CONSTRUCTING AN ABSTRACT MODEL FOR LADDER DIAGRAM DIAGNOSIS USING PETRI NETS

Jui-I Tsai and Ching-Cheng Teng

ABSTRACT

This paper proposes a method for constructing an abstract model for analyzing and diagnosing electrical circuit ladder diagrams (LDs) using Petri nets, and also supporting network-based monitoring and supervision. This approach converts normal open (NO) and normal close (NC) contacts in the LD into Petri net transitions, and converts devices (e.g., relay coils) in the LD into Petri net places. This study introduces the concepts of composite transitions, composite places, and relevant state to reduce complexity and increase readability of Petri nets for constructing abstract models. The current study constructing diagnosis of fault modeling, introduces simple matrix manipulation and the difference output vector (DOV) to determine the faulty area for diagnosis in the ladder diagram. An LD controller example demonstrates the usable approach.

Key Words: Ladder diagram; Petri net; diagnosis; abstract model; BPN; fault model.

I. INTRODUCTION

Researchers developed ladder diagrams (LDs) to replace relay symbols, and these diagrams have been popular for a long time. Using a LD, it is possible to represent the automated control process both sequentially and graphically.

Petri nets (PNs) describe and analyze information flow, and are excellent tools for modeling asynchronous concurrent systems, such as computer systems, manufacturing systems, Supply Chain Management [1], video streaming systems [2] and power systems [3, 4].

Sequence controllers play a key role in industrial practice or manufacturing systems. Traditionally, ladder diagrams have been widely applied to programmable logic controllers, while Petri nets have supplied an alternative tool for the sequence control of complex systems.

Jackman et al. [5] proposed a conceptual model and working equation for converting relay ladder logic to the PN model. Lee et al. [6] presented a method for obtaining an augmented PN from an LD, and then using the Petri net state equation as an analysis technique to validate the corresponding flow mechanism of the generated PNs. However, this approach increases the total number of nodes and links in the generated PNs [7]. Venkatesh et al. [8, 9] and Peng et al. [7] modeled an LD contact to place and increase a virtual transition. Lee et al. [10] modeled an LD connect to transition and increase a position. However, the total number of nodes and links in the generated Petri nets increases, which in turn increases complexity. Petri nets have an excellent flexibility corresponding to different models according to a variety plant. This paper proposes a Boolean Petri Net (BPN) Model.

Hierarchical control is an approach to designing large-scale discrete event systems to deduce complexity [10] as Fig. 1 illustrates. In a manufacturing system, an LD controller may use a local controller, enabling the LD controller to be diagnosed and monitored.
Boolean transition. Table I describes the BPN model the BPN model. This special transition is called the Boolean equation in a PN transition to develop equation. The approach proposed in this paper embeds the equation. The simplest way to represent an LD is by its Boolean equation. Section IV presents an example of a PN. Section III builds a table of a one or more LD corresponding to the BPN module using the Boolean function or variable can be ‘1’, in which case the related transition \( t_j \) is allowed to fire if it is enabled, or it can be ‘0’, in which case the related transition \( t_i \) is.


\[
PN = (P, T, A, I, O, in, out, M_0)
\]

where \( P = \{p_1, p_2, \ldots, p_m\}, m \geq 1 \), is a finite set of places representing the LD action state. The places are associated with a component or a set of components (i.e., a compound component) such as the actuator output, relay coil, timer, counter solenoid, or source; \( T = \{t_1, t_2, \ldots, t_n\}, n \geq 1 \), is a finite set of transitions representing event whether occurs or not. These transitions are always associates with a switch or a set of switches and represented by Boolean equations or variables.

The switch can be a normal open (NO) switch or normal closed (NC) switch. The NO switch is also called an “a” contact and the NC switch is also called a “b” contact, where \( P \cap T = \emptyset \) and \( P \cup T \neq \emptyset \).

