State Equation of Program Nets and Its Application to Reachability Analysis for SWITCH-less Nets

Shingo Yamaguchi¹, Kousuke Yamada¹, Qi-Wei Ge‡ and Minoru Tanaka¹

¹Faculty of Engineering, Yamaguchi University, 2-16-1 Tokiwadai, Ube, 755-8611, Japan
‡Faculty of Education, Yamaguchi University, 1677-1 Yoshida, Yamaguchi, 753-8513, Japan

Abstract: This paper proposes a state equation of program nets and its application to reachability analysis. Many of previous studies have described and analyzed the dynamic behavior of program nets not by equations but graphs. We first define a matrix representation of program nets, and then propose a state equation of program nets using the matrix representation. The state equation of program nets can precisely represent the state of program nets. Furthermore, we apply the state equation to the analysis of an important property of program nets, called reachability. As a result, a necessary condition for reachability is proposed for program nets with no SWITCH-node.

Keywords: dataflow program, program net, matrix representation, state equation, reachability

1. Introduction

A dataflow computer [1, 2] is a multi-processor system based on the principle of data-driven processing, and its programs are usually represented as dataflow program nets (program nets or nets for short). A program net [3, 4] is a graph representation of a dataflow program consisting of three types of nodes: AND-node, OR-node, and SWITCH-node, which represent arithmetic/logical, data merge, and context switch operations, respectively.

The dynamic behavior of many systems studied in engineering can be described by differential equations or algebraic equations [5]. It would be nice if we could describe and analyze the dynamic behavior of program nets by some equations. However, many of previous studies have described and analyzed the dynamic behavior of program nets not by equations but graphs. In Ref. [6], Ono et al. have tried to describe and analyze the dynamic behavior of program nets by converting a program net into a Petri net [5] and using a state equation of the Petri net. As an example, let us compare the program-nets-based description of an OR-node with the Petri-nets-based description. The program-nets-based description is shown in Fig. 1 (a) and the Petri-nets-based description is shown in Fig. 1 (b). In the program-nets-based description, tokens of the two input edges of the OR-node are represented separately. However, in the Petri-nets-based description, they are represented in one place unitedly. Therefore, the Petri-nets-based state equation of a program net is not sufficient to clearly describe and analyze the dynamic behavior of the program net. Thus, it is necessary to develop a state equation of program nets which can precisely represent the state of the program nets.

Program nets have been making contribution not only to the description of dataflow programs but also to the analysis of its various properties. Ge et al. have proposed graph-based verification methods for several properties: execution termination, computation determinacy [4], and token self-cleanness [7, 8]. In this paper, we deal with a new property, called reachability, for program nets. In a program net, a certain state is said to be reachable from an initial state if there exists a firing sequence of nodes for leading to the state from the initial state. Reachability is an important property for studying the dynamic behavior of dataflow programs, because any dataflow program must be reachable to its final states. However, no analysis method for reachability has been proposed till now.

![Diagram](image)

(a) program-nets-based description

(b) Petri-nets-based description

Figure 1: Program-nets-based and Petri-nets-based descriptions of an OR-node.
In this paper, we propose a state equation of program nets and its application to reachability analysis. We first define a matrix representation of program nets. We then propose a state equation of the program nets using the matrix representation. Finally, for program nets with no SWITCH-node, we give a necessary condition for reachability by applying the state equation.

2. Preliminary

This section presents the formal definition and some properties of program nets. The formal definition of program nets is given as follows:

**Definition 1:** A program net is a four-tuple $PN=(N, E, \alpha, \beta)$, where

(i) $N$ is a set of nodes; a node $z\in N$ is one of three types: AND-node ($\bigcirc$), OR-node ($\bigtriangleup$), and SWITCH-node ($\bigtriangledown$),

(ii) $E$ is a set of directed edges (edges for short); an edge $e\in E$ is one of two types: dataflow-edge ($\rightarrow$) and controlflow-edge ($\rightarrow\rightarrow$),

(iii) $\alpha$ and $\beta$ are sets of token thresholds of edges; token thresholds $\alpha_e$ and $\beta_e$ of an edge $e$ represent the number of tokens removed from $e$ and added on $e$ by a firing of the output and input node of $e$, respectively. □

Figure 2 shows an example of a program net. A token ($\bullet$) represents a single datum and is assigned to an edge of a program net. A **token distribution** $d(\tau)=(d_{e_1}(\tau), d_{e_2}(\tau), \cdots, d_{e_{|E|}}(\tau))^T$ represents a state of the program net whose $i$-th entry expresses the number of tokens on an edge $e_i$ at time $\tau$. The token distribution is changed according to the following (node) firing rules:

(i) An AND-node is fireable if each of its input edge $e_{in}$ has at least $\alpha_{e_{in}}$ tokens. A firing of a fireable AND-node removes $\alpha_{e_{in}}$ tokens from each of its input edge $e_{in}$ and adds $\beta_{e_{out}}$ tokens on each of its output edge $e_{out}$.

(ii) An OR-node is fireable if anyone of its input edges, say $e_{in}$, has at least $\alpha_{e_{in}}$ tokens. A firing of a fireable OR-node removes $\alpha_{e_{in}}$ tokens from the input edge $e_{in}$ and adds $\beta_{e_{out}}$ tokens on each of its output edge $e_{out}$.

(iii) A SWITCH-node has two input edges: one dataflow-edge and one controlflow-edge, and two output edges labeled as True or False. A SWITCH-node is fireable if its input dataflow-edge $e_{in}$ has at least $\alpha_{e_{in}}$ tokens and controlflow-edge have at least one token, respectively. A firing of a fireable SWITCH-node removes $\alpha_{e_{in}}$ tokens from its input dataflow-edge and one token from its controlflow-edge, respectively, and if the value of the token removed from the controlflow-edge is True (or False) then $\beta_{e_{True}}$ ($\beta_{e_{False}}$) tokens are added on its output edge $e_{True}$ ($e_{False}$) labeled as True (or False).

