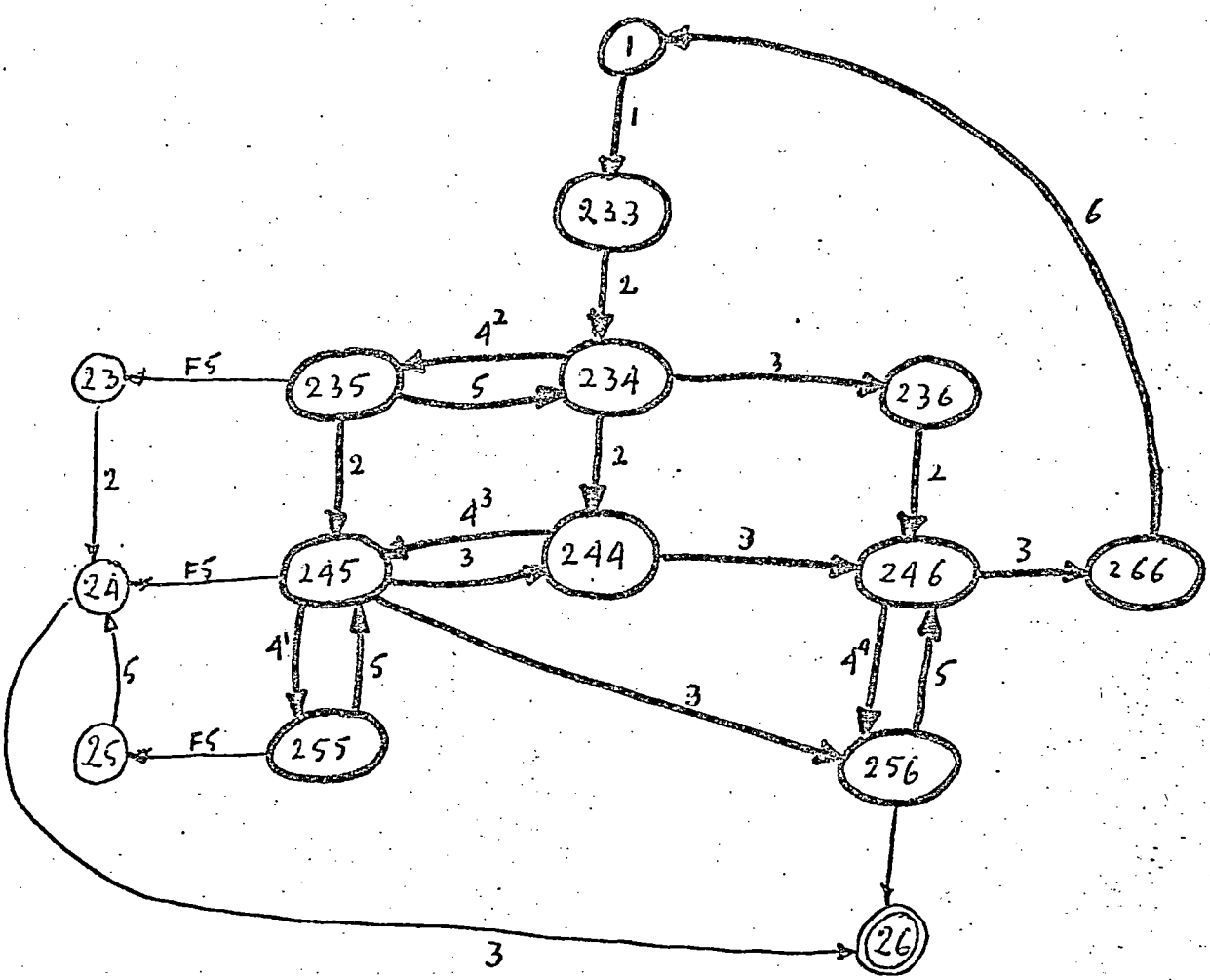


Figure 7: The ETM for the PN of figure 6

bound of t**26. In the next steps we will follow the procedure described in 3.2.9.

The possible sequences of states that include 26, and that satisfy the constraints explained in 3.2.9 are:

Q1 = [236; 246; 266]

Q2 = [256; 246; 266]

From 3.2.8:

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 (3.2.9.2) there exists that:

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

Thus, if:

$$t^{*}7 > \max(t^{**}2+\min(t^{**}3;t^{**}4^4)+t^{**}6 \ ; \ t^{**}5+\min(t^{**}3;t^{**}4^4)+t^{**}6) \qquad (4.1.1)$$

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 8 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 (4.1.1) is satisfied. The TPN of figure 8 implements the TM of figure 3 and it is recoverable in case of a loss of token in condition 5. After a failure, the system will arrive to state 26. After the process is in state 26 for a time equal to t*7, then bar 7 will fire and the TPN will return to legal state 1.

