ELSEVIER

PII: S0951-8320(97)00030-6

Reliability Engineering and System Safety **57** (1997) 129–142

0951-8320/97/$17.00

The application of Petri nets to failure analysis

T. S. Liu & S. B. Chiou

Department of Mechanical Engineering, National Chiao Tung University, Hsinchu 30010, Taiwan, ROC

(Received 18 March 1996; accepted 23 February 1997)

Unlike the technique of fault tree analysis that has been widely applied to system failure analysis in reliability engineering, this study presents a Petri net approach to failure analysis. It is essentially a graphical method for describing relations between conditions and events. The use of Petri nets in failure analysis enables to replace logic gate functions in fault trees, efficiently obtain minimal cut sets, and absorb models. It is demonstrated that for failure analysis Petri nets are more efficient than fault trees. In addition, this study devises an alternative; namely, a trapezoidal graph method in order to account for failure scenarios. Examples validate this novel method in dealing with failure analysis. © 1997 Elsevier Science Limited.

1 INTRODUCTION

A variety of methods for failure analysis have been presented, namely, reliability block diagram, failure mode and effects analysis, and fault tree analysis. This study presents a failure analysis method based on Petri net models. The Petri net theory is a graphical method using some basic symbols for describing relations between conditions and events. It can represent and analyze the dynamic behaviour of systems. So far, the Petri net model has been used in a variety of applications, e.g., manufacturing systems, computer software and hardware systems.

Petri nets have two types of nodes named place P and transition T. These nodes are connected by arcs A, i.e., arcs connect transitions to places or places to transitions. Each place may contain one or more tokens. The basic symbols are defined as follows:

- ○: Place, drawn as a circle.
- ——: Transition, drawn as a bar.
- ↑: Arc, drawn as an arrow, between places and transitions.
- •: Token, drawn as a dot, contained in places.

Moreover, Peterson [1] defined P, T, and A as:

$$P = \{P_i \mid P_i \text{ is a place}, 1 \le i \le I\},$$

where I is a positive integer

$$T = \{T_i \mid T_i \text{ is a transition}, 1 \le i \le I\}$$

$$A \subseteq (P \times T) \cup (T \times P).$$

The state in Petri nets is represented by a token number m_i, called a marking, contained in each place P_i. As shown in Fig. 1(a), there is a token in each of input places P_1 and P_2 but no token in output place P_3. Accordingly, the Petri net marking is $M = (1,1,0)$. Each transition connects to both input and output places. A transition is said to be enabled when its every input place contains at least a token. A transition fires if it is enabled, which removes a token from all of its input places and puts a token in all of its output places. This results in a change of the marking, i.e., a change of the state.

The transitions in Fig. 1(a)–(c) are enabled, since in every case both places P_1 and P_2 contain at least one token. A firing of transition T_1 moves a token from each of P_1 and P_2 to the output place P_3 (and P_4 in case (c)). Place P_1 in Fig. 1(b) originally has two tokens. After firing, a token remains in P_1, since any firing can withdraw only one token from each input place. In spite of transitions T_1 and T_2 shown in Fig. 1(d), only T_1 fires due to its input place P_1, with a token. The firing of T_1 has nothing to do with input place P_2, whereas it gives a token to each of P_3 and P_4. In Fig. 1(e), by contrast, T_1 is not enabled due to no token in P_2.

The Petri net marking before firing ($M0$) and after

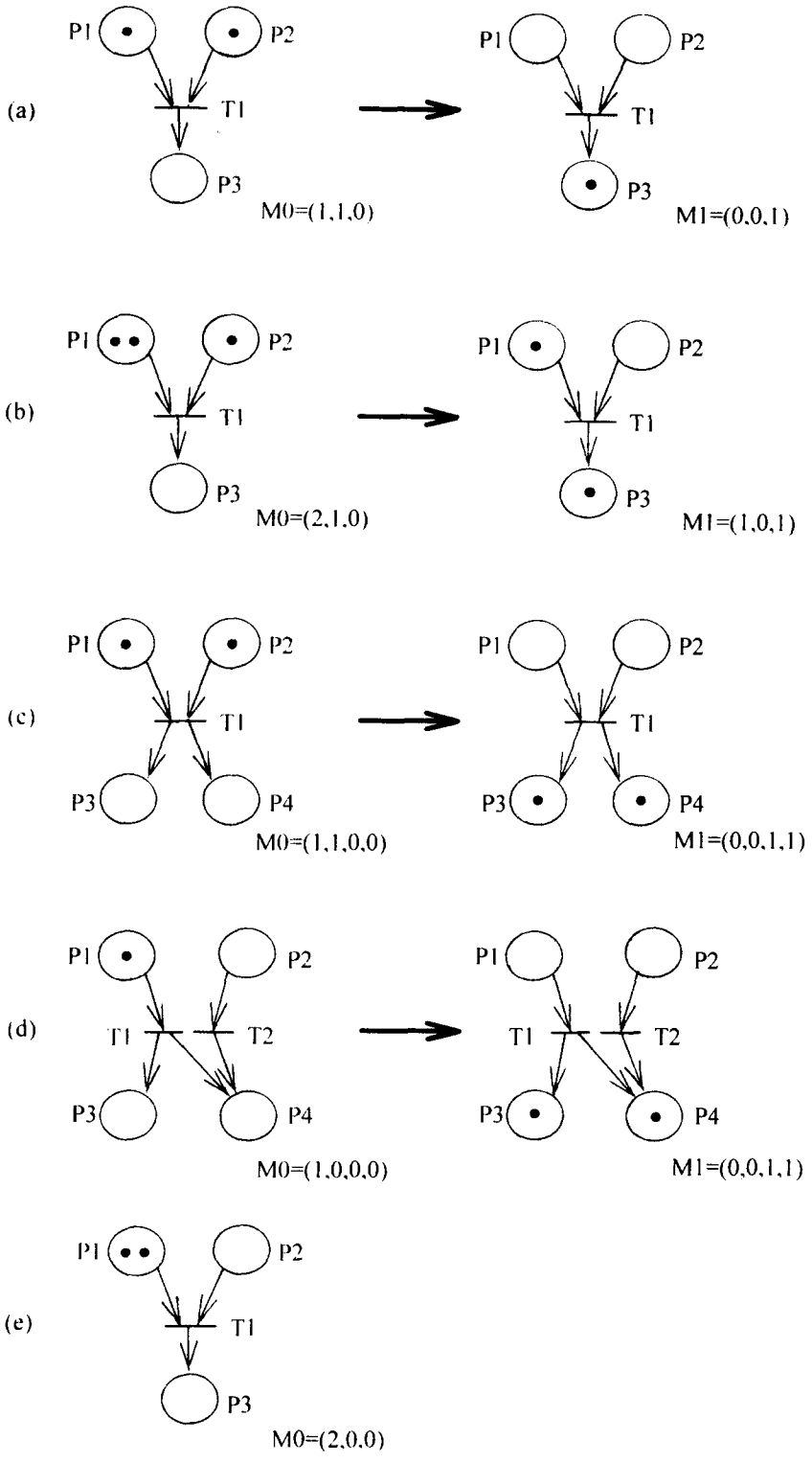


Fig. 1. Firing of transition.

firing ($M1$) are also illustrated in Fig. 1. The marking in Fig. 1(e) is invariant due to no firing, and hence no change of states. When a transition is enabled, it does not necessarily fire immediately but implies a possibility. Petri net modelling methods that involve firing conditions, the time factor, etc. have been developed [2].