\[ A \subset (P \times T) \cup (T \times P) \]

is a set of arcs (→) consisting of input arcs \( A_1(P \times T) \) and output arcs \( A_0(T \times P) \). The weight of each directed arc in this paper is 1, and \( A_k(P \times T) \) is defined as directed arcs from a place to a transition. Places are called input places and transitions are called output transitions, and the input arc is represented by a connected line as channel of token. \( A_o(T \times P) \) is defined as directed arcs from a transition to a place, the transition is called the input transition and the place is called the output place, and the output arc is represented by a connected line as channel of token. The arc may be preservation arc \( \bullet \rightarrow \) that a input arc and a output arc exist simultaneously between same place and transition [6]; \( I : T \times P \rightarrow N \) is an input function that defines as number of output arcs \( A_o(T \times P) \), where \( N = \{0, 1, 2, \ldots\} \); \( O : P \times T \rightarrow N \) is an output function that are defined as number of input arcs \( A_i(P \times T) \), where \( N = \{0, 1, 2, \ldots\} \); \( in = \{i_1, i_2, \ldots, i_n\} \) is a set of input switch, which is represented by Boolean function or variable. A set of input switches is associated with a transition \( t_j \) and is denoted by \( t_j = \{in\} \). The Boolean function or variable can be ‘1’, in which case the related transition \( t_j \) is allowed to fire if it is enabled, or it can be ‘0’, in which case the related transition \( t_i \) is.

II. BOOLEAN PETRI NET

Carl Adam Petri proposed the Petri net theory. Fig. 2 shows the structure of Petri nets in a directed bipartite graph that consists of places, transitions, and arcs. A circle with a token represents the places. A bar that indicates the flow of tokens when firing condition is satisfied, which represents the transition. Finally, a straight line that connects the place to the transition, or the transition to the place denotes the arc, which indicates the flow of tokens in the direction of the arrow.

2.1 Definition of boolean petri nets

The purpose of developing the BPN model is that this model exhibits the imply logic property in a LD. The simplest way to represent an LD is by its Boolean equation. The approach proposed in this paper embeds the Boolean equation in a PN transition to develop the BPN model. This special transition is called the “Boolean transition.” Table I describes the BPN model.
<table>
<thead>
<tr>
<th>Modules</th>
<th>Ladder Diagrams</th>
<th>Boolean Equations</th>
<th>Boolean Petri Nets</th>
<th>Modules</th>
<th>Ladder Diagrams</th>
<th>Boolean Equations</th>
<th>Boolean Petri Nets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1</td>
<td><img src="image1" alt="Image" /></td>
<td>$M = A$</td>
<td><img src="image2" alt="Image" /></td>
<td>Type 3</td>
<td><img src="image3" alt="Image" /></td>
<td>$M = AB$</td>
<td><img src="image4" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td><img src="image5" alt="Image" /></td>
<td>$M = T_A$</td>
<td><img src="image6" alt="Image" /></td>
<td>Type 4</td>
<td><img src="image7" alt="Image" /></td>
<td>$M = A + B$</td>
<td><img src="image8" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td><img src="image9" alt="Image" /></td>
<td>$M = A$</td>
<td><img src="image10" alt="Image" /></td>
<td>Type 5</td>
<td><img src="image11" alt="Image" /></td>
<td>$M = AB$</td>
<td><img src="image12" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td><img src="image13" alt="Image" /></td>
<td>$M = A + M$</td>
<td><img src="image14" alt="Image" /></td>
<td>Type 6</td>
<td><img src="image15" alt="Image" /></td>
<td>$M_1 = A$</td>
<td><img src="image16" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td><img src="image17" alt="Image" /></td>
<td>$M_2 = A$</td>
<td><img src="image18" alt="Image" /></td>
<td></td>
<td></td>
<td>$M_1 = M_2$</td>
<td></td>
</tr>
<tr>
<td>Type 2</td>
<td><img src="image19" alt="Image" /></td>
<td>$\overline{M} = A$</td>
<td><img src="image20" alt="Image" /></td>
<td>Type 7</td>
<td><img src="image21" alt="Image" /></td>
<td>$M = (A + M)\overline{B}$</td>
<td><img src="image22" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td><img src="image23" alt="Image" /></td>
<td>$\overline{M} = T_A$</td>
<td><img src="image24" alt="Image" /></td>
<td>Type 8</td>
<td><img src="image25" alt="Image" /></td>
<td>$M = (A + M)\overline{B} = AB$</td>
<td><img src="image26" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td><img src="image27" alt="Image" /></td>
<td>$N = M$</td>
<td><img src="image28" alt="Image" /></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type 9</td>
<td><img src="image29" alt="Image" /></td>
<td>$\overline{M} = A$</td>
<td><img src="image30" alt="Image" /></td>
<td>Type 9</td>
<td><img src="image31" alt="Image" /></td>
<td>$Timer = A$</td>
<td><img src="image32" alt="Image" /></td>
</tr>
<tr>
<td></td>
<td><img src="image33" alt="Image" /></td>
<td></td>
<td><img src="image34" alt="Image" /></td>
<td></td>
<td></td>
<td>$M = T_A$</td>
<td></td>
</tr>
<tr>
<td>Modules</td>
<td>Ladder Diagrams</td>
<td>Boolean Equations</td>
<td>Boolean Petri Nets</td>
<td>Modules</td>
<td>Ladder Diagrams</td>
<td>Boolean Equations</td>
<td>Boolean Petri Nets</td>
</tr>
<tr>
<td>---------</td>
<td>----------------</td>
<td>-------------------</td>
<td>-------------------</td>
<td>---------</td>
<td>----------------</td>
<td>-------------------</td>
<td>-------------------</td>
</tr>
<tr>
<td>Type 10</td>
<td><img src="image1" alt="Diagram" /></td>
<td>$M = A$</td>
<td>$N = A$</td>
<td>Type 11</td>
<td><img src="image2" alt="Diagram" /></td>
<td>$\bar{B} = T_\Delta$</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\bar{B} = T_\Delta + C$</td>
<td>$C = T_\Delta \bar{B}$</td>
<td>Type 12</td>
<td><img src="image3" alt="Diagram" /></td>
<td>$N = A\bar{N}$</td>
<td>$M = A\bar{N}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\bar{B} = T_\Delta + C$</td>
<td>$C = T_\Delta \bar{B}$</td>
<td>Type 13</td>
<td><img src="image4" alt="Diagram" /></td>
<td>$N = B\bar{M}$</td>
<td>$M = A\bar{N}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>$\bar{B} = T_\Delta + C$</td>
<td>$C = T_\Delta \bar{B}$</td>
<td>Type 14</td>
<td><img src="image5" alt="Diagram" /></td>
<td>$N = B\bar{M} + B + M$</td>
<td>$\bar{N} = B + A\bar{N}$</td>
</tr>
<tr>
<td>Timer</td>
<td><img src="image6" alt="Diagram" /></td>
<td>$B = A(T_\Delta)$</td>
<td>$C = T_\Delta$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
not allowed to fire; \( out = \{ out_1, out_2, \ldots, out_n \} \) is a set of output actuator which is associated with a \( p_i \) and is denoted by \( p_i = \{ out \} \). \( M_0(P) \) is the initial marking that uses a token to represent the place status.