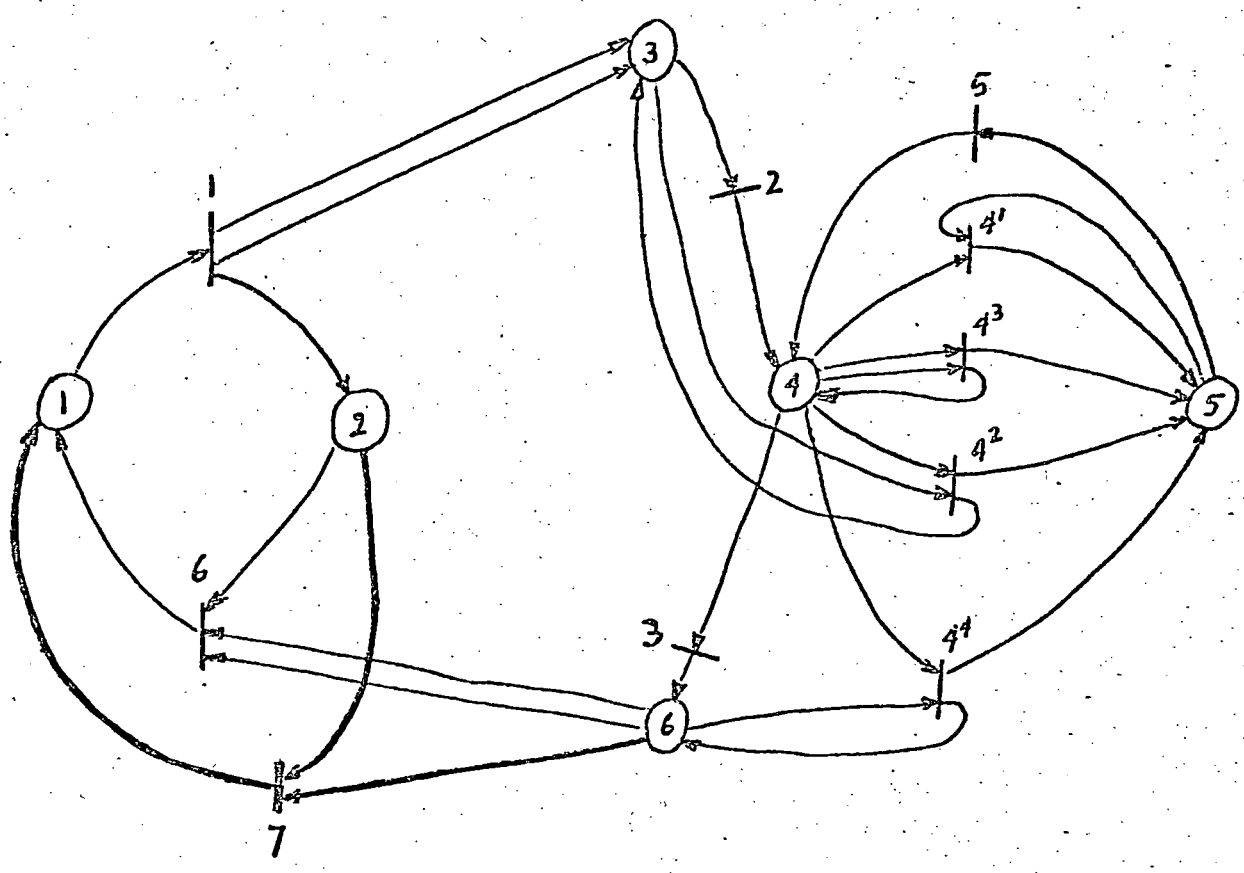


Figure 8: A recoverable TPN for the TM of figure 3

5. RECOVERABILITY OF A COMMUNICATION PROTOCOL

The study of the communication protocols in this paper is motivated by practical reasons. During the last 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 protocols processes is of considerable importance.

The presentation in this section is based, in part, on the study presented in [4] and on the examples given in [1]. The new model, the TPN, is used. The examples presented here are a simplified model of the IMP-IMP protocols used in the ARPANET. The study of these protocols are presented by the two following examples.

5.1 EXAMPLE 1

In this section the protocol of figure 9 is studied. This protocol is presented in [1]. We suppose that a possible failure is the loss of the message M. This means that a token in M can disappear. The dotted line from E to A represent the preparation of a new message by the sender. The dotted line from D to B represents the receiving process.

In order to simplify the example we suppose that the

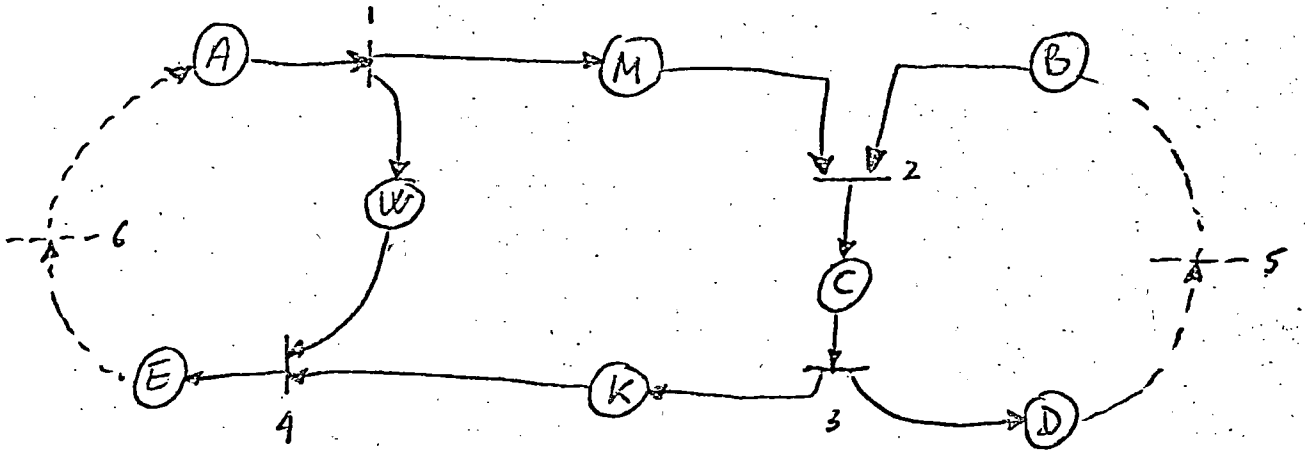


Figure 9: A PN of a protocol process

dotted line between E and A is activated before the line between D and B. This means that the receiver is ready to receive before the sender is ready to send. This assumption only simplifies the explanations and it does not reduce the generality of the example.

The ETM of the PN of figure 9 is given in figure 10. The ETM shows that exist only one illegal state, WB, and this state is also terminal. Section 4 shows that in order to transform such a process to recoverable, 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 in 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 in the state of the sender. In case of a failure, the sender will send again a transmition of the lost message. This means that:

$$IC(7) = W$$

and:

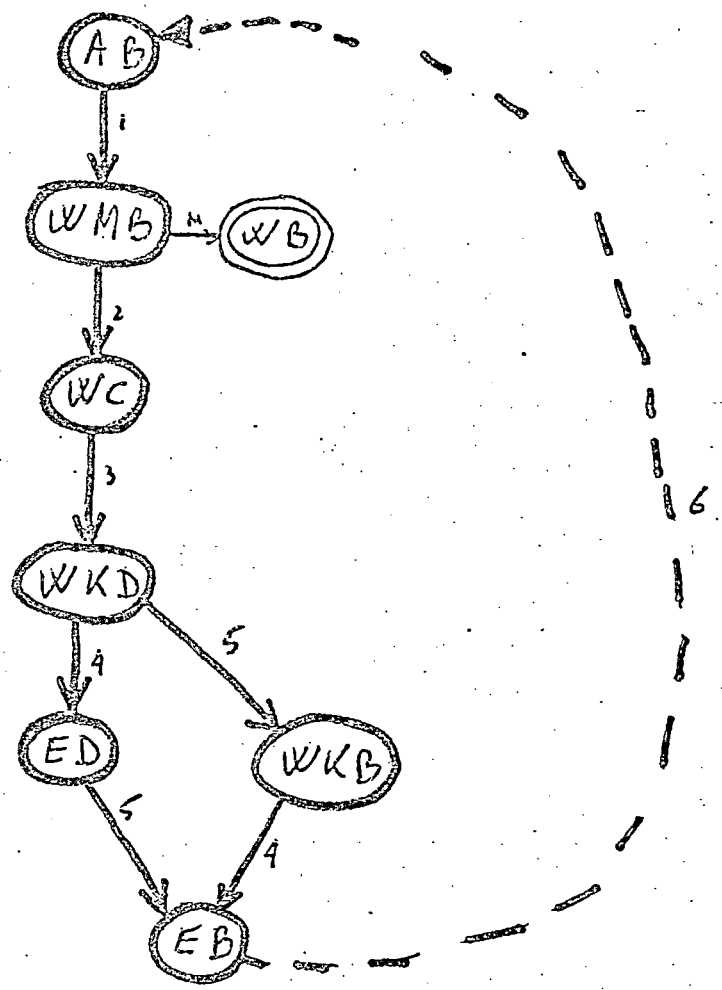


Figure 10: ETM for the PN of figure 9

$$OC(7) = MW$$

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

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

$$t*7 > t**W$$

From figures 9 and 10 it is possible to show that:

$$t**W = t**2 + t**3 + t**4$$

Thus, t*7 has to satisfy:

$$t*7 > t**2 + t**3 + t**4 \qquad (5.1.1)$$

The recoverable TPN and its corresponding ETM are shown in figures 11 and 12 respectively. This TPN is recoverable from failures of type "loss of token" in F. Note that if (5.1.1) is not satisfied then the ETM is infinite and the process is <u>not</u> recoverable. In many practical systems the t*7, that satisfies (5.1.1), can be very large. In these cases, the protocol of the next example can be used.