The application of Petri nets to reliability engineering has been presented for reliability evaluation [3, 4], fault-tolerant analysis [5], safety analysis [6], reliability growth [7], Markov models [8], and stochastic models [9]. The literature used both Petri nets and fault tree analysis methods for software safety analysis [10], fault diagnosis [11], logic

flowgraph methodology [12], state equation representation of logic operation [13], and analysis of coherent fault trees [14]. The current study presents a matrix method based on Petri nets that are exemplified to outperform fault trees for reliability analysis. In addition, a trapezoidal graph method is devised to facilitate failure analysis and fault detection in the presence of sensors.

2 THE USE OF A PETRI NET TO REPRESENT FAULT TREE

In a manner similar to fault tree analysis, graphical models based on Petri nets can be constructed to represent cause-and-effect relationship among events. Since Boolean logic symbols are commonly used to account for failure causation, this work begins with examining logic gates based on Petri nets representation.

The logic operation for n binary variables requires 2^{2^n} functions. Hence, dealing with two binary variables, the number of Boolean functions is sixteen, among which only twelve functions deserve examination due to repetition of the other four. According to the enabling rules illustrated in Fig. 1, every logic gate can be represented by a Petri net model. The following presents these Petri net models, as shown in Fig. 2, where P_1 and P_2 are denoted as input places and P_3, the output place of transitions.

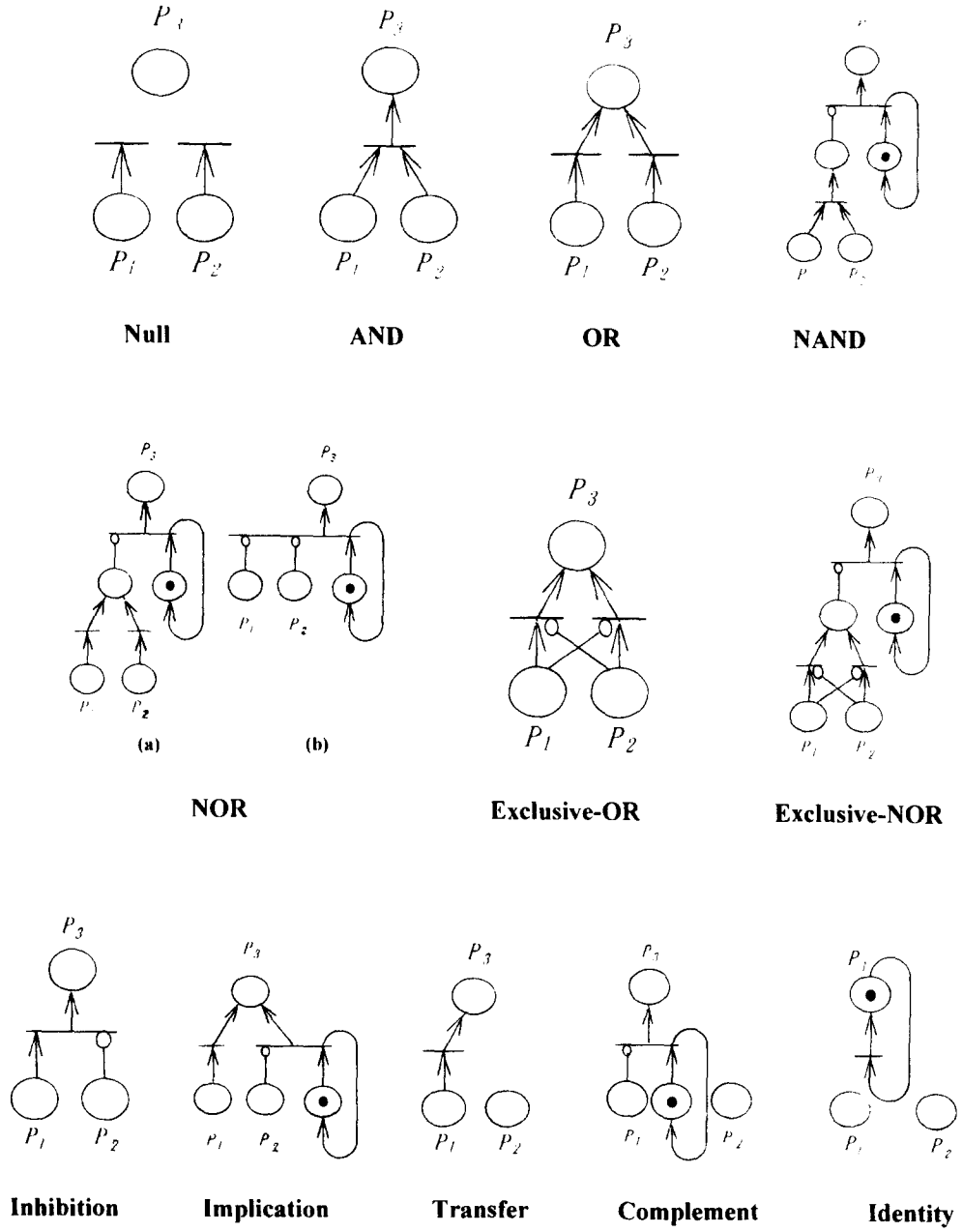


Fig. 2. Petri net model of logic operation.

1. Null ($P_3 = 0$): P_3 never occurs without regard to the situation of its inputs. There is no arc that connects to any output place, as depicted in Fig. 2.
2. AND ($P_3 = P_1\ P_2 = P_1 \cdot P_2$): In Fig. 2, both P_1 and P_2 must have at least one token [15] to achieve transition firing.
3. OR ($P_3 = P_1 + P_2$): At least one of input places must contain at least one token [14] in order to fire, as depicted in Fig. 2.
4. NAND ($P_3 = (P_1P_2)'$): As the complement of the AND function, it needs an inhibitor arc, whose end is always marked by a small circle, at the output place of the AND Petri net model, combining a self-looped place, as depicted in Fig. 2, to yield inputs of a transition.
5. NOR ($P_3 = (P_1 + P_2)'$): This is the complement of the OR function. As shown in Fig. 2-NOR(a), NOR needs an inhibitor arc at the output place of an OR model, accompanied by a self-looped place to constitute inputs of a transition. This model can be further simplified to become Fig. 2-NOR(b), in which two input places directly connect to the transition by inhibitor arcs.
6. Exclusive-OR ($P_3 = P_1P_2' + P_1'P_2$): A transition fires if either input place contains tokens but does not fire if both input places have one or no tokens. Figure 2 shows that each place inhibits transition firing of the other place, which is linked to an inhibitor arc [1].
7. Exclusive-NOR ($P_3 = P_1P_2 + P_1'P_2'$): A transition fires if, as illustrated in Fig. 2, both P_1 and P_2 have tokens or no token.
8. Inhibition ($P_3 = P_1P_2'$): A transition fires if, as illustrated in Fig. 2, P_1 has tokens but P_2 has no token.
9. Implication ($P_3 = P_1 + P_2'$): Fig. 2 depicts that P_1 must have tokens or P_2 no token to enable a transition to fire.
10. Transfer ($P_3 = P_1$): The token in P_1 directly transfers to P_3 shown in Fig. 2, while without relation with P_2.
11. Complement ($P_3 = P_1'$): It fires as long as P_1 has no token, as depicted in Fig. 2.
12. Identity ($P_3 = 1$): Fig. 2 depicts that place P_3 always contains a token. There is no arc that connects to input places.

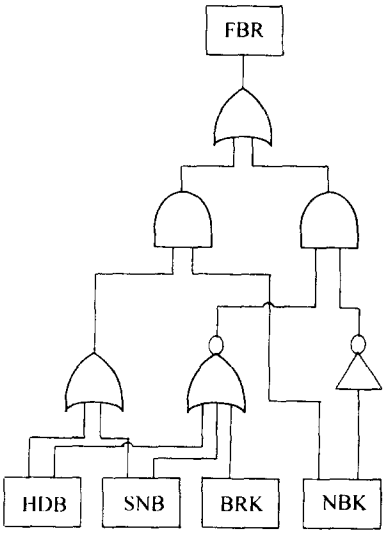
To exemplify the advantage of Petri net models over fault trees, consider a failure event of brake systems used in urban transit vehicles. There are four basic events, namely

1. *HDB*: holding brake signal
2. *SNB*: Snow brake signal
3. *NBK*: No brake demand
4. *BRK*: Service brake signal.