In this paper, program nets are assumed to satisfy the following conditions:

(i) A program net has a single source AND-node, called start node $s$, and a single sink node, called termination node $t$. Every node is located on a path from $s$ to $t$ and has at most two input edges and two output edges.

(ii) For every edge $e$, its token thresholds $\alpha_e$ and $\beta_e$ are $1$, i.e. $\forall e\in E, \alpha_e=\beta_e=1$.

(iii) Each node can fire at time epoch $0, 1, 2, \cdots$, and especially start node $s$ fires only once at time epoch $0$. Two or more nodes may fire at the same time. A single firing of a node takes 1 unit time.

(iv) The initial token distribution is $\mathbf{0}$.

Conditions (i) and (ii) are reasonable, because any program net can be transformed into the program net which satisfies the conditions. Due to limitation of this paper, the transforming method is omitted. Under the above conditions (iii) and (iv), without loss of generality, we can add the input edge $e_0$ of start node $s$ to a program net and give initial token distribution as $d(0)=(d_{e_0}(0), d_{e_1}(0), \cdots, d_{e_{|E|}}(0))^T=(1, 0, \cdots, 0)^T$. 

---

516
3. Matrix Representation of Program Nets

In this section, we propose a matrix representation of program nets.

**Definition 2:** A matrix representation of a program net is a four-tuple \((A, \Theta_{\text{AND}}, \Theta_{\text{OR}}, \Theta_{\text{SW}})\), where

(i) \(A\) is a \(|N| \times |E|\) matrix, called incidence matrix, whose \(ij\)-th entry \(a_{ij}\) is given by

\[
\begin{align*}
 a_{ij}^+ &= a_{ij}^+ - a_{ij}^- \\
 a_{ij}^- &= \begin{cases} 1 & \text{if } z_i \text{ has } e_j \text{ as its output edge}, \\ 0 & \text{otherwise}, \end{cases}\\
 a_{ij}^- &= \begin{cases} 1 & \text{if } z_i \text{ has } e_j \text{ as its input edge}, \\ 0 & \text{otherwise}, \end{cases}
\end{align*}
\]

[\(a_{ij}^+\)] and [\(a_{ij}^-\)] are called output incidence matrix \(A^+\) and input incidence matrix \(A^-\), respectively;

(ii) \(\Theta_{\text{AND}}\) is a \(|N| \times |N|\) diagonal matrix, called \(\text{AND-node discriminant matrix}\), whose \(ii\)-th entry \(\theta_{\text{AND}ii}\) is given by

\[
\theta_{\text{AND}ii} = \begin{cases} 1 & \text{if } z_i \text{ is an AND-node}, \\ 0 & \text{otherwise}; \end{cases}
\]

Similarly, \(\Theta_{\text{OR}}\) and \(\Theta_{\text{SW}}\) are \(|N| \times |N|\) diagonal matrices, called \(\text{OR-node discriminant matrix}\) and \(\text{SWITCH-node discriminant matrix}\), respectively, whose \(ii\)-th entries are given like \(\theta_{\text{AND}ii}\).

**Definition 3:** For a program net, the AND-node incidence matrix \(A_{\text{AND}}\) is a \(|N| \times |E|\) matrix, whose \(ij\)-th entry \(a_{\text{AND}ij}\) is given by

\[
\begin{align*}
 a_{\text{AND}ij}^+ &= a_{\text{AND}ij}^+ - a_{\text{AND}ij}^- \\
 a_{\text{AND}ij}^- &= \begin{cases} 1 & \text{if } z_i \text{ is an AND-node and has } e_j \text{ as its output edge}, \\ 0 & \text{otherwise}, \end{cases}\\
 a_{\text{AND}ij}^- &= \begin{cases} 1 & \text{if } z_i \text{ is an AND-node and has } e_j \text{ as its input edge}, \\ 0 & \text{otherwise}, \end{cases}
\end{align*}
\]

[\(a_{\text{AND}ij}^+\)] and [\(a_{\text{AND}ij}^-\)] are called AND-node output incidence matrix \(A_{\text{AND}}^+\) and AND-node input incidence matrix \(A_{\text{AND}}^-\), respectively;

Similarly, the OR-node incidence matrix \(A_{\text{OR}}\) and SWITCH-node incidence matrix \(A_{\text{SW}}\) are \(|N| \times |E|\) matrices whose \(ij\)-th entries \(a_{\text{OR}ij}^+ = a_{\text{OR}ij}^+ - a_{\text{OR}ij}^-\) and \(a_{\text{SW}ij}^+ = a_{\text{SW}ij}^+ - a_{\text{SW}ij}^-\) are given like \(a_{\text{AND}ij}^+\), \(a_{\text{AND}ij}^-\), \(a_{\text{AND}ij},\) \([a_{\text{OR}ij}^+], [a_{\text{OR}ij}^-], [a_{\text{SW}ij}^+], [a_{\text{SW}ij}^-]\) are called OR-node output incidence matrix \(A_{\text{OR}}^+\) and OR-node input incidence matrix \(A_{\text{OR}}^-\), SWITCH-node output incidence matrix \(A_{\text{SW}}^+\), and SWITCH-node input incidence matrix \(A_{\text{SW}}^-\), respectively.

From Definitions 2 and 3, we can conduct the following property.

**Property 1:** For AND-node output incidence matrix \(A_{\text{AND}}^+\) and AND-node input incidence matrix \(A_{\text{AND}}^-\), the following equations hold:

\[
A_{\text{AND}}^+ = \Theta_{\text{AND}} A^+ \\
A_{\text{AND}}^- = \Theta_{\text{AND}} A^-
\]

Similar equations hold for \(A_{\text{OR}}^+, A_{\text{OR}}^-, A_{\text{SW}}^+, A_{\text{SW}}^-\).