5.2 <u>EXAMPLE 2</u>

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

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

In the PN that represents this protocol there exist

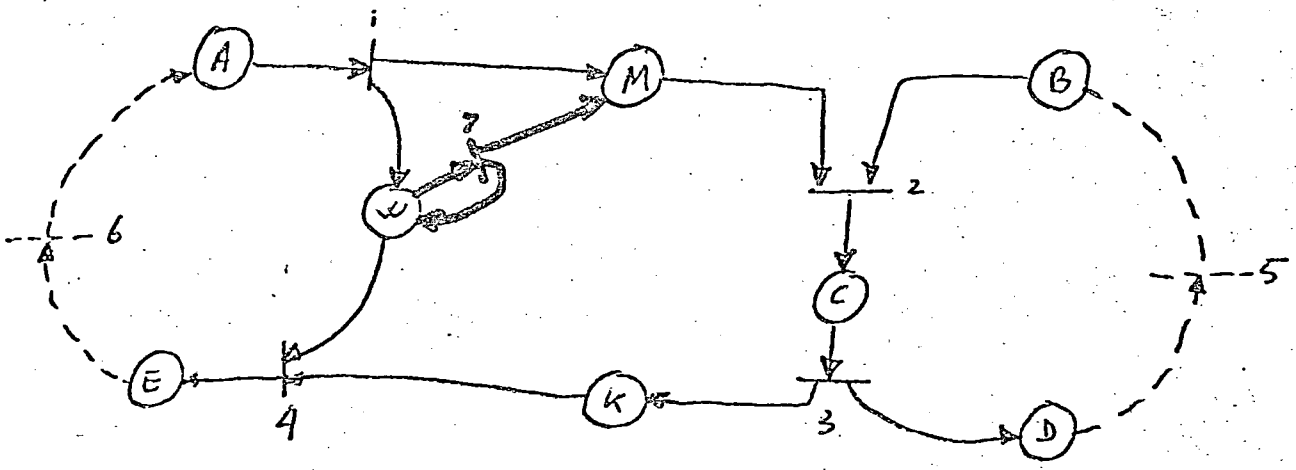


Figure 11: Recoverable TPN for the TM of figure 9

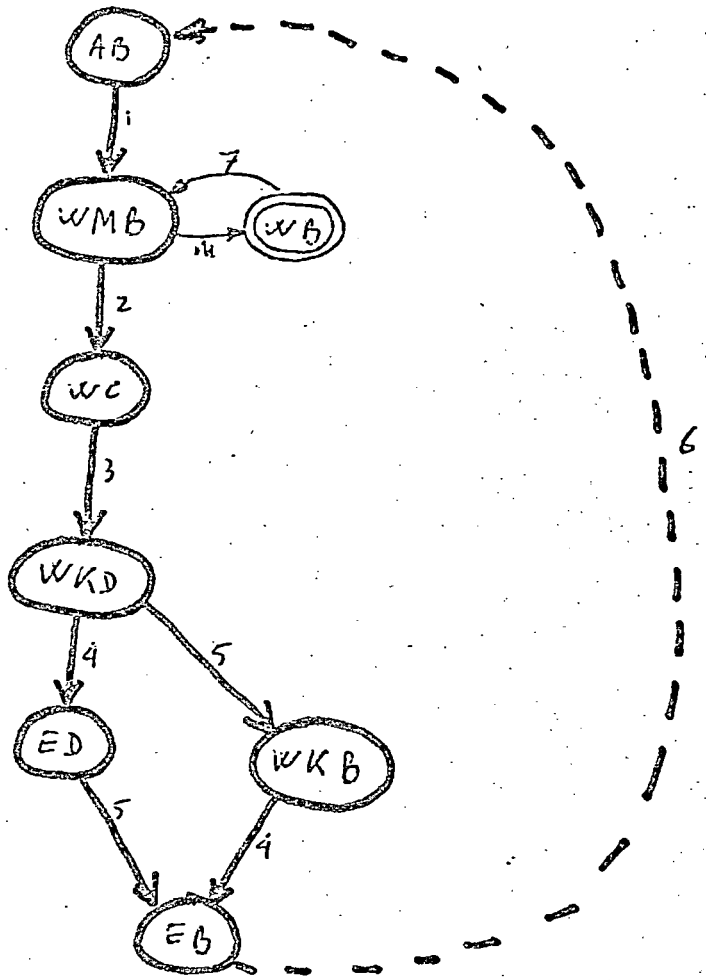


Figure 12: ETM for the TPN of figure 11

different conditions Mi ; (i = 1,2,...n). Each Mi correspond to the message carrying the sequence number i.

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

Ai = ready to send message i

Bi = ready to receive message i

Ki = acknowledge to message i is sended

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

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

For simplicity, in the present example, we assume that n=2 (the same approach is applyable in the general case). Figure 13 shows the PN for this case. This PN is similar to two instances of the PN shown in figure 9. The only difference is in the dotted lines. The dotted lines represent the sender and the receiver processes. In this case, these processes are responsible of the correct sequencing of the messages.

Figure 14 shows the corresponding ETM for the case that a failure can occur in M1 or M2, assuming initial state A1B1. Also here, we suppose that the receiver is ready to