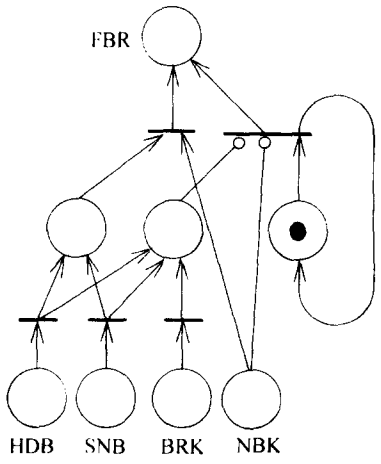
A logic equation representing the faulty brake event (*FBR*) of the brake system can be formulated as

$$FBR = ((HDB + SNB)\cdot NBK) + ((\overline{HDB + SNB + BRK})\cdot \overline{NBK}) \quad (1)$$

where $\bar{}$ denotes NOT. On the right hand side of eqn (1), the term ((HDB + SNB)· NBK) account for the situation that the vehicle is not braked although the vehicle is commanded to brake. The other term ($(\overline{HDB + SNB + BRK}\cdot \overline{NBK})$) accounts for the situation that not to brake whereas the vehicle brakes. Either situation represents malfunction of the brake system—the top event. Its logic gate model is drawn in Fig. 3(a). Instead, one can resort to its Petri net model as shown in Fig. 3(b) that is constructed based on Fig.



(a)



(b)

Fig. 3. Model transformation between Logic gate and Petri net.

2. Each of four basic places may have a token depending on which basic events occur. Enabling rules exemplified in Fig. 1 in turn determine transition firing and token movement. Consequently, if a token is present at the place for *FBR*, system failure is confirmed.

Petri nets can also represent condition gates in fault trees, as shown in Fig. 4. An inhibit gate belongs to a synchronized Petri net or interpreted Petri net whose transitions are associated with event E. A delay gate corresponds to a timed Petri net, in which to fire needs a time delay. Moreover, a R out of N gate belongs to a generalized Petri net model with N inputs. The output place will be a median place with weighted input arc R connecting to another transition. When R out of N tokens enter the median place, the transition fires and a token thus enters the output place.

Various event symbols in a fault tree can be replaced by only two simple symbols in a Petri net model, namely, places and steps as shown in Fig. 5. Both basic events and resultant events can be transformed into places. Undeveloped events in fault trees, if necessary, can be replaced by places not to be developed or steps to be developed to other Petri nets. Note that transfer symbols in fault trees are equivalent to steps in Petri net models that serve to simplify Petri net structures.

3 FAILURE PROBABILITY CALCULATION

As long as the failure probability of basic events in a fault tree is given, the reliability or failure probability of the top event can he calculated. It is known that calculating failure probability for a fault tree depends on gates. However, for Petri nets it depends on symbols of transitions, places and arcs.

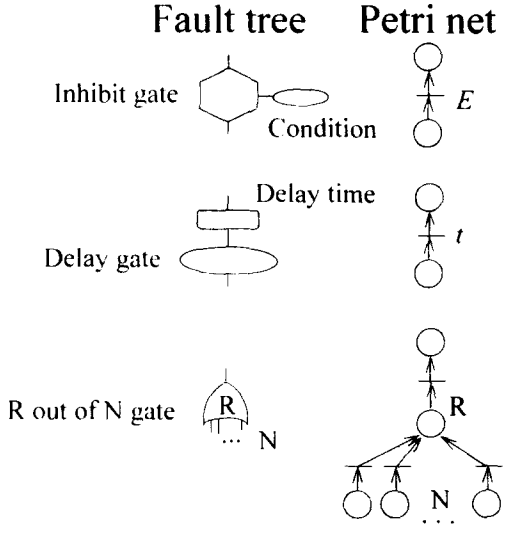


Fig. 4. Condition gates and Petri net models.

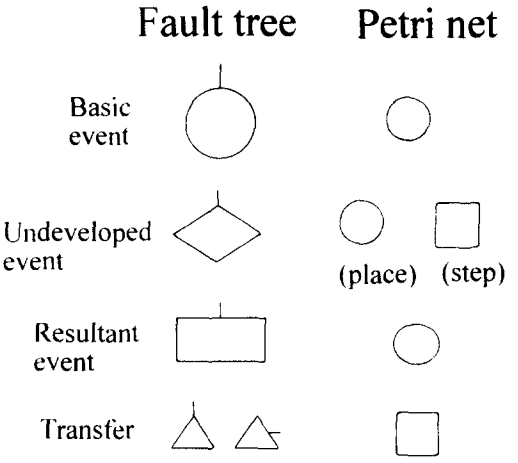


Fig. 5. Event symbols and Petri net models.

The failure probability P_f in case of a transition with multi-input places can be written as:

$$P_f = \prod_{i=1}^{n} P_{fi} \tag{2}$$

where P_{fi} denotes the failure probability of the ith event. By contrast, the failure probability for a place with multi-input transitions is written as:

$$P_f = 1 - \prod_{i=1}^{n} (1 - P_{fi}). \tag{3}$$

Symbols of gates, Petri nets, and equations for failure probability are shown in Fig. 6. It is not necessary to calculate for the transition or place with single input.

4 MINIMAL CUT SET AND PATH SET

An approach to generating minimal cut sets for a fault tree was defined by Dimitri [15]. Dealing with the fault tree depicted in Fig. 7, eight steps are required to obtain minimal cut set as listed in Table 1. Using Petri

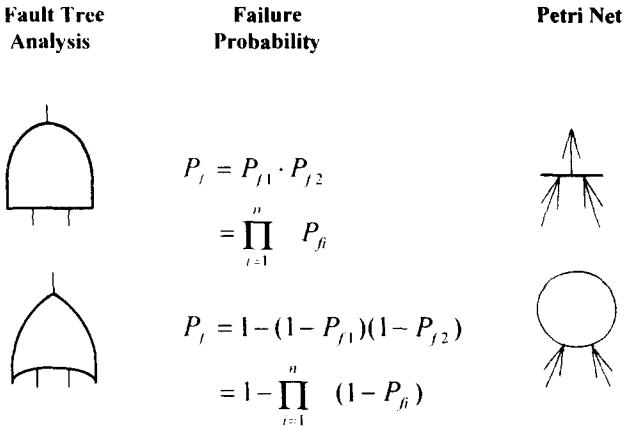


Fig. 6. Failure probability calculation.

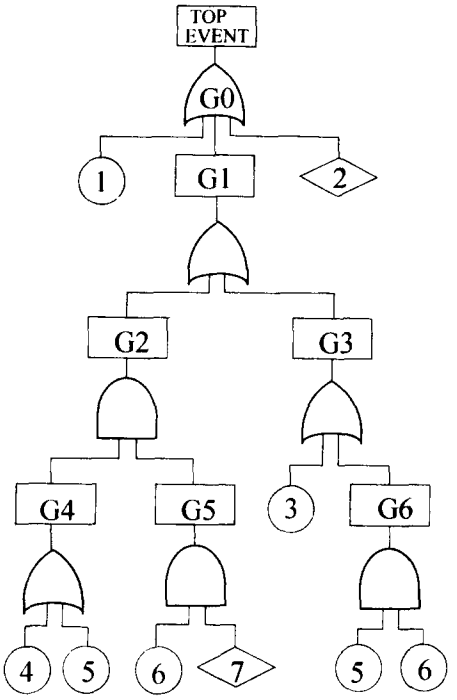


Fig. 7. Fault tree example.