As an example, we give the matrix representation of the program net shown in Fig. 2 below.

\[
A = \begin{bmatrix}
 s & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
 z_1 & 0 & -1 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\
 z_2 & 0 & 0 & 0 & -1 & 0 & 1 & 1 & 0 & 0 \\
 z_3 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
 z_4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
 z_5 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
 t & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}
\]

\[
\Theta_{\text{AND}} = \text{diag}(1, 1, 0, 1, 0, 0, 1) \\
\Theta_{\text{OR}} = \text{diag}(0, 0, 0, 0, 1, 1, 0) \\
\Theta_{\text{SW}} = \text{diag}(0, 0, 1, 0, 0, 0, 0)
\]
4. State Equation of Program Nets

In this section, we propose a state equation of program nets using the matrix representation of program nets proposed above. We first give the definitions of firable vector and firing vector of program nets. Then we give a method of calculating firable vector. Finally, we present a state equation of program nets.

4.1 Firable Vector and Firing Vector

Intuitively, a firable vector of a program net shows which nodes are firable, and a firing vector of a program net shows which nodes are fired. We give the formal definitions of firable vector and firing vector as follows:

**Definition 4:** The firable vector $E(\tau)$ of a program net at time $\tau$ is a $|N|$-row vector whose $i$-th entry $e_i(\tau)$ is given by

$$e_i(\tau) = e_{AND_i}(\tau) + e_{OR_i}(\tau) + e_{SW_i}(\tau),$$

where $e_{AND_i}(\tau)$ is 1 if node $z_i$ is an AND-node firable at $\tau$, 0 otherwise, $e_{OR_i}(\tau)$ and $e_{SW_i}(\tau)$ are given like $e_{AND_i}(\tau)$.

$(e_{AND_1}(\tau), \cdots, e_{AND_{|N|}}(\tau))^T$, $(e_{OR_1}(\tau), \cdots, e_{OR_{|N|}}(\tau))^T$, and $(e_{SW_1}(\tau), \cdots, e_{SW_{|N|}}(\tau))^T$ are called AND-node firable vector $E_{AND}(\tau)$, OR-node firable vector $E_{OR}(\tau)$, and SWITCH-node firable vector $E_{SW}(\tau)$, respectively.

The firing vector $F(\tau)$ of a program net at time $\tau$ is a $|N|$-row vector whose $i$-th entry $f_i(\tau)$ is given by

$$f_i(\tau) = \begin{cases} 
1 & \text{if } z_i \text{ is fired at } \tau, \\
0 & \text{otherwise,} 
\end{cases}$$

where inequality $E(\tau) \geq F(\tau)$ must be satisfied.

We need the following operators to calculate firable vector $E(\tau)$ from token distribution $d(\tau)$.

**Definition 5:** Let $A$ be an $n$-column vector $(\lambda_1, \lambda_2, \cdots, \lambda_n)$, $\mu$ be an $n$-row vector $(\mu_1, \mu_2, \cdots, \mu_n)^T$, and $u(x)$ is unit step function, i.e. $u(x) = 1$ if $x \geq 0$; 0 otherwise.

(a) $\lambda \otimes \mu \overset{\text{def}}{=} \prod_{i=1}^{n} u(\mu_i - \lambda_i)$

(b) $u(\lambda) \overset{\text{def}}{=} (u(\lambda_1), u(\lambda_2), \cdots, u(\lambda_n))$

We can calculate $E(\tau) = E_{AND}(\tau) + E_{OR}(\tau) + E_{SW}(\tau)$ from $d(\tau)$ by using the following theorem.

**Theorem 1:** At any time $\tau$, the AND-node firable vector $E_{AND}(\tau)$, OR-node firable vector $E_{OR}(\tau)$, and SWITCH-node firable vector $E_{SW}(\tau)$ of a program net satisfy

(a) $E_{AND}(\tau) = A_{\overline{AND}} \otimes d(\tau)$;

(b) $E_{OR}(\tau) = u(A_{\overline{OR}} d(\tau) - 1_{|N| \times 1})$;

(c) $E_{SW}(\tau) = A_{\overline{SW}} \otimes d(\tau)$,

respectively, where $1_{|N| \times 1}$ is a $|N|$-row vector with all elements 1.

**Proof:** (i) We are firstly to prove part (a). An AND-node $z_i$ is firable iff $\forall e_j \in E, a_{\overline{AND}_{ij}} \leq d_{e_j}(\tau)$. This implies that the $i$-th entry $e_{AND_i}(\tau)$ of $E_{AND}(\tau)$ satisfies

$$e_{AND_i}(\tau) = \begin{cases} 1 & \text{if } \forall e_j \in E, a_{\overline{AND}_{ij}} \leq d_{e_j}(\tau), \\
0 & \text{otherwise.} \end{cases}$$

(1)

For the $i$-th column vector of $A_{\overline{AND}}$, the right side of the given equation is as follows:

$$(a_{\overline{AND}_i}, a_{\overline{AND}_i}, \cdots, a_{\overline{AND}_i \in E}) \otimes d(\tau) = \prod_{e_j \in E} u(d_{e_j}(\tau) - a_{\overline{AND}_i}) = Eq.(1)$$

Therefore $E_{AND}(\tau) = A_{\overline{AND}} \otimes d(\tau)$ holds.

- 518 -
(ii) We prove part (b). An OR-node $z_i$ is firable iff $\exists e_j \in \{e_{KL}, e_{KR}\}, a_{OR,i}^j \leq d_{ej}(\tau)$, where $e_{KL}$ and $e_{KR}$ be the left and right input edge of $z_i$. This implies that the $i$-th entry $\varepsilon_{ORi}(\tau)$ of $\varepsilon_{OR}(\tau)$ satisfies

$$
\varepsilon_{ORi}(\tau) = \begin{cases} 
1 & \text{if } \exists e_j \in \{e_{KL}, e_{KR}\}, a_{OR,i}^j \leq d_{ej}(\tau), \\
0 & \text{otherwise.}
\end{cases}
$$

(2)

For the $i$-th column vector of $A_{OR}^-$, the right side of the given equation is

$$
u((a_{OR,0}^i, a_{OR,1}^i, \cdots, a_{OR,E}^i) \mathbf{d}(\tau - 1) = \sum_{e_j \in E} a_{OR,i}^j d_{ej}(\tau) - 1 = \text{Eq.(2)}.
$$

Therefore $\varepsilon_{OR}(\tau) = u(A_{OR}^+ \mathbf{d}(\tau) - 1_{|N| \times 1})$ holds.