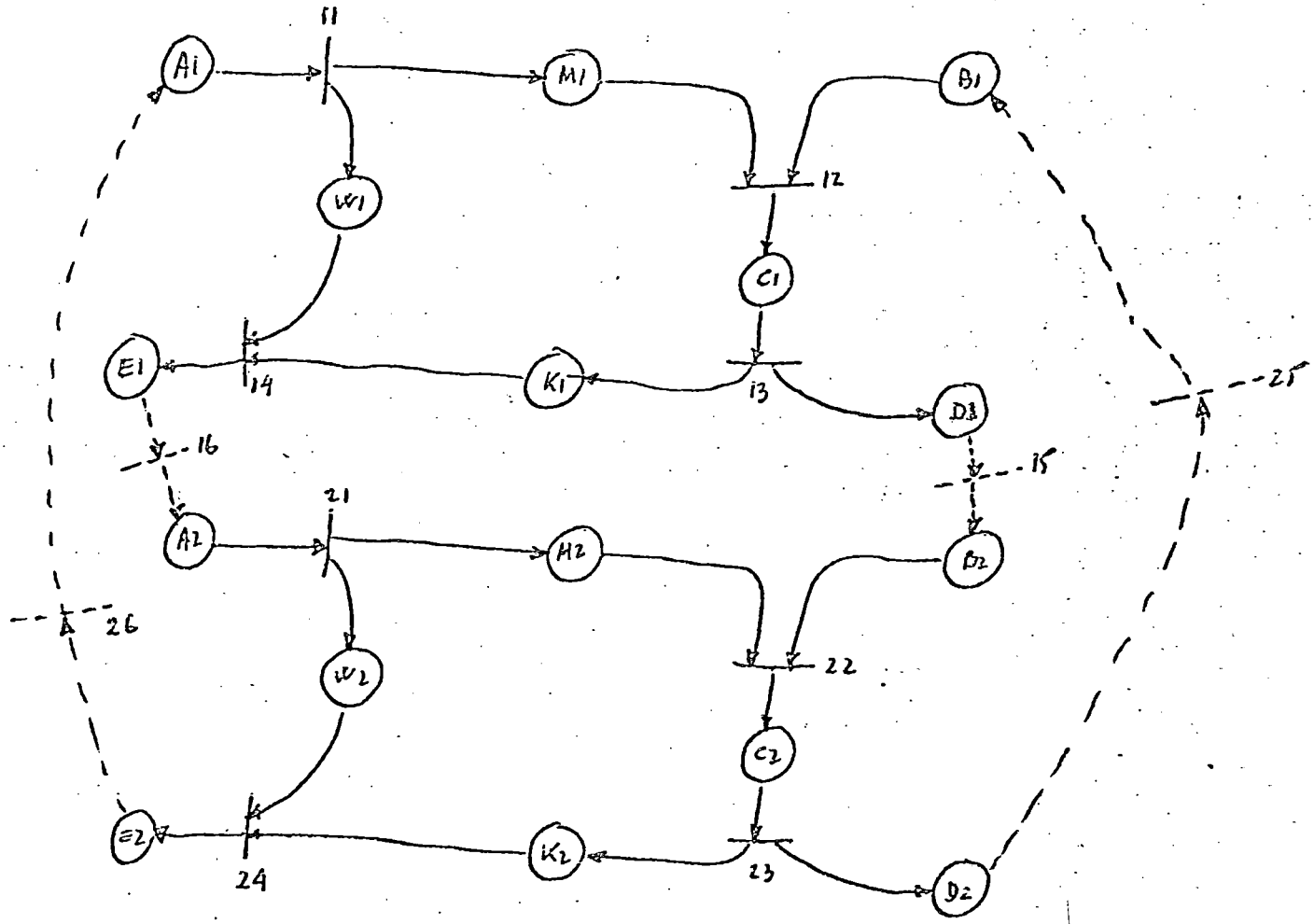


Figure 13: PN of a protocol process

receive before the sender is ready to send. This assuption again simplifies the analysis of this example, but it does not reduce its generality.

The ETM of figure 14 is similar to two instances of the ETM shown in figure 10. In order to convert the PN of figure 13 to recoverable, the approach is 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 exists:

1. IC(17) = W1
2. OC(17) = W1M1
3. T*17 > t**12 + t**13 + t**14 (5.2.1)
4. IC(27) = W2
5. OC(27) = W2M2
6. T*27 > t**22 + t**23 + t**24 (5.2.2)

This TPN is shown in figure 15 and it is recoverable.

But, what happens if (5.2.1) or (5.2.2) are not satisfied?. In this case, bar 17 or bar 27 can fire before it is sure that the TPN is in an illegal state. This means that the bars 17 or 27 can fire also in legal states. In order to simplify the following explanations, for the case that (5.2.1) or (5.2.2) are not satisfied we assume that:

$$t^{**}12 + t^{**}13 + t^{**}14 > t^{*}17 > t^{**}12 + t^{**}13 \qquad (5.2.3)$$