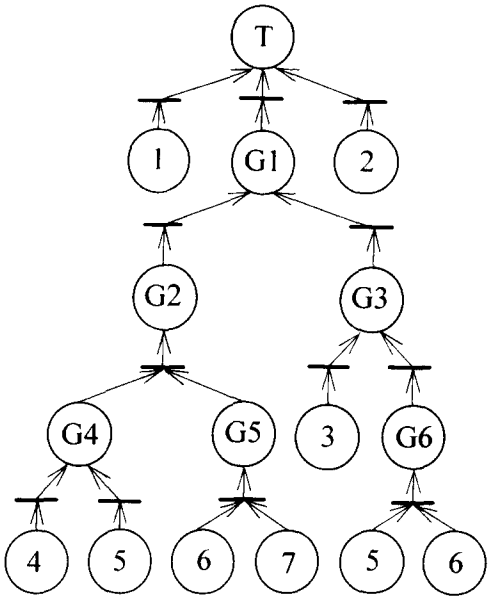


Fig. 8. Petri net corresponding to Fig. 7.

net models this section provides a new graphic method in order to determine minimal cut sets and path sets. The procedure is described as follows:

1. For basic places connected by arcs to reach the top place, if there is no multi-input transition along the arc route, the basic places become a minimal cut set.
2. If there exist multi-input transitions along the arc route that connects to the top place, locate their input place by a horizontal arrangement.
3. It is not necessary to deal with any more upper routes in the horizontal arranged place.
4. Replace the horizontal arranged places by their basic places, i.e., the horizontal arranged places will be sub-top places, and repeat procedure 1 to obtain its basic places.
5. Remove supersets whose elements contain elements of other cut sets. This results in the minimal cut set.

Consider the Petri net depicted in Fig. 8, only three steps, achieved by five computer instructions in total, are required to achieve the minimal cut set as listed in Table 2. The underlined numbers in Table 2 show that if upper routes in the horizontal arranged place are considered again (procedure 1(c)), it will still yield the same cut sets as those found in procedure 1(b). As a consequence, compared with the fault tree in Fig. 7 that needs eight steps, comprising nine computer instructions, to obtain the minimal cut set, the present Petri net method is more efficient.

To find path sets in Petri nets, this study proposes a model named dual Petri net as shown in Fig. 9. Multi-input transitions of a place can be combined into a transition with multi-inputs connected to the place. Conversely, a multi-input transition can be decomposed into transitions each with an input. After transforming a Petri net into a dual Petri net, minimal path sets can be found by performing the same procedure as that for finding minimal cut sets described above. For example, dealing with the Petri net shown in Fig. 8. its dual Petri net is drawn in Fig.

Table 1. Steps for obtaining cut sets of the fault tree in Fig. 7

Step No.	1	2	3	4	5	6	7	8
	GO	1	1	1	1	1	1	1
	2	2	2	2	2	2	2	
		G1	G2	G4,G5	G4,G5	4,G5	4,6,7	4,6,7
			G3	G3	3	5,G5		
					G6	3	3	3
						5,6	5,6	5,6
No. of instructions for each step	1	1	1	1	1	2	1	1
Total instruction numbers					9			

Table 2. Steps for obtaining cut sets of the Petri net in Fig. 8

Step No.	1	2	3
Places	1 2 4,G5 5,G5 6,7,G4 3 5,6	1 2 4,6,7 5,6,7 6,7,4 6,7,5 3 5,6	1 2 4,6,7 3 5,6
No. of instructions for each step	2	2	1
Total instruction numbers		5	

10, and three minimal path sets [1,2,3,4,5], [1,2,3,6], and [1,2,3,5,7] are obtained as depicted in Table 3.

5 A MATRIX METHOD

Minimal cut sets and path sets can he found at the same time using the present matrix method to analyze the Petri net from a top place to basic places. This method proceeds as follows:

1. Write down the numbers of places by making a horizontal arrangement if the output place is connected by multi-arcs to transitions.
2. Write down the numbers of places by making a vertical arrangement if the output place is connected by an arc to a common transition.
3. When all places are replaced by basic places, a matrix is established. If there is common entry located between rows or columns, it is the entry shared for each row or column. The column vectors of the matrix represent cut sets while row vectors path sets.
4. Remove the supersets to obtain the minimal cut sets and minimal path sets.

Figure 11 gives an example for the Petri net depicted in Fig. 8. Consequently, minimal cut sets are [1], [2], [3], [5,6], and [4,6,7] while minimal path sets include [1,2,3,6], [1,2,3,4,5], and [1,2,3,5,7]. They are exactly the same as the results in Tables 1 and 3, respectively. The present matrix method is more efficient than the conventional fault tree in that it is not necessary to transform the Petri net into a dual one in order to obtain both minimal cut sets and path sets.

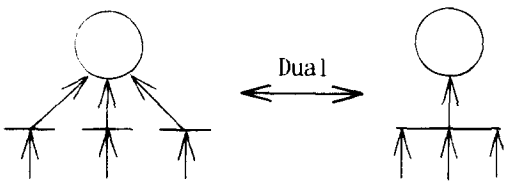


Fig. 9. Dual Petri net.

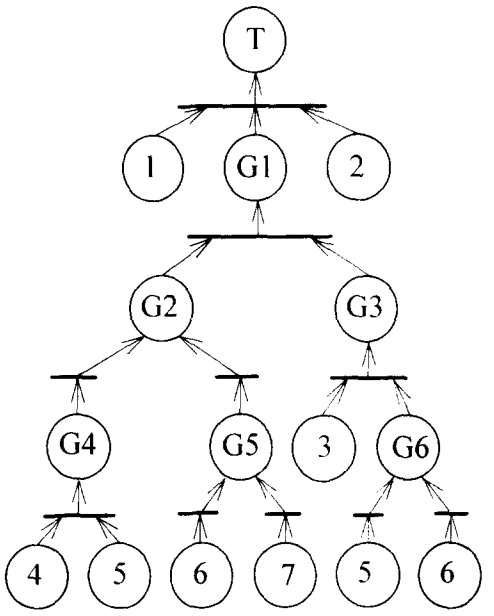


Fig. 10. Dual Petri net of Fig. 8.

6 ABSORPTION OF PETRI NET

A Petri net can be simplified as long as the firing time is not taken into account, i.e., the transfer of a token from an input place to an output place does not take time. When place P_l contains a token as shown in Fig. 12(a), transition T_l satisfies the firing condition. Hence, the token can be moved to place P_m. However, moving a token across transition T_l does not take time, since as depicted in Fig. 12(a) the transition is not timed. Therefore, places P_l and P_m and transition T_l can be altogether absorbed to become place P_l only, as shown in Fig. 12(b). In a similar manner, a token in place P_l can be subsequently transferred to P_n This Petri net model is thus absorbed to become a place $P_l = P_n$ as shown in Fig. 12(c).