A transition is enabled if the number of tokens at the place is larger than or equal to the number of input arcs. A transition is firing if the enabled transition is fired and its transition states are true (i.e., the Boolean equation is true). When a transition fires, it moves the tokens from input places to output places along the input equation is true). When a transition fires, it moves the tokens from input places to output places along the input arcs and output arcs, as Fig. 2 illustrates. This moves the token of place \( A \) to place \( B \) along directed arcs if transition \( t_i \) is firing. A marking is denoted as an \( m \)-vector, where \( m \) is the total number of place \( P \), while \( m(p_i) \) is represented by the number of tokens at place \( p_i \) [11].

For the marking \( m_0 \), there is an enabled transition \( t_1 \). If there is a firing of transition \( t_1 \), then the marking is immediately reachable to \( m' \) from \( m_0 \), denoted by \( m_0(t_1) \rightarrow m' \). A marking \( m_i \) is said reachable from \( m_0 \) if there exists a sequence of firings that transforms \( m_0 \) to \( m_i \). \( R(m_0) \) is defined as the set of all reachable markings from \( m_0 \). \( F(m_0) \) is defined as the set of all firing sequences from \( m_0 \). A place \( p_i \) is said to be bounded for an initial marking \( m_0 \) if \( \exists k > 0, m(p_i) \leq k \), and \( \forall m \in R(m_0) \). Specifically, it is said to be safe if \( k = 1 \). A marking \( m_0 \) is said to be live for a Petri net if every marking has been reached from \( m_0 \), which indicates it is possible to fire any transition of the net by some firing sequence [12, 13]. If \( m_0 \) may be reached from any marking, The Petri net is said to be reversible.

To simulate the behavior of LD, this approach changes a state or marking according to defined firing rules for the Boolean Petri nets model.