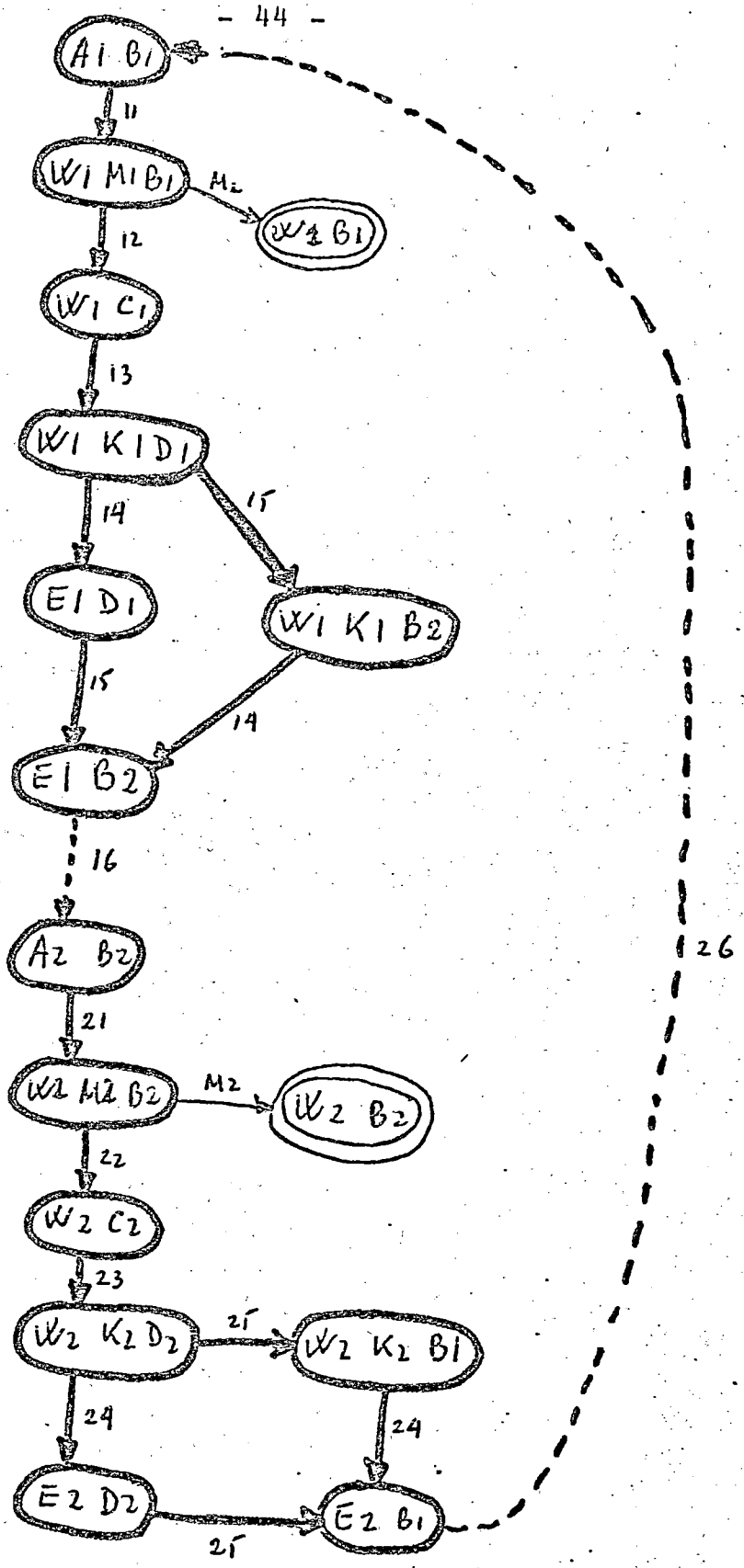


Figure 14: ETM for the PN of figure 13

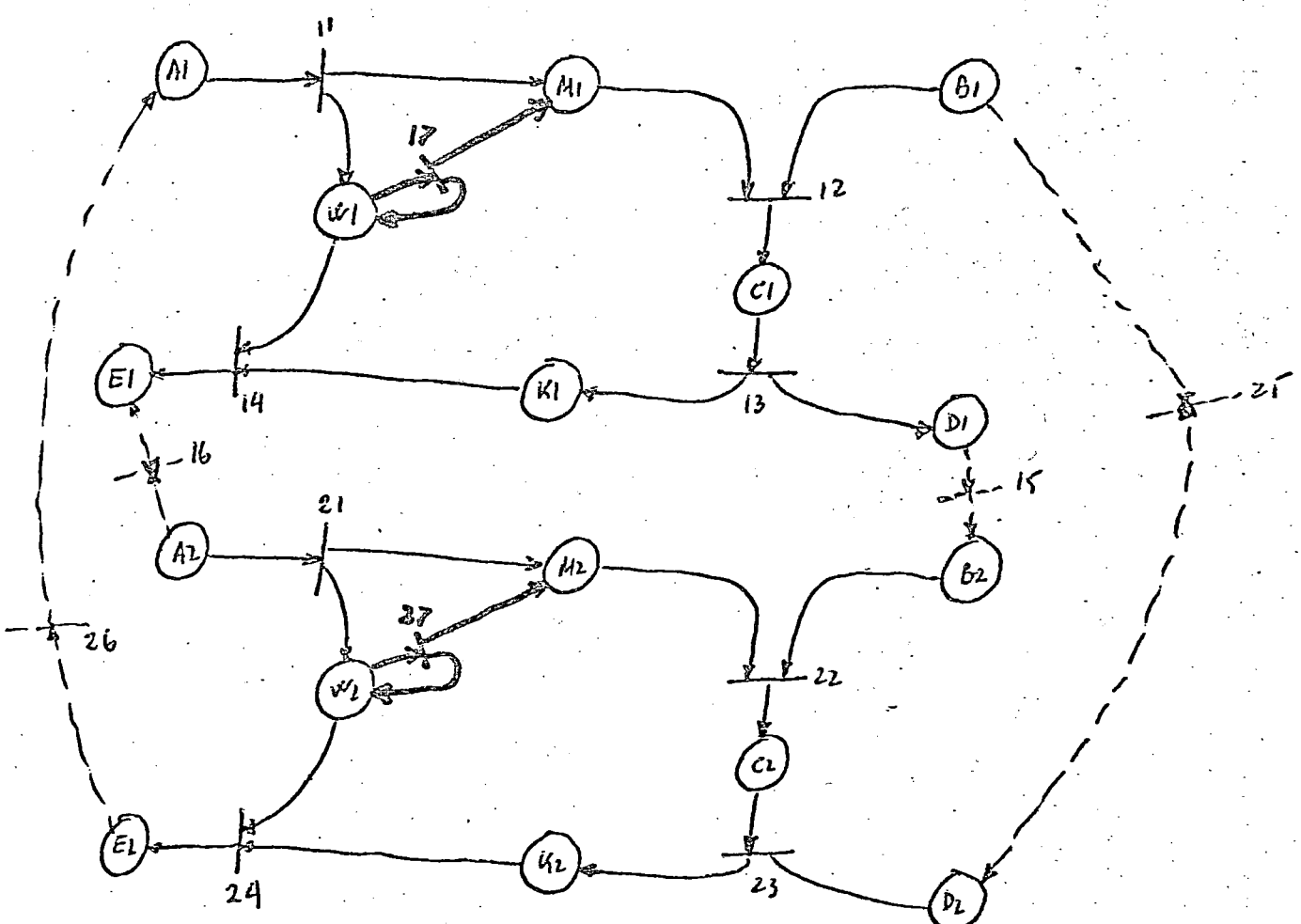


Figure 15: Recoverable TPN for the PN of figure 13

$$t^{**}22 + t^{**}23 + t^{**}24 > t^{*}27 > t^{**}22 + t^{**}23 \qquad (5.2.4)$$

The same approach is applicable for the general case in which (5.2.1) and (5.2.2) are not satisfied.

The TM for the TPN described in figure 15, for the case that (5.2.3) and (5.2.4) are satisfied, it is shown in figure 16. 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 14. 