If a transition has multi-input places, the Petri net can not be absorbed according to the above procedure. In case of hierarchical transitions consist-

Table 3. Steps for obtaining path sets of the Petri net in Fig. 8

Step No.	1	2	3
Places	4,5,G3,1,2 6,G3,1,2 7,G3,1,2	4,5,3,5,1,2 6,3,6,1,2 7,3,5,1,2	1,2,3,4,5 1,2,3,6 1,2,3,5,7

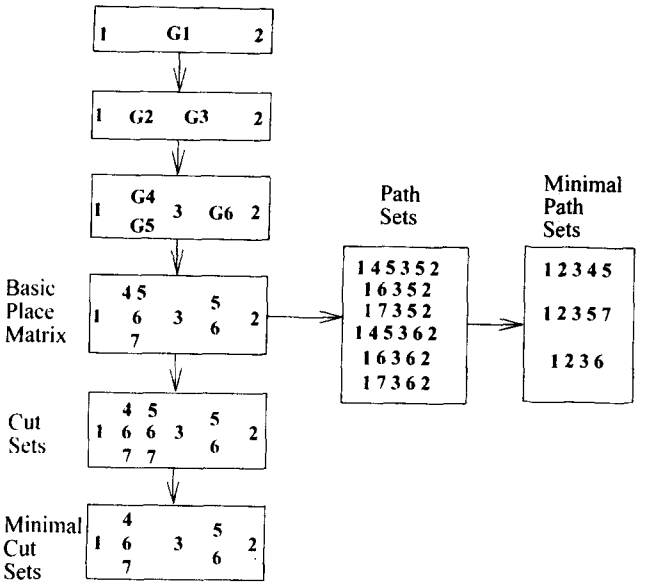


Fig. 11. Method for finding minimal cut sets and path sets.

ing of multi-input places, the transitions can be combined into a transition with basic multi-input places. For instance, in Fig. 13(a) T_1 has two input places P_i and P_j whereas an output place P_l is in turn one of input places for T_2. Both transitions can be combined into T_m endowed with three basic input places P_i, P_j, and P_k as shown in Fig. 13(b). In an analogous manner, the Petri net in Fig. 14(a) can be absorbed to become Fig. 14(b), and hence Fig. 14(c). It is interesting to note that the reorganized basic places for each transition constitute a minimal cut set.

7 MARKING TRANSFORMATION

The state of a Petri net is represented by marking M. The kth state M_k determines the next state M_{k+1}; i.e.

$$M_{k+1} = M_k + A^T S \quad k = 0,1,2,\ldots,n \qquad (4)$$

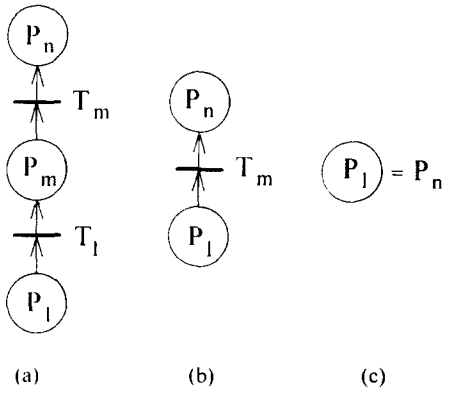


Fig. 12. Absorption of Petri net with transition from input to output place.

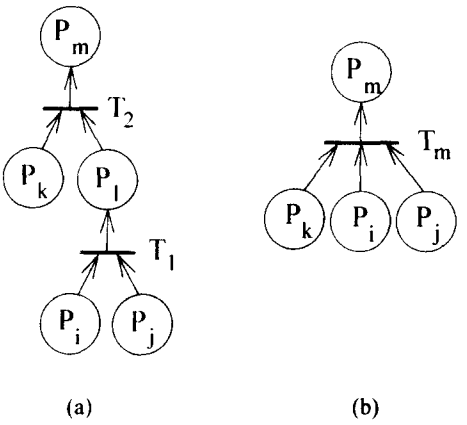


Fig. 13. Absorption of Petri net containing hierarchical transitions.

where

M_k is a column vector whose ith component is the marking of place P_i.

A^T is an incidence matrix whose rows are associated with places, and columns are associated with transitions. Each column corresponds to a marking modification when the associated transition fires.

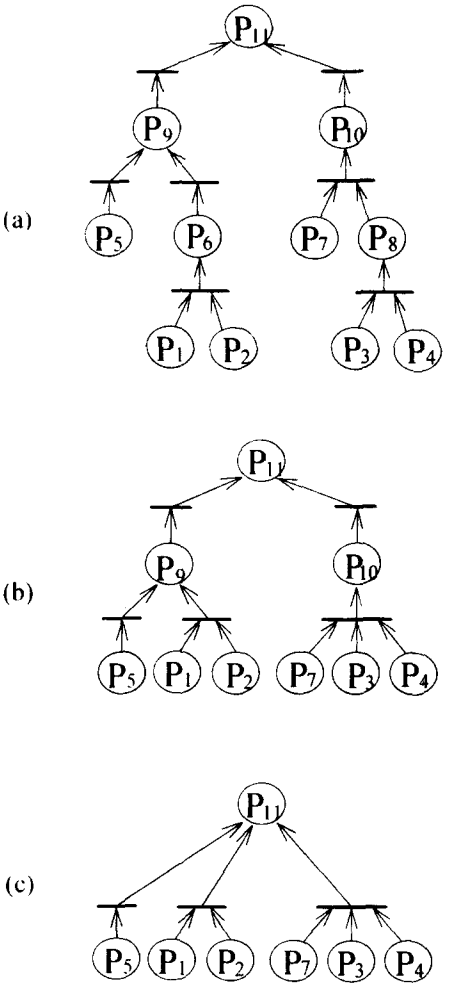


Fig. 14. Absorption of Petri net.

S represents a column vector whose ith component denotes the firing times of T_i.

Combining all marking transformations from an initial marking M_0 to final marking M_n, eqn (4) can be rewritten as

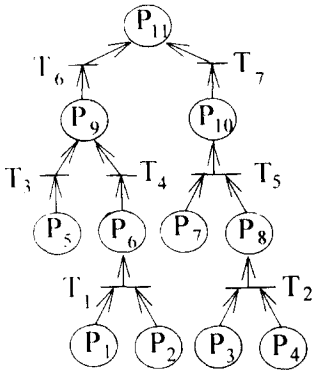
$$M_n = M_0 + A^T \sum \tag{5}$$

where

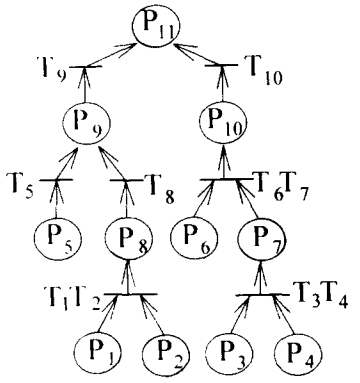
$$A^T \sum = \Delta M = M_n - M_0$$

and $\sum$ denotes a firing-count vector.

In Petri nets, whenever a token enters the top place, i.e., when the top event occurs in fault trees, the system becomes faulty. Therefore, by letting the last component in the marking column vector be the marking of the top place, the final marking M_n for failure analysis is of the form $M_n = [**\cdots *1]^T$, where n denotes the nth modification, and * represents the number of tokens that are not taken care of in all places other than the top place.



(a)



(b)

Fig. 15. Petri net for illustrating marking transformation and renumbering.

For example [14], dealing with the Petri net shown in Fig. 15(a), its incidence matrix becomes:

$$A^T = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \\ P_8 \\ P_9 \\ P_{10} \\ P_{11} \end{array} \begin{array}{c} \begin{array}{ccccccc} T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 \end{array} \\ \begin{pmatrix} -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 \\ 0 & 1 & 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 \end{pmatrix} \end{array} \tag{6}$$

Assume an initial marking $M_0 = [10111010000]^T$. A firing sequence $T_2T_5\ T_7$, i.e., the firing-count vector $\Sigma_1 = [0100101]^T$ transforms M_0 to $M_n = [10001000001]^T$. In a similar manner, a firing sequence T_3T_6, i.e. the firing-count vector $\Sigma_2 = [0010010]^T$ transforms M_0 to $M_n = [10110010001]^T$. This study treats the marking as a system state. Thus, the variation of failure states can he delineated by proposed marking transformation.