### 2.2 State equation

The firing definition easily shows that the token moves from state \( M_{k-1} \) to another state \( M_k \) by the \( k \)th firing, and \( U_k \) is a firing vector which can be given in terms of the following matrix state equation for Petri nets [12]

\[
M_k = M_{k-1} + A^T U_k
\]  

where \( U_k \) is called firing vector, and \( A^T \) is called the incidence matrix for any given topological structure of Petri nets, defined by

\[
A^T(p_i, t_j) = \begin{cases} 
-A_o(p_i, t_j) \\
0 \\
A_i(t_j, p_i)
\end{cases}, \quad \text{where } 1 \leq i \leq m, \quad 1 \leq j \leq n.
\]  

\[\leq i \leq m, \quad 1 \leq j \leq n.\]  

\[\leq i \leq m, \quad 1 \leq j \leq n.\]

---

**Fig. 3.** A simple example.

**Fig. 4.** The reduction result of Fig. 3.

**Fig. 5.** (a) A simple example and (b) The composite result of Fig. 5a.

Note that \( M_k \) must be a vector of nonnegative integers [12]. The firing vector will then select an appropriate column of \( A^T \) such that

\[
M_{k-1} + A^T U_k \geq 0 \quad \text{for each } k
\]  

\[\leq i \leq m, \quad 1 \leq j \leq n.\]  

\[\leq i \leq m, \quad 1 \leq j \leq n.\]

### 2.3 Definition of action dominance and equivalence

**Dominance.** An action \( p_1 \) is said to dominate another action \( p_2 \) in an irredundant place iff every exist of token for \( p_2 \) is also exist of token for \( p_1 \), i.e., the life of a token of \( p_1 \) is longer than \( p_2 \), denoted as \( m(p_1) > m(p_2) \). The reduction of the place \( p_1 \) to be analyzed is based on the dominance relation.

**Example 1.** Fig. 3 shows a PN in which \( p_1 \) is dominated by \( p_2 \) and \( p_3 \), i.e., \( m(p_1) > m(p_2) \) and \( m(p_1) > m(p_3) \). Fig. 4 shows the reduction result.

**Equivalence.** The actions \( p_1 \) and \( p_2 \) are equivalent if exist of token is same condition for \( p_1 \) and \( p_2 \), i.e., \( m(p_1) > m(p_2) \) and \( m(p_2) > m(p_1) \). The composite of the place \( p_1 \) and \( p_2 \) to be analyzed is based on the equivalent action.

**Example 2.** Fig. 5a shows a PN in which \( p_2 \) is equivalent to \( p_3 \), i.e., \( m(p_3) > m(p_2) \) and \( m(p_2) > m(p_3) \). Fig. 5(b) shows the composite result.
III. LADDER DIAGRAM MODEL USING BOOLEAN PETRI NET

3.1 Model of basic modules

In ladder diagrams, the horizontal line (rung) and the associated elements represent Boolean equations. Similarly, in Boolean Petri nets, the associated transitions represent Boolean equations. In ladder diagrams, the symbol “○” represents the dependent element of the equation (coil). Similarly, in Boolean Petri nets, the symbol “○” represents the dependent element of the equation (place). In ladder diagrams, “|” represents the independent element (normal open contacts), while in Boolean Petri nets, “|” represents the independent element (input transitions). A diagonal line placed in the middle of these symbols (i.e., “|/|”) represents normal closed contacts, which indicate that the negated value of the variable is used. Similarly, bar “|” represents the output transition. In ladder diagrams, variables (contacts) placed in a series represent the AND Boolean function, while contacts placed in parallel represent the OR Boolean function. The rungs are executed in order from top to bottom. Therefore, the Type 8 ladder diagram in Table 1 represents Boolean equations $M = (A + M)B$ and $N = M$ [14]. In Boolean Petri nets, a similar input transition represents $(A + M)$, denoted as $t_1: (A + M)$, which is a composite transition. Conversely, output transitions represent $B$, denoted as $t_2: (B)$, and output places are denoted as $p_2: (M, N)$ which are composite places. Finally, Table I summarizes some typical LD modules and their corresponding Boolean Petri net models, where $S$ is a pseudo source and the composite and decomposite of Boolean Petri net are as shown in Table II.