(iii) For part (c), a similar argument as part (a) gives the proof. Q.E.D.

4.2 State Equation of Program Nets

A token distribution $\mathbf{d}(\tau)$, i.e. a state, of a program net changes with firing of nodes. In this paper, since a firing of every node is supposed to take 1 unit time, the next state of $\mathbf{d}(\tau)$ is $\mathbf{d}(\tau + 1)$. We say a relation equation between $\mathbf{d}(\tau)$ and $\mathbf{d}(\tau + 1)$ as state transition equation of program nets. For any time $\tau_f$, we say a relation equation between $\mathbf{d}(0)$ and $\mathbf{d}(\tau_f)$ as state equation of program nets.

We need the following matrices $A_{ORL}^-$ and $A_{ORR}^-$ to give the definition of state (transition) equation of program nets.

**Definition 6:** $A_{ORL}^-$ is a $|N| \times |E|$ matrix, called OR-node left input incidence matrix, whose $ij$-th entry $a_{ORL,i}^j$ is given by

$$
a_{ORL,i}^j = \begin{cases} 
1 & \text{if } z_i \text{ is an OR-node and has } e_j \text{ as its left input edge,} \\
0 & \text{otherwise.}
\end{cases}
$$

$A_{ORR}^-$ is a $|N| \times |E|$ matrix, called OR-node right input incidence matrix, whose $ij$-th entry is given like $a_{ORL,i}^j$.

Furthermore, we need to use the following matrices $G(\tau)$ and $H(\tau)$ [9] to deal with non-deterministic behaviors of OR-nodes and SWITCH-nodes.

**Definition 7:** $G(\tau)$ is a $|E| \times |E|$ diagonal matrix whose $ii$-th entry $g_{ii}(\tau)$ is given by

$$
g_{ii}(\tau) = \begin{cases} 
1 & \text{if } e_i \text{ is an input edge of an OR-node whose firing removes one token from } e_i, \\
0 & \text{otherwise.}
\end{cases}
$$

**Proposition 1:** For a program net with $\mathbf{d}(\tau)$, we have

$$
G(\tau) = \text{diag}(u(\mathbf{d}_{ORL}(\tau) - 1) + \text{diag}(u(-A_{OR}^- (A_{OR}^-)^T A_{ORL}^- \mathbf{d}_{ORL}(\tau)))u(\mathbf{d}_{ORR}(\tau) - 1)),
$$

where $\mathbf{d}_{ORL}(\tau) = (A_{OR}^-)^T A_{ORL}^- \mathbf{d}(\tau)$, $\mathbf{d}_{ORR}(\tau) = (A_{OR}^-)^T A_{ORR}^- \mathbf{d}(\tau)$.

**Definition 8:** $H(\tau)$ is a $|E| \times |E|$ diagonal matrix whose $ii$-th entry $h_{ii}(\tau)$ is given by

$$
h_{ii}(\tau) = \begin{cases} 
1 & \text{if } e_i \text{ is an output edge of a SWITCH-node whose firing adds one token to } e_i, \\
0 & \text{otherwise.}
\end{cases}
$$

Using the above matrices, we have a state transition equation of program nets as follows:

**Lemma 1:** At any time $\tau = 0, 1, \cdots$, a program net satisfies

$$
\mathbf{d}(\tau + 1) = \mathbf{d}(\tau) + (A_{AND} + A_{OR}^+ - A_{OR}^- G(\tau) + A_{SW}^+ H(\tau) - A_{SW}^-)^T \mathcal{F}(\tau).
$$

(3)

**Proof:** Since a firing of every node is supposed to take 1 unit time, the next state of $\mathbf{d}(\tau)$ is $\mathbf{d}(\tau + 1)$. Let $\Delta \mathbf{d}^-(\tau)$ be the distribution of tokens removed from $\mathbf{d}(\tau)$, and $\Delta \mathbf{d}^+(\tau)$ be the distribution of tokens added to $\mathbf{d}(\tau)$, it is obvious that

$$
\mathbf{d}(\tau + 1) = \mathbf{d}(\tau) - \Delta \mathbf{d}^-(\tau) + \Delta \mathbf{d}^+(\tau).
$$

(4)
(I) We first prove that the following equation holds.

\[ \Delta d^- (\tau) = (A_{AND}^- + A_{OR}^- G(\tau) + A_{SW}^-)^T F(\tau) \]  

(5)

Let \( \Delta d^-_{AND} (\tau) \), \( \Delta d^-_{OR} (\tau) \), and \( \Delta d^-_{SW} (\tau) \) be the distributions of tokens removed from \( d(\tau) \) by firing of AND-nodes, OR-nodes, and SWITCH-nodes, respectively, we have \( \Delta d^- (\tau) = \Delta d^-_{AND} (\tau) + \Delta d^-_{OR} (\tau) + \Delta d^-_{SW} (\tau) \).

(i) We are to prove

\[ \Delta d^-_{AND} (\tau) = (A_{AND}^-)^T F(\tau). \]  

(6)

For an AND-node \( z_i \) and an edge \( e_j \), the \( ij \)-th entry \( a^-_{ANDij} \) of \( A_{AND}^- \) implies the number of tokens removed from \( e_j \) by a firing of \( z_i \). At a time \( \tau \), the number of tokens removed from \( e_j \) depends whether \( z_i \) is fired or not. Using the \( ij \)-th entry \( f_1(\tau) \) of \( F(\tau) \), the number of tokens removed from \( e_j \) is given as \( a^-_{ANDij} f_1(\tau) \). Therefore, let \( \Delta d^-_{ANDj} (\tau) \) be the \( j \)-th entry of \( \Delta d^-_{AND} (\tau) \), \( \Delta d^-_{ANDj} (\tau) = \sum_{0 \leq i \leq N} a^-_{ANDij} f_1(\tau) = (a^-_{AND1j}, a^-_{AND2j}, \ldots, a^-_{ANDNj})^T F(\tau) \). Thus, Eq.(6) holds.