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 occurrance 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 occurrance of a pseudo failure, the execution will always return to the legal states. The solution of this problem is

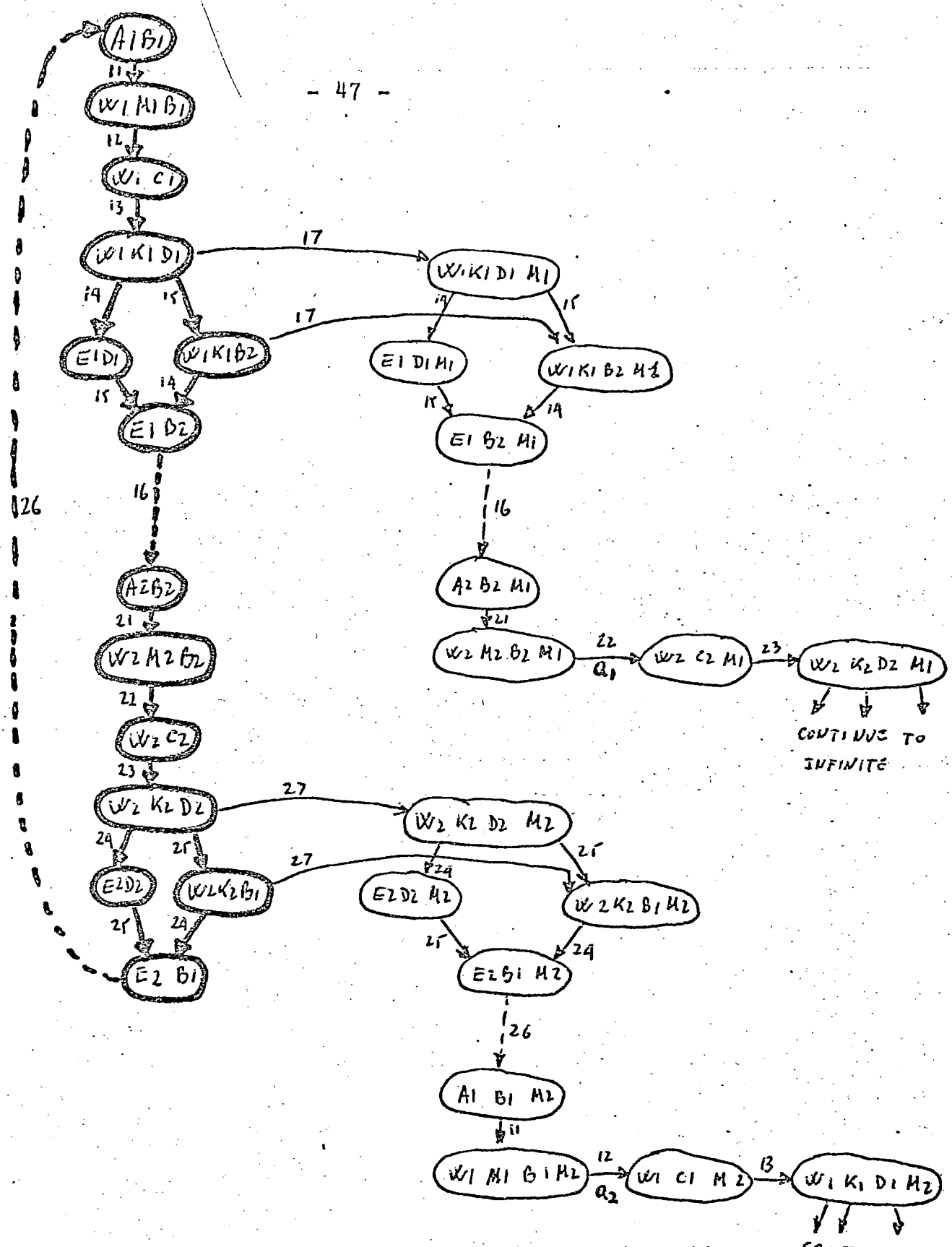


Figure 16: TM for the case that (5.2.3) and (5.2.4) are satisfied

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 similar way as was done in [2] for the case of "loss of token". But, several particular cases can be easily solved without such an analysis.