8 RENUMBERING INCIDENCE MATRIX

This paper proposes an entry arrangement method to obtain incidence matrices in a unified manner:

1. Assign numbers to basic places.
2. The numbers of places and transitions are the same. If there are multiple input places connected to a common transition, the number of this transition has as many characters as the number of input places.
3. Number other output places and transitions.
4. Put the numbers in entries of an incidence matrix.
5. Append a column with its last entry as -1 to the right of the matrix. A square matrix is thus formed.

In fact, the square matrix is a triangular matrix. If there are n basic places and totally l places, the upper-left $n \times n$ sub-matrix is a negative identity matrix, while its upper-right $n \times (l - n)$ sub-matrix is a null matrix.

Based on the procedure, the number modification of places and transitions is carried out for the Petri net

model depicted in Fig. 15(a), which results in Fig. 15(b). As a result, its incidence matrix becomes:

$$A^T = \begin{array}{c} \\ P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \\ P_6 \\ P_7 \\ P_8 \\ P_9 \\ P_{10} \\ P_{11} \end{array} \begin{array}{c} \begin{array}{cccccccccccc} T_1 & T_2 & T_3 & T_4 & T_5 & T_6 & T_7 & T_8 & T_9 & T_{10} & T_{11} \end{array} \\ \left(\begin{array}{cccccccccccc} -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & -1 \end{array} \right) \end{array} \tag{7}$$

This resultant matrix facilitates subsequent marking transformation analysis. It has a simple form and can be used to avoid a wrong firing sequence, since the numbers of places and transitions are the same. Whenever a token appears in P_i it means T_i can fire. In the case of a transition with two input places, there are two characters for numbering. It leads to two entries each with value of 1 in a row. These two entries are underlined for ease of identification. The transition with underlined entries does not fire unless tokens appear in both input places.

Assume an initial marking $M_0 = [1011010000]^T$ and put this vector under the renumbering incidence matrix. As depicted in Fig. 16, the transition T_3T_4 can fire since each of the corresponding places P_3 and P_4 has a token. The firing of T_3T_4 removes the tokens of P_3 and P_4, puts one token in P_7 and transforms the marking to become $M_1 = [100001\ 10000]^T$. In a similar manner, the transition T_6T_7 can fire since their corresponding places P_6 and P_7 possess tokens, respectively. The firing of T_6T_7 removes the tokens of P_6 and P_7, puts one token in P_{10}, and transforms the marking to yield $M_2 = [10000000010]^T$. Finally, transition T_{10} fires and ends up with the marking $M_3 = [10000000001]^T$. The last entry of this marking turns out to be 1, which represents a system failure.

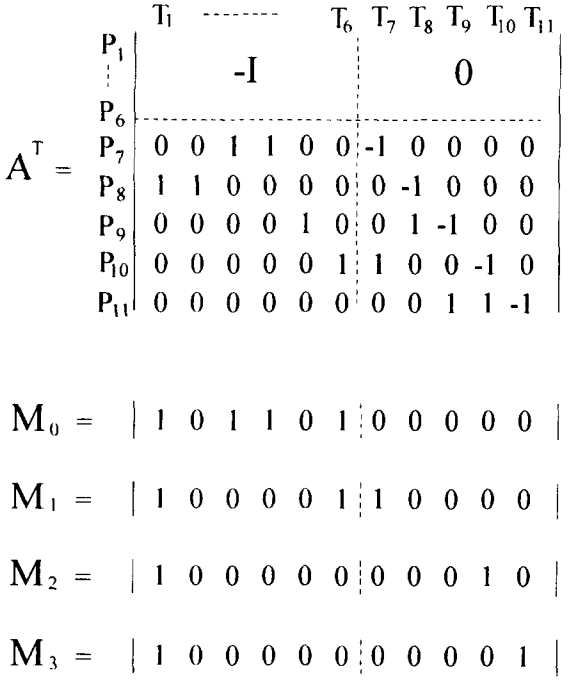


Fig. 16. Marking transformation by renumbered incidence matrix.

9 TRAPEZOIDAL GRAPH METHOD

A new diagrammatic method is proposed in this section. Retaining the lower part in the renumbering incidence matrix, as depicted in eqn (7), but excluding the negative identity and null sub-matrices results in a rectangular graph. It contains an $(l-n)\times n$ sub-matrix and an $(l-n)\times(l-n)$ triangular sub-matrix. Deleting further the null part of this triangular sub-matrix yields a trapezoidal graph, shown in Fig. 17.

Every entry with digit of 1 in Fig. 16 can be represented by a thin bar as shown in Fig. 17. However, if in any row with digit of 1 there are entries that have been underlined, the bar is drawn as thick one. Under the trapezoidal graph, write down numbers from 1 to the number of the top place, and write down the numbers of output places beside the numbers of basic places on the left side of this graph. Accordingly, token transformation in the Petri net model can be carried out as follows:

1. Put tokens on the top of the graph if the corresponding numbers of basic places contain tokens.
2. Let the tokens fall down.
3. The tokens will hit the bars.
4. As long as either a thin bar loads a token or a thick bar fully load tokens, a token will roll horizontally to the right hand side.
5. Once the rolling token hits the slope in the graph, it falls down again and may hit a lower bar.
6. Repeat steps 4 and 5 until no token can be rolled any further.

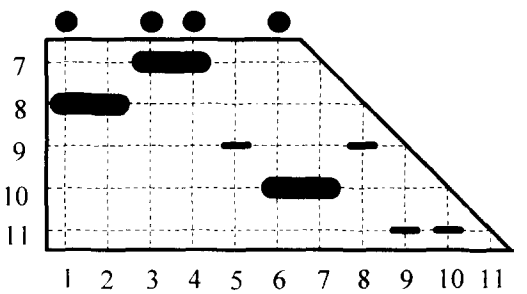


Fig. 17. Token transformation using trapezoid.

7. If any token rolls down to reach the number of the top place, it represents that the system fails.

In the present example, since $M_0 = [1011010000]^T$, each of the basic places P_1, P_3, P_4 and P_6 has one token. Thus, put the tokens on the top of the trapezoidal graph. The tokens for places P_1 and P_6 drop, and subsequently stay on their corresponding thick bar. Moreover, the tokens in P_3 and P_4 also drop and are fully loaded by the thick bar. One of the tokens rolls horizontally to the right hand side, hits the slope and hence drops again. It is in turn loaded by the thick bar that has already loaded a token coming from P_6. The bar fully loads tokens now, therefore one of the tokens is allowed to roll to the right until it hits the slope. This token drops again to hit a lower thin bar. This token rolls out again, hits slope and drops to arrive at number 11, i.e., the number of the top place. Accordingly, the system fails.

Furthermore, it can be seen that simply one event P_1 does not cause this system to fail, since it does not constitute a cut set. This fact can also be checked based on the same graph. To that end, let a token corresponding to number 1 fall down from the graph top. Although the token is in turn loaded by a thick bar, no subsequent movement can be activated. Suppose, however, event P_2, also occurs, P_1 and P_2 will form a cut set. It is observed that another token falling down from number 2 enables a thick bar to fully load tokens. One of the two tokens in turn rolls horizontally to the right. As a consequence, the token drops to reach number 11, which accounts for system failure. Hence, the proposed trapezoid method has been verified to be effective.

Note that by either minimal cut sets or logic algorithm methods, the cause and effect relationship among events in a system can not be disclosed. They can only identify whether events lead to system failure; therefore, they can be treated as 'static' analysis methods. On the contrary, the present novel method is capable to account for failure scenarios. The token motion in the trapezoidal graph behaves in a manner similar to transition firing and token transferring in Petri nets. The success of this trapezoidal graph method is attributed to its accounting for numbering evolution in the incidence matrix.