(ii) Next we are to prove

\[ \Delta d^-_{OR} (\tau) = (A_{OR}^- G(\tau))^T F(\tau). \]  

(7)

For an OR-node \( z_i \) and an edge \( e_j \), the \( ij \)-th entry \( a^-_{ORij} \) of \( A_{OR}^- \) implies the number of tokens removed from \( e_j \) by a firing of \( z_i \). At a time \( \tau \), the \( jj \)-th entry \( g_{jj}(\tau) \) of \( G(\tau) \) shows whether tokens are actually removed from \( e_j \) or not. Furthermore, the number of tokens removed from \( e_j \) also depends whether \( z_i \) is fired or not. Using \( g_{jj}(\tau) \) and \( f_1(\tau) \), the number of tokens removed from \( e_j \) is given as \( a^-_{ORij} g_{jj}(\tau) f_1(\tau) \). Therefore, let \( \Delta d^-_{ORj} (\tau) \) be the \( j \)-th entry of \( \Delta d^-_{OR} (\tau) \), \( \Delta d^-_{ORj} (\tau) = \sum_{0 \leq i \leq N} a^-_{ORij} g_{jj}(\tau) f_1(\tau) = (a^-_{OR1j}, a^-_{OR2j}, \ldots, a^-_{ORNj})^T F(\tau) \). Thus, Eq.(7) holds.

(iii) Similarly as \( \Delta d^-_{AND} (\tau) \), we can prove

\[ \Delta d^-_{SW} (\tau) = (A_{SW}^-)^T F(\tau). \]  

(8)

(iv) From Eqs.(6)–(8), Eq.(5) holds.

(II) Next we are to prove

\[ \Delta d^+ (\tau) = (A^+_{AND} + A^+_{OR} + A^+_{SW} H(\tau))^T F(\tau). \]  

(9)

Let \( \Delta d^+_{AND} (\tau) \), \( \Delta d^+_{OR} (\tau) \), and \( \Delta d^+_{SW} (\tau) \) be the distributions of tokens added to \( d(\tau) \) by firing of AND-nodes, OR-nodes, and SWITCH-nodes, respectively, it is obvious that \( \Delta d^+ (\tau) = \Delta d^+_{AND} (\tau) + \Delta d^+_{OR} (\tau) + \Delta d^+_{SW} (\tau) \).

(i) We are to prove

\[ \Delta d^+_{AND} (\tau) = (A^+_{AND})^T F(\tau). \]  

(10)

For an AND-node \( z_i \) and an edge \( e_j \), the \( ij \)-th entry \( a^+_{ANDij} \) of \( A^+_{AND} \) implies the number of tokens added to \( e_j \) by a firing of \( z_i \). At a time \( \tau \), the number of tokens added to \( e_j \) depends whether \( z_i \) is fired or not. Using \( f_1(\tau) \), the number of tokens added to \( e_j \) is given as \( a^+_{ANDij} f_1(\tau) \). Therefore, let \( \Delta d^+_{ANDj} (\tau) \) be the \( j \)-th entry of \( \Delta d^+_{AND} (\tau) \), \( \Delta d^+_{ANDj} (\tau) = \sum_{0 \leq i \leq N} a^+_{ANDij} f_1(\tau) = (a^+_{AND1j}, a^+_{AND2j}, \ldots, a^+_{ANDNj})^T F(\tau) \). Thus, Eq.(10) holds.

(ii) Similarly as \( \Delta d^+_{AND} (\tau) \), we can prove

\[ \Delta d^+_{OR} (\tau) = (A^+_{OR})^T F(\tau). \]  

(11)

(iii) Next we are to prove

\[ \Delta d^+_{SW} (\tau) = (A^+_{SW} H(\tau))^T F(\tau). \]  

(12)

For a SWITCH-node \( z_i \) and an edge \( e_j \), the \( ij \)-th entry \( a^+_{SWij} \) of \( A^+_{SW} \) implies the number of tokens added to \( e_j \) by a firing of \( z_i \). At a time \( \tau \), the \( jj \)-th entry \( h_{jj}(\tau) \) of \( H(\tau) \) shows whether tokens are actually added to \( e_j \) or not. Furthermore, the number of tokens added to \( e_j \) also depends whether \( z_i \) is fired or not. Using \( h_{jj}(\tau) \) and \( f_1(\tau) \), the number of tokens added to \( e_j \) is given as \( a^+_{SWij} h_{jj}(\tau) f_1(\tau) \). Therefore, let \( \Delta d^+_{SWj} (\tau) \) be the \( j \)-th entry of \( \Delta d^+_{SW} (\tau) \), \( \Delta d^+_{SWj} (\tau) = \sum_{0 \leq i \leq N} a^+_{SWij} h_{jj}(\tau) f_1(\tau) = ((a^+_{SW1j}, a^+_{SW2j}, \ldots, a^+_{SWNj}))^T F(\tau) \). Thus, Eq.(12) holds.
(iv) From Eqs.(10)-(12), Eq.(9) holds.

(III) Using Eqs.(5),(9) and $\mathbf{A}_{AND} = \mathbf{A}_{AND}^+ - \mathbf{A}_{AND}^-$, the right side of Eq.(4) can be rewritten as

$$
d(\tau) + (\mathbf{A}_{AND} + \mathbf{A}_{OR}^+ - \mathbf{A}_{OR}^- \mathbf{G}(\tau) + \mathbf{A}_{SW}^+ \mathbf{H}(\tau) - \mathbf{A}_{SW}^-)^T \mathbf{F}(\tau).
$$

and thus this lemma holds.