3.2 Model of faulty ladder diagram

An LD circuit fault may generally be classed as both stuck-at 0 (s-a-0) and stuck-at 1 (s-a-1) type; the stuck-at 0 fault is like a NO switch and the stuck-at 1 fault is like an NC switch. Hence the fault model of the ladder diagram can be modeled as a Petri net. An LD of the possessed example fault is illustrated in Fig. 6a and the Petri net model is illustrated in Fig. 6b. Where fault $f_1$ are represented s-a-0 to represent the switch $A$ is stuck at open, fault $f_2$ is represented s-a-1 to represent the switch $B$ is stuck at close. According to Eq. (3), the incidence matrix is

$$A^T = \begin{bmatrix} p_1 & t_1 & p_2 & t_2 \\ -1 & 1 & 1 & -1 \end{bmatrix}.$$  

The faults classified in the two cases are interpreted as below.

Case 1. Assume $f_1$ is s-a-0 fault and initial marking is

$$M_0 = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

When the bottom is pushed $A = 1$.

$$U_1 = t_1: (A) \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad U_{f_1} = t_1: (A \cdot f_1) \begin{bmatrix} 0 \\ 0 \end{bmatrix}.$$  

$$U_{1/f_1} = t_1: (A/A \cdot f_1) \begin{bmatrix} 1/0 \\ 0 \end{bmatrix},$$

where $U_1$ and $U_{f_1}$ are represented as the fault free and faulty firing vector, respectively. $U_{1/f_1}$ is represented as the fault free/faulty firing vector.

According to Eq. (2)

$$M_{1/f_1} = M_0 + A^T U_{1/f_1} = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

$$+ \begin{bmatrix} t_1 & t_2 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 1/0 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 1 \\ 0 \end{bmatrix} + \begin{bmatrix} -1/0 \\ 1/0 \end{bmatrix} = \begin{bmatrix} 0/1 \\ 1/0 \end{bmatrix},$$

i.e., $M_1 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$, $M_{f_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$, where $M_1$ and $M_{f_1}$ are represented as the fault free and faulty marking vector, respectively. $M_{1/f_1}$ is represented as the fault free/faulty marking vector.

In the LD circuit, $f_1$ fault means the coil $C$ is not active since the switch $A$ is stuck at open.

Case 2. Assume $f_2$ is s-a-1 fault and current marking is

$$M_1 = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix}$$

When the bottom is pushed $B = 1$.

$$U_2 = \begin{bmatrix} t_1 \\ t_2: (B/B \cdot \bar{f}_2) \end{bmatrix} \begin{bmatrix} 0 \\ 1/0 \end{bmatrix}$$

© 2010 John Wiley and Sons Asia Pte Ltd and Chinese Automatic Control Society
Table II. Composite and Decomposite of Boolean Petri nets.

<table>
<thead>
<tr>
<th>LD</th>
<th>Boolean Equation</th>
<th>Decomposite</th>
<th>Composite</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>$C = A$</td>
<td>$S \rightarrow A$</td>
<td>($C,D$)</td>
</tr>
<tr>
<td>2.</td>
<td>$D = A$</td>
<td>$S \rightarrow D$</td>
<td>($D)$</td>
</tr>
<tr>
<td>3.</td>
<td>$C = D$</td>
<td>$S \rightarrow C$</td>
<td>($A,B$)</td>
</tr>
<tr>
<td>4.</td>
<td>$C = A$</td>
<td>$S \rightarrow C$</td>
<td>($A+B$)</td>
</tr>
<tr>
<td>5.</td>
<td>$C = A \bar{B}$</td>
<td>$S \rightarrow C$</td>
<td>($A \bar{B}$)</td>
</tr>
<tr>
<td>6.</td>
<td>$D = C$</td>
<td>$S \rightarrow D$</td>
<td>($D$)</td>
</tr>
</tbody>
</table>

Fig. 6. (a) An LD of possessed fault and (b) A Petri net model of possessed fault.

IV. APPLICATION EXAMPLE

This section illustrates a practical example of hierarchical control system in Fig. 1. The local controller is an LD circuit. This circuit can be modeled by BPN and is simplified to obtain an abstract model using Table I of the preceding section. The system fault can be diagnosed by the difference between the LD response and abstract model response. The differences are as decision of supervisor agent.