10 FAULT DETECTION USING PETRI NETS

In Petri nets, every place accounts for a state in a system. When a fault tree is transformed into a Petri net model, the presence of tokens in the top place implies that failure occurs. In an autonomous Petri net, a token may appear in an input place and satisfy the firing condition. As a consequence, the token moves into the output place. A basic failure thus leads to the higher hierarchical failure or serious failure as shown in Fig. 18(a).

In practice, machines and factories may contain sensors to detect failures and prevent basic failure from endangering system safety, for which sensors can be incorporated in Petri nets as shown in Fig. 18(b). When the basic failure occurs, the firing makes a token enter the intermediate place for isolating, while another token enters the sensor block to represent detection by a sensor. If the detection is ineffective, the token will leave the sensor block. The subsequent firing of a higher hierarchical transition causes the tokens to move into the output place. As a result, failure occurs.

In addition, to model corrective maintenance or replacement of a faulty unit following fault detection, tokens in the intermediate place and the sensor block

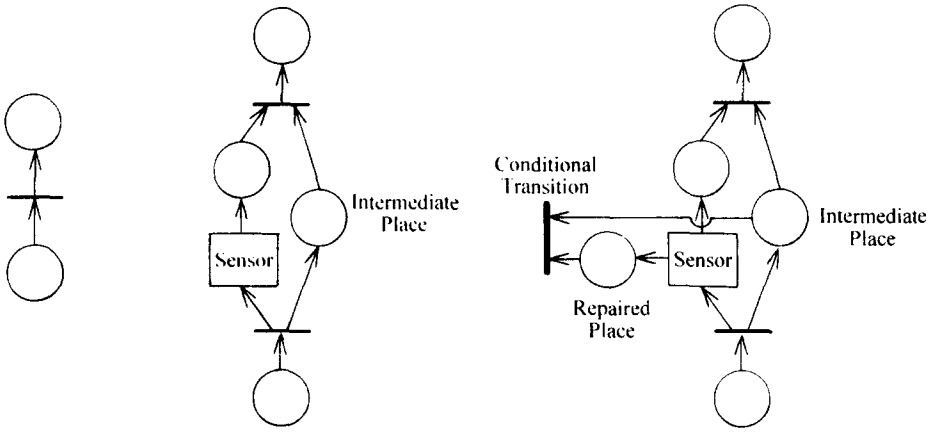


Fig. 18. Fault detection and repaired process in Petri net.

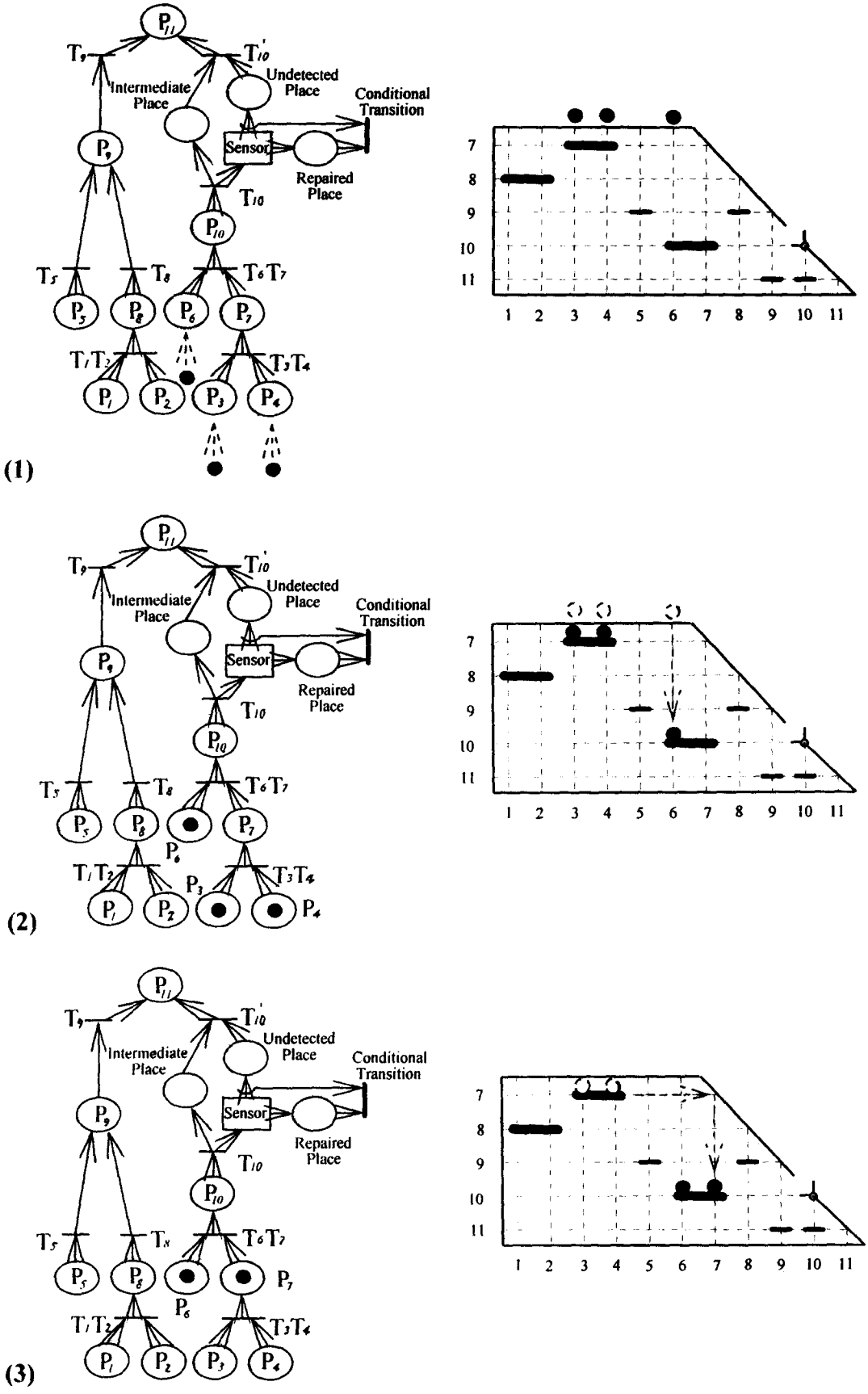
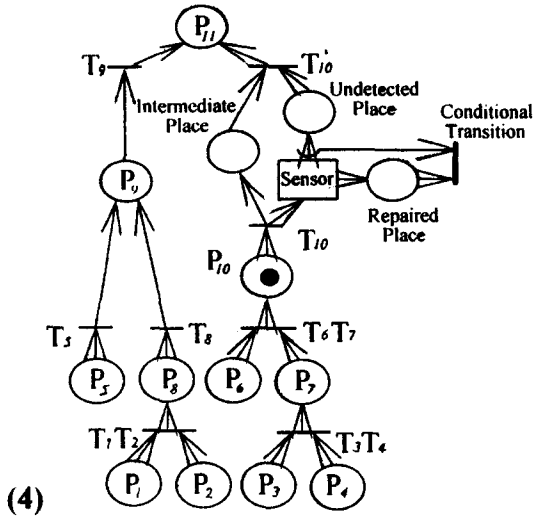
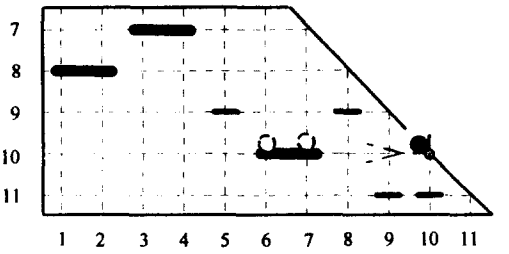
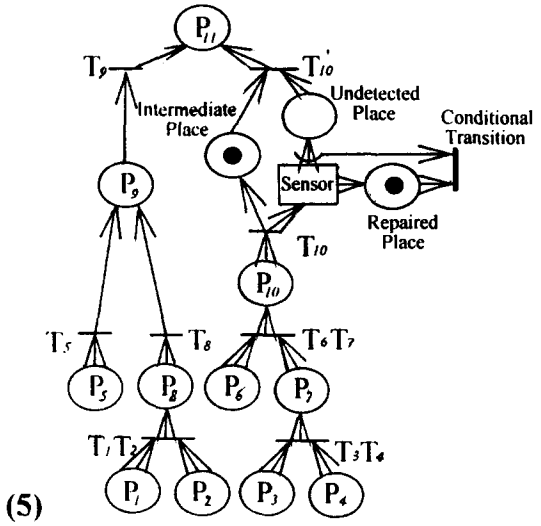