From Lemma 1, we can obtain a state equation of program nets as follows:

**Theorem 2:** For any time $\tau_f (\geq 0)$, a program net satisfies

$$
d(\tau_f) = d(0) + \sum_{0 \leq \tau < \tau_f} (\mathbf{A}_{AND} + \mathbf{A}_{OR}^+ - \mathbf{A}_{OR}^- \mathbf{G}(\tau) + \mathbf{A}_{SW}^+ \mathbf{H}(\tau) - \mathbf{A}_{SW}^-)^T \mathbf{F}(\tau).
$$  \hspace{1cm} (13)

**Proof:** Writing the equation of Lemma 1 for $\tau = 0, 1, \ldots, \tau_f - 1$, i.e.

$$
d(1) = d(0) + (\mathbf{A}_{AND} + \mathbf{A}_{OR}^+ - \mathbf{A}_{OR}^- \mathbf{G}(0) + \mathbf{A}_{SW}^+ \mathbf{H}(0) - \mathbf{A}_{SW}^-)^T \mathbf{F}(0)
$$

$$
d(2) = d(1) + (\mathbf{A}_{AND} + \mathbf{A}_{OR}^+ - \mathbf{A}_{OR}^- \mathbf{G}(1) + \mathbf{A}_{SW}^+ \mathbf{H}(1) - \mathbf{A}_{SW}^-)^T \mathbf{F}(1)
$$

$$
\vdots
$$

$$
d(\tau_f) = d(\tau_f - 1) + (\mathbf{A}_{AND} + \mathbf{A}_{OR}^+ - \mathbf{A}_{OR}^- \mathbf{G}(\tau_f - 1) + \mathbf{A}_{SW}^+ \mathbf{H}(\tau_f - 1) - \mathbf{A}_{SW}^-)^T \mathbf{F}(\tau_f - 1)
$$

and adding them, we can obtain Eq.(13). Thus this theorem holds.

Q.E.D.

**Corollary 1:** For a SWITCH-less net, the followings hold.

(i) At any time $\tau = 0, 1, \cdots$, the net satisfies

$$
d(\tau + 1) = d(\tau) + (\mathbf{A}_{AND} + \mathbf{A}_{OR}^+ - \mathbf{A}_{OR}^- \mathbf{G}(\tau))^T \mathbf{F}(\tau);
$$  \hspace{1cm} (14)

(ii) For any time $\tau_f (\geq 0)$, the net satisfies

$$
d(\tau_f) = d(0) + \sum_{0 \leq \tau < \tau_f} (\mathbf{A}_{AND} + \mathbf{A}_{OR}^+ - \mathbf{A}_{OR}^- \mathbf{G}(\tau))^T \mathbf{F}(\tau).
$$  \hspace{1cm} (15)

Q.E.D.

As an application of using the state equation of program nets, for the program net shown in Fig. 2, we calculate $d(\tau_0 + 1)$ from $d(\tau_0) = (d_{e_0}(\tau_0), d_{e_1}(\tau_0), \cdots, d_{e_9}(\tau_0))^T = (0, 0, 0, 0, 1, 1, 0, 0, 0)^T$. We calculate the firable vector $\mathcal{E}(\tau_0)$ at time $\tau_0$ using Theorem 1.

$$
\mathcal{E}_{AND}(\tau_0) = \mathbf{A}_{AND}^- \otimes d(\tau_0) = (0, 0, 0, 1, 0, 0, 0)^T
$$

$$
\mathcal{E}_{OR}(\tau_0) = u((\mathbf{A}_{OR}^- d(\tau_0)) - 1) = (0, 0, 0, 0, 1, 0, 0)^T
$$

$$
\mathcal{E}_{SW}(\tau_0) = \mathbf{A}_{SW}^- \otimes d(\tau_0) = 0
$$

$$
\mathcal{E}(\tau_0) = \mathcal{E}_{AND}(\tau_0) + \mathcal{E}_{OR}(\tau_0) + \mathcal{E}_{SW}(\tau_0) = (0, 0, 0, 1, 1, 0, 0)^T
$$

We decide a firing vector $\mathcal{F}(\tau_0)$ based on $\mathcal{E}(\tau_0)$. In this example, let $\mathcal{F}(\tau_0) = \mathcal{E}(\tau_0)$. Note that two or more nodes may fire at the same time.

In order to use the state transition equation, we need calculate $\mathbf{G}(\tau_0)$ and $\mathbf{H}(\tau_0)$ using Proposition 1 and Definition 8.

$$
d_{ORI}(\tau_0) = (\mathbf{A}_{ORI}^-)^T \mathbf{A}_{ORI}^- d(\tau_0) = 0
$$

$$
d_{ORR}(\tau_0) = (\mathbf{A}_{ORR}^-)^T \mathbf{A}_{ORR}^- d(\tau_0) = (0, 0, 0, 0, 1, 0, 0, 0, 0)^T
$$

$$
\mathbf{G}(\tau_0) = \text{diag}(u(d_{ORI}(\tau_0) - 1) + \text{diag}(u'(\mathbf{A}_{OR}^-)^T \mathbf{A}_{OR}^- d_{ORI}(\tau_0)) u(d_{ORR}(\tau_0) - 1))
$$

$$
= \text{diag}(0, 0, 0, 0, 1, 0, 0, 0, 0)
$$

$$
\mathbf{H}(\tau_0) = 0
$$

Using the state transition equation, we can obtain $d(\tau_0 + 1)$ as follows:

$$
\Delta d^-(\tau_0) = (\mathbf{A}_{AND}^- + \mathbf{A}_{OR}^- \mathbf{G}(\tau_0) + \mathbf{A}_{SW}^-)^T \mathbf{F}(\tau_0) = (0, 0, 0, 0, 1, 0, 0, 0, 0)^T
$$

$$
\Delta d^+(\tau_0) = (\mathbf{A}_{AND}^+ + \mathbf{A}_{OR}^+ + \mathbf{A}_{SW}^+)^T \mathbf{F}(\tau_0) = (0, 0, 0, 0, 0, 0, 1, 1, 0)^T
$$

$$
d(\tau_0 + 1) = d(\tau_0) + \Delta d^-(\tau_0) + \Delta d^+(\tau_0) = (0, 0, 0, 0, 0, 0, 1, 1, 0)^T
$$

- 521 -
5. Application on Reachability Analysis

This section describes an application of the state equation to reachability analysis of SWITCH-less nets.

The formal definition of reachability is given in the following.

**Definition 9:** In a program net, a token distribution \( d \) is said to be **reachable** from the initial token distribution \( d(0) \) if there exists a firing sequence of nodes for leading to \( d \) from \( d(0) \).