4.1 Constructing an abstract model using a Boolean PN

To start a three-phase motor, an LD controller uses type of Y-Δ starting to limit starting current, as shown in Fig. 7 and symbol descriptions in Table III. In the LD controller, the bottom $Pb_1$ is control relay coil $M$, $Y$ and timer coil active. The motor enters the starting state when NO contacts of $M$ and $Y$ are turned on. Next, the relay coil $Y$ turns off after delay time $T_\Delta$, and the motor returns to the normal state when the relay coil $Y$ turns off and relay coil $D$ turns on. Finally, the motor stops if the bottom $Pb_2$ is pushed or the current is overload. This LD controller can be specified as follows:

Step (1) The motor is commanded to start ($Pb_1$).
Step (2) The motor starting time is $T_\Delta$.
Step (3) The motor is commanded to stop ($Pb_2$).
Step (4) The motor will stop if the current is overloaded.

\[ M_{2/f_2} = M_1 + A^T U_2 = \begin{bmatrix} p_1 \\ p_2 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \end{bmatrix} \]

\[ \begin{bmatrix} t_1 & t_2 \\ -1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} 0 \\ 1/0 \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \]

\[ \begin{bmatrix} 1/0 \\ -1/0 \end{bmatrix} = \begin{bmatrix} 1/0 \\ 0/1 \end{bmatrix} \]

i.e., $M_2 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$; $M_{f_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$.

In the LD circuit, $f_2$ fault means the coil $C$ maintain action since the switch $B$ is stuck at close.
The implicit specification is as following:
Spec) The relay coil $D$ and relay coil $Y$ are mutually exclusive.

The transformation from the ladder diagram (in parallel) to the abstract model (in series) is based on the following steps:

Step (1) A rung or compound rung of LD is converted to a Boolean Petri net module using Table I or the Boolean equation. LD controller then assembles Boolean Petri net modules, as Fig. 8 shows, where (1), (2) ... and (9) correspond to the number of LD rungs.

Step (2) A Boolean Petri net can be given after eliminating the redundant or pseudo places (i.e., the $S$ place), as Fig. 9 illustrates.

Step (3) An abstract model can be obtained according to dominance relation reduce some places (in this case, an abstract model reduce place $p_2$), and eliminating some redundant elements (i.e., the coil of time or auxiliary relay), as Fig. 10 shows.

### Table III. The Descriptions of Symbol

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PL1</td>
<td>Indicator light of green</td>
<td>a</td>
<td>“b” contact of Push bottom</td>
</tr>
<tr>
<td>PL2</td>
<td>Indicator light of yellow</td>
<td>b</td>
<td>“a” contact of Push bottom</td>
</tr>
<tr>
<td>PL3</td>
<td>Indicator light of red</td>
<td>c</td>
<td>“a” contact of relay</td>
</tr>
<tr>
<td></td>
<td>Relay</td>
<td>d</td>
<td>“b” contact of relay</td>
</tr>
<tr>
<td></td>
<td>Timer</td>
<td>e</td>
<td>“a” contact of timer</td>
</tr>
<tr>
<td></td>
<td>Stuck at 0 (s-a-0) switch</td>
<td>f</td>
<td>“b” contact of timer</td>
</tr>
<tr>
<td>1~9</td>
<td>Rung number</td>
<td>g</td>
<td>“b” contact of over load</td>
</tr>
</tbody>
</table>

4.2 Properties of the proposed Boolean Petri net

The reachability of a PN is a tree, which uses states as nodes and transitions as arcs [16]. The construction of this tree starts from the root node. The root node is represented as the initial state, and the arcs outgoing from the root node are marked by the corresponding enabled
transition. The arc will outgo to a new node (state) from the firing of the corresponding transition (arc). The above procedures are repeated until they produce duplicate nodes. Terminal nodes are identical to existing nodes, which have no any enable transitions are met. In the reachability tree, dash lines indicate nodes.

Due to the similar processes of PN reachability tree, this study presents the reachability tree of the proposed BPN in Fig. 9. For the sake of simplicity in representing the node (node) in the reachability tree, define the state variable vector in the reachability tree as \([p_1 \ p_2 \ p_3 \ p_4]\), and allow the initial state to be \([1 \ 0 \ 0 \ 0]\). If transition \(t_1 : (Pb_1)\) fires when \(Pb_1 = 1\) (i.e. \(Pb_1\) is active), the state moves to \([0 \ 1 \ 1 \ 0]\). If transition \(t_3 : (Pb_2)\) fires when \(Pb_2 = 1\) (i.e. \(Pb_2\) is active), then the state moves to \([1 \ 0 \ 0 \ 0]\). Subsequently, if transition \(t_2 : (T_\Delta)\) is enabled and fires, then the state moves to \([0 \ 1 \ 0 \ 1]\), while if transition \(t_{32} : (Pb_2)\) is enabled and fires, the state moves to \([1 \ 0 \ 0 \ 0]\).