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

Suppose that a "cut-set" of pseudo illegal arcs is chosen in the TM of figure 16. Since the cut-set include only pseudo illegal states it divides the TM into two parts:

1. part 1 includes all the legal states and part of the pseudo illegal states,
2. part 2 includes only all the pseudo illegal states that are not included in part 1.

In our example the cut-set of arcs [a1 , a2] in figure 16 is chosen.

If bars are added so 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 occurrance of a pseudo failure. If the conditions above are satisfied, after the occurrance 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 15, such that:

IC(18) = B1M2

OC(18) = B1

IC(28) = B2M1

OC(28) = B2

The new TPN is shown in figure 17 and the correspondent TM in figure 18. Figure 18 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 3.2.5:

$$t^{**}_{18}(W1M1B1M2) < t^{*}_{12}(W1M1B1M2) \qquad (5.2.5)$$

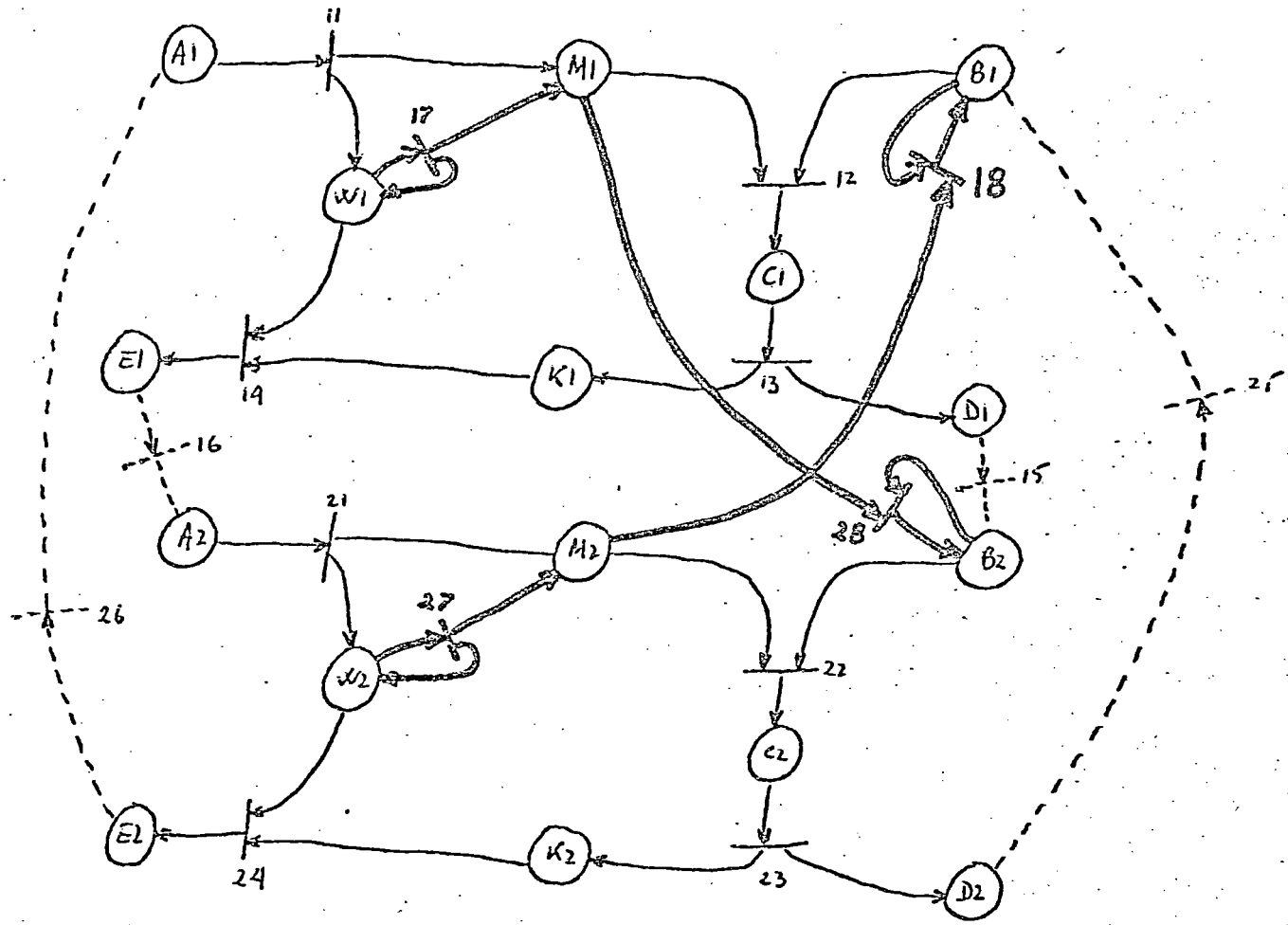


Figure 17: A new TPN

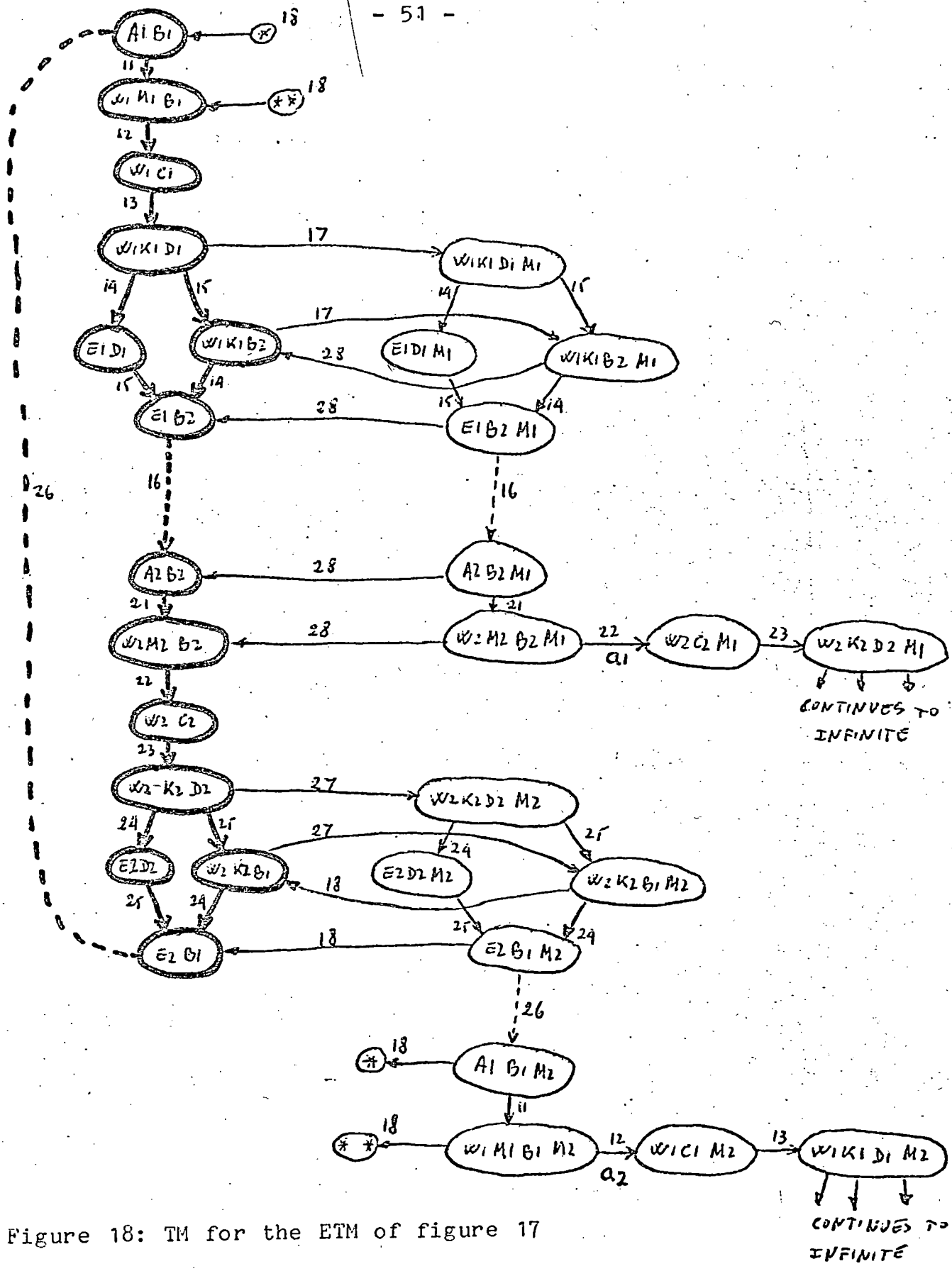


Figure 18: TM for the ETM of figure 17

$$t^{**}_{28}(W2M2B2M1) < t^{*}_{22}(W2M2B2M1) \qquad (5.2.6)$$

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

$$t^{*}_{12}(W1M1B1M2) = t^{*}12 \qquad (5.2.7)$$

Note that in this case t*12 is the minimal time that can elapse between a token is being placed in 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:

$$t^{*}_{22}(W2M2B2M1) = t^{*}22 \qquad (5.2.8)$$

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

From figure 17 and 18 it is possible to show that:

$$t^{**}_{18}(W1M1B1M2) = t^{**}18 - t^{*}11 - t^{*}26 \qquad (5.2.9)$$

and $$t^{**}_{28}(W2M2B2M1) = t^{**}28 - t^{*}21 - t^{*}16 \qquad (5.2.10)$$

Replacing (5.2.7) and (5.2.9) in (5.2.5) the result is:

$$t^{**}18 - t^{*}11 - t^{*}26 < t^{*}12 \qquad (5.2.11)$$

and replacing (5.2.8) and (5.2.10) in (5.2.6) the result is:

$$t^{**}28 - t^{*}21 - t^{*}16 < t^{*}22 \qquad (5.2.12)$$

The TM of the TPN of figure 17, with the constraints given by (5.2.3), (5.2.4), (5.2.11), and (5.2.12) is shown

in figure 19. 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 16. 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 can be also considered as legal states. Thus, all the states of figure 19 can be considered legal. These two ways of interpretation are equally convenient.

The TPN of figure 17, with the constraints (5.2.3), (5.2.4), (5.2.11) and (5.2.12) was designed so that it is recoverable under failures of kind "loss of tokens" in M1 or M2. The ETM of figure 20 shows this property.

The process, as given by the TM of figure 19 or the TPN of figure 17 (and the constrains in the execution times), has interesting properties, as following:

1. The messages are received in the same order that they are sent. This property is shown directly from figure 19. 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 precessor of W2M2B2M1 is A2B2M1), and in this case M1 is received first (the only successor of W2M2B2M1 is W2M2B2). In the