The complexity of the reachability problem for SWITCH-less nets is NP-complete, because a SWITCH-less net is equivalent to a conflict-free Petri net [10] and the complexity of the reachability problem for conflict-free Petri nets is NP-complete [11].

We provide a method of analyzing reachable of program nets. The feature of the method is based on the state equation of program nets. From Corollary 1 (i), we can conduct the following Lemma.

**Lemma 2:** At any time \( \tau=0,1,\cdots \), a SWITCH-less net satisfies

\[
(I_{[1]}+\Omega)d(\tau+1) = (I_{[1]}+\Omega)d(\tau)+(A^+\Omega)^TF(\tau),
\]

where \( \Omega = (A^\text{ORL})^TA^\text{ORR}+(A^\text{ORR})^TA^\text{ORL} \).

**Proof:** Multiplying both sides of the equation in Corollary 1 (i) on the left by \( (I_{[1]}+\Omega) \), we can obtain

\[
(I_{[1]}+\Omega)d(\tau+1) = (I_{[1]}+\Omega)d(\tau)+(I_{[1]}+\Omega)(A_{\text{AND}}+A^+_\text{OR} - A^-\text{OR} G(\tau))^T F(\tau).
\]

Since \( (I_{[1]}+\Omega) \) is symmetric, i.e., \( (I_{[1]}+\Omega)^T=(I_{[1]}+\Omega) \), the transpose of \((I_{[1]}+\Omega)(A_{\text{AND}}+A^+_\text{OR} - A^-\text{OR} G(\tau))^T\) can be rewritten as follows:

\[
(A_{\text{AND}}+A^+_\text{OR} - A^-\text{OR} G(\tau))(I_{[1]}+\Omega) = (A^+ - A^-\text{AND} - A^-\text{OR} G(\tau))(I_{[1]}+\Omega)
\]

\[
= A^+ - A^-\text{AND} + A^+\Omega - A^-\text{OR} G(\tau)(I_{[1]}+\Omega)
\]

From \( A^-\text{AND} \Omega = 0 \) and \( A^-\text{OR} G(\tau)(I_{[1]}+\Omega) = A^-\text{OR} \) (the former is obvious because \( A^-\text{AND}(A^\text{ORL})^T = 0 \) and \( A^-\text{AND}(A^\text{ORR})^T = 0 \), the later is to be proved later), then

Eq.(18) \[ \Rightarrow \] \[ A^+ - A^-\text{AND} + A^+\Omega - A^-\text{OR} \]

\[ \Rightarrow \] \[ A^+ + \Omega \]

Therefore, the right side of Eq.(17) can be rewritten as \((I_{[1]}+\Omega)d(\tau)+(A^+\Omega)^TF(\tau)\) and thus Eq.(16) holds.

Finally, we are to prove \( A^-\text{OR} G(\tau)(I_{[1]}+\Omega) = A^-\text{OR} \). Let \( z_i \) be an OR-node, \( e_{kL} \) and \( e_{kR} \) be the left and right input edge of \( z_i \),

\[
A^-\text{OR} G(\tau)(I_{[1]}+\Omega)
\]

\[
= z_i \left[ \begin{array}{ccc} e_{kL} & e_{kR} & \cdots \vspace{0.3cm} \\
0 & 0 & 0 & \cdots \vspace{0.3cm} \\
0 & 0 & 0 & \cdots \end{array} \right] (I_{[1]}+\Omega)
\]

\[
= z_i \left[ \begin{array}{ccc} e_{kL} & e_{kR} & \cdots \vspace{0.3cm} \\
0 & 0 & 0 & \cdots \vspace{0.3cm} \\
0 & 0 & 0 & \cdots \end{array} \right] (I_{[1]}+\Omega)
\]

\[
= z_i \left[ \begin{array}{ccc} e_{kL} & e_{kR} & \cdots \vspace{0.3cm} \\
0 & 0 & 0 & \cdots \vspace{0.3cm} \\
0 & 0 & 0 & \cdots \end{array} \right] (I_{[1]}+\Omega)
\]
Recalling \( g_{kL}(\tau) + g_{kR}(\tau) = 1 \) from the definition of \( G(\tau) \), then
\[ \text{Eq.}(19) = A_{OR}^- \]

Therefore \( A_{OR}^- G(\tau)(I_{[E]} + \Omega) = A_{OR}^- \) holds. Thus this lemma holds.

Q.E.D.

From the above lemma, we can conduct the following theorem.

**Theorem 3:** For any time \( \tau_f \ (\geq 0) \), a SWITCH-less net satisfies
\[ (I_{[E]} + \Omega) d(\tau_f) = (I_{[E]} + \Omega) d(0) + (A + A^+ \Omega)^T \sum_{0 \leq \tau < \tau_f} F(\tau) \]
\[ \text{Eq.}(20) \]
where \( \Omega = (A_{ORL})^T A_{ORR}^- (A_{ORR})^T A_{ORL}^- \).

\[ \square \]

**Proof:** Writing the equation of Lemma 2 for \( \tau = 0, 1, \cdots, \tau_f - 1 \), i.e.
\[ (I_{[E]} + \Omega) d(1) = (I_{[E]} + \Omega) d(0) + (A + A^+ \Omega)^T F(0) \]
\[ (I_{[E]} + \Omega) d(2) = (I_{[E]} + \Omega) d(1) + (A + A^+ \Omega)^T F(1) \]
\[ \vdots \]
\[ (I_{[E]} + \Omega) d(\tau_f) = (I_{[E]} + \Omega) d(\tau_f - 1) + (A + A^+ \Omega)^T F(\tau_f - 1) \]
and adding them, we can obtain Eq.(20). Thus this theorem holds.

Q.E.D.

Eq.(20) can be rewritten as
\[ (A + A^+ \Omega)^T x = (I_{[E]} + \Omega) \Delta d \]
\[ \text{Eq.}(21) \]
where \( \Delta d = d(\tau_f) - d(0) \) and \( x = \sum_{0 \leq \tau < \tau_f} F(\tau) \). Here \( x \) is a \(|N|\)-row vector, called firing count vector, whose \( i \)-th entry \( x_i \) is the total firing number of the \( i \)-th node for leading to \( d(\tau_f) \) from \( d(0) \).