**Proposition 1.** The proposed BPN is live.

**Proof.** Consider a case based on the reachability tree in Fig. 11. This figure shows that there is no terminal node. Therefore, there always exist some sample path such that any transitions can eventually fire to reach any states from the initial state \(p_1\), i.e. \(R(m_0) = \{p_2, p_3, p_4\}\), \(F(m_0) = \{t_1, t_2\}\). According to this definition, the proposed BPN is live.

**Proposition 2.** The proposed BPN is reversible.

**Proof.** The reachability tree in Fig. 11 indicates that there is no terminal node. Therefore, there always exist some sample path such that any transitions can eventually fire to reach the initial state \(p_1\) from any states (i.e. \(p_2, p_3, p_4\)), \(p_1 \in R(p_2), p_1 \in R(p_3), p_1 \in R(p_4)\). According to this definition, the proposed BPN is reversible.

**Proposition 3.** The proposed BPN is bound.

**Proof.** In a stable PN, the number of tokens in any place will not grow infinitely. The reachability tree in Fig. 11 indicates that one and only one marked token corresponds to any specific state. Therefore, the number of marked tokens in \(p_1, p_2, p_3\) and \(p_4\) is bounded above by 1. According to this definition, the proposed BPN is bounded and safe.

Similarly, the proposed abstract model in Fig. 10 is live, reversible, and safe. In the Petri net model, the live is represented as a reachable starting state (i.e. \(Y\) state) and running state (i.e. \(D\) state) from the ideal state (i.e. \(P_L\) state), the safe is represented as only existing in one state, the reversible is represented as returning to the ideal state (i.e. \(P_L\) state) from any other state (i.e. \(Y\) state and \(D\) state).

### 4.3 State equation

According to Eq. (2), Fig. 9 shows that the state equation is \(M_k = M_{k-1} + A^T U_k\)

\[
A^T = \begin{bmatrix}
  t_1 & t_2 & t_3 & t_4 \\
  -1 & 0 & 1 & 1 \\
  1 & 0 & -1 & -1 \\
  1 & -1 & -1 & 0 \\
  0 & 1 & 0 & -1
\end{bmatrix}
\]

\[
p_1 = \begin{bmatrix}
  p_1 \\
  0 \\
  0 \\
  0
\end{bmatrix}
\]

\[
p_2 = \begin{bmatrix}
  p_2 \\
  1 \\
  1 \\
  0
\end{bmatrix}
\]

\[
p_3 = \begin{bmatrix}
  p_3 \\
  0 \\
  0 \\
  1
\end{bmatrix}
\]

\[
p_4 = \begin{bmatrix}
  p_4 \\
  0 \\
  0 \\
  1
\end{bmatrix}
\]
In reality, places $p_2: (X, M, Timer)$, $p_3: (Y, PL_2)$, and $p_1: (D, PL_3)$ are compounded places in Petri nets. Therefore, the state of $P_1 = \{ p_1, p_2, p_3, p_4 \}$ can be decomposed into the $P = \{ p_{L_1}, X, M, Timer, Y, PL_2, D, PL_3 \}$ state, so $M_0^T(P) = [1 \ 0 \ 0 \ 0]$ can be transferred into $M_0^T(P) = [1 \ (0 \ 0 \ 0) \ (0 \ 0) \ (0 \ 0) \ (0 \ 0)]$. Similarly, $M_1^T(P), M_2^T(P), M_3^T(P)$ and $M_4^T(P)$ can be decomposed into places $[0 \ (1 \ 1 \ 1) \ (1 \ 1) \ (0 \ 0)], [0 \ (1 \ 1 \ 1) \ (0 \ 0) \ (1 \ 1)], [1 \ (0 \ 0 \ 0) \ (0 \ 0) \ (0 \ 0)]$ and $[1 \ (0 \ 0 \ 0) \ (0 \ 0) \ (0 \ 0)]$, respectively.
4.4 Analysis and diagnosis of fault modeling

Assume that the faults \( f_1 \), \( f_2 \) and \( f_3 \) in the LD are stuck at 0 (s-a-0), as Fig. 7 illustrates. The fault can then be modeled into Petri nets as shown in Fig. 12(a), (b), and (c), respectively. In case 1, a transition \( t_1 : (pb_1) \) is fired since \( pb_1 \) is active. However, the transition \( t_f : \) \((pb_1 f_1) = 0 \) cannot be fired since \( f_1 \) is stuck at 0. Similarly, in case 2, \( t_f : (X_2 f_2) = 0 \), \( t : (X_2) = 1 \). In case 3, \( t_f : (D_2 f_3) = 0 \), \( t : (D_2) = 1 \). For simple calculation of state equation, a control vector \( U_k \) contains the fault free Boolean equation and faulty Boolean equation of transitions, as denoted by \( t_f : \) (fault free/fault) = 1/0. Fig. 13 shows fault free model and fault model simulated structure.