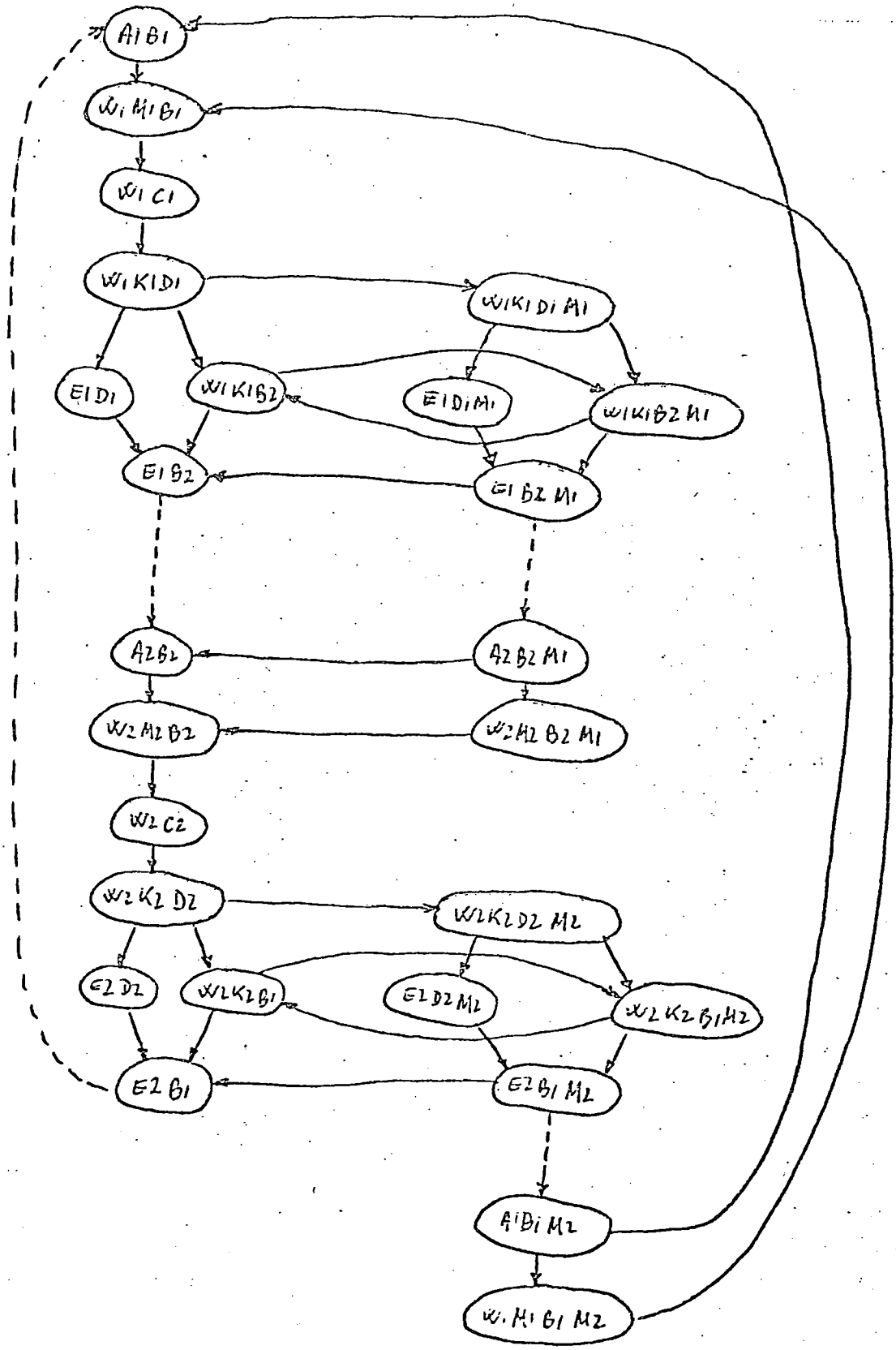


Figure 19: TM for the TPN of figure 17

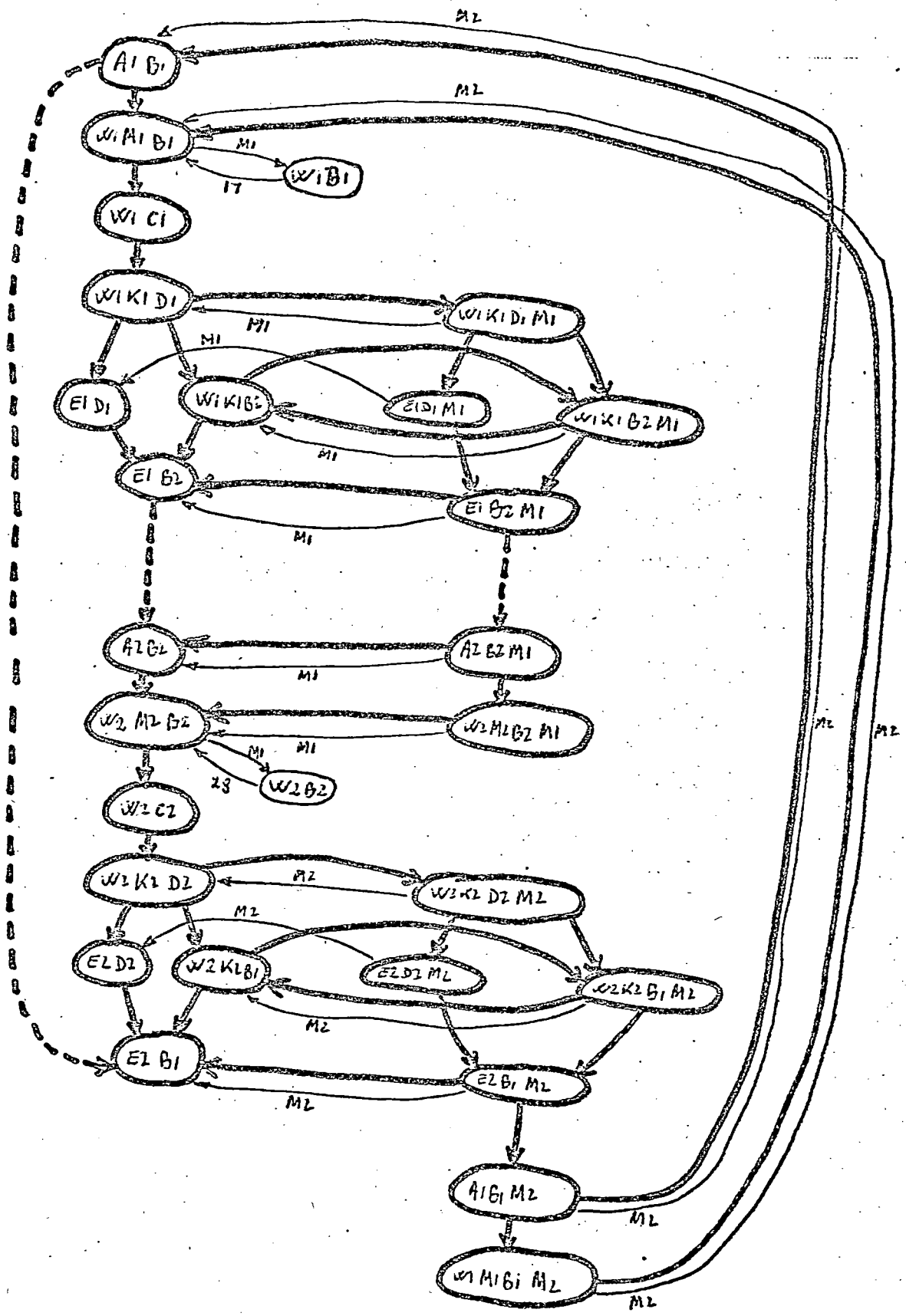


Figure 20: ETM for the TPN of figure 17

same way, when the process is in W1M1B1M2 the message M2 was sent first and it will be received first. This limitation in the order of the messages can be removed, in part, if the sequence number of each message is chosen from more than two possibilities [4].

2. Unequalities (5.2.11) and (5.2.12) can be rewritten as:

$$t^{**}18 < t^{*}12 + t^{*}11 + t^{*}26 \qquad (5.2.11a)$$

$$t^{**}28 < t^{*}22 + t^{*}21 + t^{*}16 \qquad (5.2.12a)$$

(5.2.11a) shows that the maximal time that 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 17). The same relation exist in (5.2.12a).

3. As shown before, $t^{*}12$ represents the minimal propagation time of the message M1. In a certain way, $t^{**}18$ represent 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 (5.2.11a):

$$t^{**}18 - t^{*}12 < t^{*}11 + t^{*}26 \qquad (5.2.11b)$$

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 (5.2.12a).

6. CONCLUSIONS

6.1 SUMMARY

[1], [2] and this paper are very closely connected each to each other. Since these three papers form an almost indivisible unit this conclusions are related to the entire set, and not only to this paper.

In these papers the problem of recoverability of processes have been modeled and formally defined using elements of the Petri-net. 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 failures was given. This way of designing 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 that are 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 limitated 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, a TPN can be designed so that it executes the given TM and it is recoverable from a given failure of type "loss of token".

But, in this recoverable TPN it is necessary to accept constraints in 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 these papers for the study of failures of type "loss of tokens" can be applied in order to explore other types of failures.

Other authors ([3] in section 7.2) 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 set of papers 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.

6.2 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 occurrance of other kind of failures. Among these, "generation of illegal tokens",etc.
2. The further research of the TPN model.
3. The formal analysis of other properties of processes,such as:
   (a) "fail-soft",
   (b) "fail-tolerant",
   (c) "best-effort"
4. The research of the transfer of failures among processes in an hierarchical structure.

REFERENCES

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

[2] Merlin, P.M. A Study on Recoverability of Processes. Technical Report #47; Department of Information and Computer Science; University of California, Irvine, 92664; April 1974.

[3] Larson, K.C. Computation Graphs; Department of Information and Computer Science; University of California, Irvine, California, 92664; 1974.

[4] Postel, J.B. A Graph Model Analysis of Computer Communications Protocols; Computer Science Department; UCLA; Los Angeles, California, 1974.