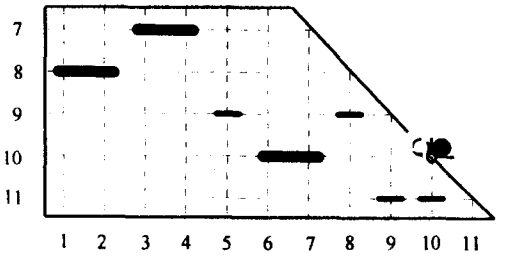
Fig. 19. Sequence of fault detection.



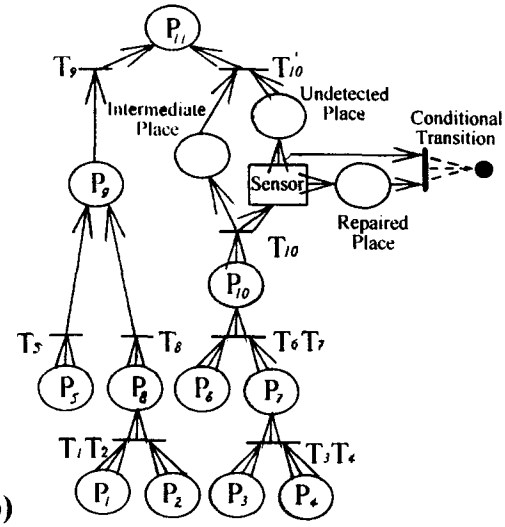


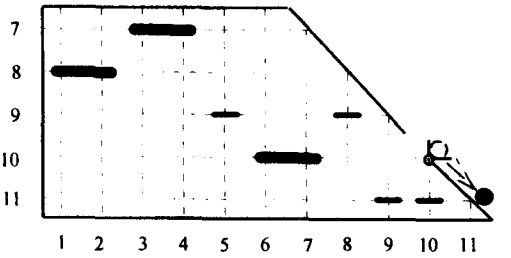
(4)





(5)





(6)

Fig. 20. Sequence of fault detection(II).

should be eliminated after the failure is repaired. This can be achieved by installing a conditional transition, as shown in Fig. 18(c), which is connected to both intermediate place and sensor block. If the basic failure has been detected and is under repair, tokens will be put in both intermediate place and repaired place. After the repair work is done, the conditional transition fires, and tokens in both places will move out of the Petri net.

Figures 19 and 20 exemplify fault detection. Suppose that a sensor is employed for detecting a fault that occurs at place P_{10} and the system has initial marking at P_3, P_4, and P_6 where each contains a token. The firing of transition T_3T_4 transfers tokens in both P_3 and P_4 into a token in place P_7. The tokens in P_6 and P_7 are in turn transferred into a token at P_{10} due to the firing of T_6T_7. Consequently, fault arising from P_{10} can be detected by the sensor, and the token is transferred into an intermediate place. During a maintenance period, a token stays at a repaired place. After repair, tokens located at both intermediate and repaired places vanish due to firing of the conditional transition.

Furthermore, the foregoing scenario of repair activity can also be illustrated using the trapezoidal graph. To that end, as depicted in Fig. 18, make a notch on the slope corresponding to the number of places that represent sensors, and at the notch attach a horizontal inverse L-shaped hinge. When a token rolls to the right hand side on this level, the token will be stopped by the hinge. It dictates that fault is detected by the sensor installed at the corresponding place. Once repaired, the hinge turns clockwise, and the token will roll out of the trapezoid along the slope. This represents the absence of the failure.

11 CONCLUSION

This paper has presented a Petri net method to efficiently obtain minimum cut sets and path sets. Additionally, the proposed matrix method represents a new approach to arranging place numbers so as to obtain both minimum cut sets and path sets.

Place and transition numbers in Petri nets has been modified to result in incidence matrices, which are convenient for marking development, i.e., failure state analysis. Furthermore, by exploiting evolution of renumbering, the renumbered incidence matrix has been transformed into a trapezoidal graph so as to account for system failure scenarios. It is an effective graphical method capable to undertake the function of Petri nets. Examples have demonstrated that marking transfer, system failure and fault detection can be achieved using this proposed method.

REFERENCES

1. Peterson, J. L., *Petri Net Theory and the Modelling of Systems*. Prentice-Hall, Englewood Cliffs, New Jersey, 1981.
2. David, R. and Alla, H., Petri net for modelling of dynamic systems— a survey. *Automatica*, 1994, **30**(2), 175–202.
3. Kumar, V. and Aggarwal, K. K., Petri net modelling and reliability evaluation of distributed processing systems. *Reliability Engineering & System Safety*, 1993, **41,** 167–176.
4. Misra, K. B., *New Trends in System Reliability Evaluation*. Elsevier Science, Amsterdam, 1993.
5. Viswanadham, N., *Reliability of Computer and Control Systems*. Elsevier Science, New York, 1987.
6. Leveson, N. G. and Stolzy, J. L., Safety analysis using petri nets. *IEEE Transactions on Software Engineering*, 1987, **133,** 386–397.
7. Shabalin, A. N., Generation of models for reliability growth. In *Proceedings of the* 1992 *Annual Reliability and Maintenance Symposium*, IEEE, New York, pp. 299–302, 1992.
8. Haverkort, B. R. and Trivedi, K. S., Specification techniques for Markov reward models. *Discrete Event Dynamic Systems: Theory & Applications*, 1993, **3,** 219–247.
9. Sahner, R. A. and Trivedi, K. S., A software tool for learning about stochastic model. *IEEE Transactions on Education*, 1993, **361,** 56–61.
10. McGraw, R. J., Shimeall, T. J. and Gill, J. A., Software safety analysis in heterogeneous multiprocessor control systems. In *Annual Reliability and Maintenance Symposium* 1991 *Proceedings*, IEEE, New York, pp. 290–294, 1991.
11. Viswanadham, N. and Johnson, T. L., Fault detection and diagnosis of automated manufacturing systems. In *Proceedings of the 27th IEEE Conference on Decision and Control*, Austin, TX. Vol. 3, pp. 2301–2306, 1988.
12. Muthukumar, C. T., Guarro, S. B. and Apostolakis, G. E., Logic flowgraph methodology: a tool for modelling embedded systems. In *Proceedings of the IEEE/AIAA 10th Digital Avionies Systems Conference*, New York, pp. 103–109, 1991.
13. Khan, A. A., State equation representation of logic operations through a petri net. *Proceedings of the IEEE*, 1981, **694,** 485–487.
14. Hura, G. S. and Atwood, J. W., The use of petri nets to analyze coherent fault trees. *IEEE Transactions on Reliability*, 1988, **375,** 469–474.
15. Dimitri, K., *Reliability Engineering Handbook*, Vol. 2. Prentice-Hall, Englewood Cliffs, New Jersey, 1991.