A difference output vector (DOV) = fault free output vector-fault output vector. If has fault occur then difference output vector \( \neq 0 \). The fault is covered area from place of negative value to place of positive value, and the faulty path flows through transition \( t_i \) in DOV.

**Case 1.** Assume \( f_1 \) is s-a-0.

\[
\begin{align*}
A_{f_1}^T &= \begin{bmatrix}
-1 & 0 & 1 & 1 \\
1 & 0 & -1 & -1 \\
1 & -1 & -1 & 0 \\
0 & 1 & 0 & -1
\end{bmatrix} \\
&= \begin{bmatrix}
1 & 0 & 1 & 1 \\
-1 & 0 & 1 & 1
\end{bmatrix}^{T} + \begin{bmatrix}
1 & 0 & -1 & -1 \\
0 & 1 & -1 & 0 \\
1 & 1 & 0 & -1
\end{bmatrix} = \begin{bmatrix}
0/1 \\
0/0 \\
0/1
\end{bmatrix}
\end{align*}
\]

Thus the fault is located from \( p_1 \) to \( p_2 \) and \( p_3 \) as Fig. 14 indicates, and the fault path flows through \( t_1 : (pb_1) \).
between rung 1 and rung 3 in Fig. 7. In physical terms, this means the motor cannot start rotation.

**Case 2.** Assume \( f_2 \) is s-a-0.

\[
\begin{align*}
\begin{bmatrix} t_1 \\ t_2 \end{bmatrix} &= \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}, \quad \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \\
A_{f_2}^T &= p_2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \\
M_0 &= p_2 \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \\
U_1 &= \begin{bmatrix} t_1 : (Pb_1) \\ t_2 : (X_2) \end{bmatrix}, \\
M_1(f_2) &= M_0 + A_{f_2}^T U_1 = p_2 \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \\
M_2(f_2) &= M_1 + A_{f_2}^T U_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}.
\end{align*}
\]

DOV = [0 0 1]^T. The faulty area is covered only in \( p_{f_2} \) as Fig. 12b indicates, and the fault path flows through \( t_{f_2} : (X_2) \). Thus, the fault is located between rung 2 and rung 5 in Fig. 7. In physical terms, this means the motor cannot run.

**Case 3.** Assume \( f_3 \) is s-a-0.

\[
\begin{align*}
M_0 &= p_3 \begin{bmatrix} 0 \\ 1 \end{bmatrix}; \\
M_1 &= M_0 + A_{f_3}^T U_1,
\end{align*}
\]
V. CONCLUSIONS

This paper proposes the Boolean Petri net using the Boolean equation, constructs a ladder diagram module using the Petri net and develops an abstract model using Petri nets to diagnose local faults in the LD. The diagnostic process employs simple matrix manipulation and DOV to determine the faulty area for diagnosing the ladder diagram. This study also provides an example using composite transition, composite place, and relevant state to reduce complexity and increase readability of the Petri nets. The proposed methodology is useful and clear.

REFERENCES


**Jui-I Tsai** was born in Yunlin, Taiwan, in 1958. He received his B.S. degree in 1985 from the Department of Electronic Engineering, Feng Chia University, Taichung, and M.S. degree in 1994 from the Institute of Medical Engineering, National Cheng Kung University, Taiwan. He is a Ph.D. candidate at Department of Electric Engineering, National Chiao Tung University, Hsinchu. His research interests include Petri nets and IC testing.

**Ching-Cheng Teng** was born in Taiwan, in 1938. He received the B.S. degree in Electric Engineering from the Nation Cheng Kung University, Taiwan, in 1961. From 1991 to 1998, he was the chairman of the Department of Electrical and Control Engineering, Nation Chiao Tung University, Hsinchu, Taiwan. He is currently a professor in the Department of Electric Engineering at the same university. His research interests include $H_\infty$ optimal control, signal processing, and fuzzy neural systems.