Since Eq.(21) is a set of linear algebraic equations, Eq.(21) has a solution \( x \) iff \( (I_{[E]} + \Omega) \Delta d \) is orthogonal to every solution \( y \) of its homogeneous system, \( (A + A^+ \Omega) y = 0 \). Let \( r \) be the rank of \( (A + A^+ \Omega) \), and partition \( (A + A^+ \Omega) \) into the following form:
\[ A + A^+ \Omega = \begin{bmatrix} [E]_{r-r} & 0 \\ 0 & [N]_{-r-r} \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \]
\[ \text{Eq.}(22) \]
where \( A_{12} \) is a nonsingular square matrix of order \( r \). A set of \( ([E]_{r-r}) \) linearly independent solutions for \( (A + A^+ \Omega) y = 0 \) can be given as the \( ([E]_{r-r}) \times [E] \) rows of the following \( ([E]_{r-r}) \times [E] \) matrix \( B_f \):
\[ B_f = [I_{[E]}_{r-r}] : -A_{11}^{-1} A_{12}^T A_{12}^{-1} \]
\[ \text{Eq.}(23) \]

Since \( (A + A^+ \Omega) B_f^T = 0 \), the condition that Eq.(21) has a solution \( x \) can be rewritten as \( B_f(I_{[E]} + \Omega) \Delta d = 0 \). Therefore, we have the following necessary condition for reachability in a SWITCH-less net.

**Theorem 4:** If a token distribution \( d \) is reachable from \( d(0) \) in a SWITCH-less net, then
\[ B_f(I_{[E]} + \Omega)(d - d(0)) = 0, \]
where \( B_f \) is given by Eq.(23).

However, it is difficult to give a sufficient condition of reachability because there is no efficient method of calculating an integer solution of the state equation.

The contrapositive of Theorem 4 provides the following sufficient condition for non-reachability.

**Corollary 2:** In a SWITCH-less net, a token distribution \( d \) is not reachable from \( d(0) \) \( (+d) \), if
\[ B_f(I_{[E]} + \Omega)(d - d(0)) \neq 0 \]
where \( B_f \) is given by Eq.(23).

\[ \square \]

We give an example of analyzing reachability by using our method. Figure 3 shows the SWITCH-less net used in this example. Consider an initial token distribution
\[ d(0) = (d_{e1}(0), d_{e2}(0), d_{e3}(0), d_{e4}(0), d_{e5}(0), d_{e6}(0), d_{e7}(0), d_{e8}(0), d_{e9}(0), d_{e10}(0), d_{e11}(0), d_{e12}(0))^T \]
\[ = (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0)^T \]
and two token distributions

\[ \mathbf{d}_1 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0) T \] \text{ and } \mathbf{d}_2 = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0) T.

\( \mathbf{d}_1 \) is reachable from \( \mathbf{d}(0) \), because there exists the following firing sequence:

\[ \begin{align*}
\mathbf{d}(0) &= (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0) T \\
\xrightarrow{e} & (0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0) T \xrightarrow{z_1} (0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0) T \\
\xrightarrow{z_2} & (0, 1, 0, 0, 0, 0, 0, 2, 1, 0, 0, 0) T \xrightarrow{z_3} (0, 0, 1, 0, 0, 0, 0, 0, 2, 1, 0, 0, 0) T \\
\xrightarrow{z_4} & (0, 0, 2, 0, 0, 0, 0, 0, 0, 0, 1, 1) T \xrightarrow{z_5} (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0) T \\
\xrightarrow{z_6} & (0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0) T \xrightarrow{z_7} (0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0) T \\
\xrightarrow{z_8} & (0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 0) T = \mathbf{d}_1.
\end{align*} \]

From the above firing sequence, we can obtain the following firing count vector \( \mathbf{x} \):

\[ \mathbf{x} = (1, 4, 1, 1, 2, 2, 4, 2, 4, 4) T. \]

On the other hand, \( \mathbf{d}_2 \) is not reachable from \( \mathbf{d}(0) \), because there exists no firing sequence of nodes for leading to \( \mathbf{d}_2 \) from \( \mathbf{d}(0) \).

For the SWITCH-less net, matrix \((\mathbf{A} + \mathbf{A}^+ \Omega)\) is

\[
\begin{bmatrix}
  e_3 & e_6 & e_{12} & e_0 & e_1 & e_2 & e_4 & e_5 & e_7 & e_9 & e_{10} & e_{11} & e_{13} \\
  s & 0 & 0 & 0 & -1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
  t & 0 & 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & -1 \\
  z_1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  z_2 & 1 & 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\
  z_3 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
  z_4 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
  z_5 & -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 \\
  z_6 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 0 \\
  z_7 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
  z_8 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1 & 0 & 0
\end{bmatrix}
\]
The matrix \((A + A^+\Omega)\) is of rank 10 and can be partitioned in the form of Eq. (22), where

\[
A_{11} = \begin{bmatrix}
0 & 0 & 0 & -1 \\
0 & 0 & -1 & 0 \\
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
-1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & -1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1
\end{bmatrix}
\]

and \(A_{12} = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & -1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1 & 1
\end{bmatrix}\).

Thus, the matrix \(B_f\) can be obtained by Eq. (23):

\[
B_f = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & -1 & 0 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & -1
\end{bmatrix}.
\]

Further, matrix \((I_{|E|} + \Omega)\) is

\[
\begin{bmatrix}
\epsilon_3 & \epsilon_6 & \epsilon_{12} & \epsilon_0 & \epsilon_1 & \epsilon_2 & \epsilon_4 & \epsilon_5 & \epsilon_9 & \epsilon_{10} & \epsilon_{11} & \epsilon_{13} \\
\epsilon_3 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\epsilon_6 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\epsilon_{12} & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\epsilon_0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\epsilon_1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
\epsilon_2 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\epsilon_4 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
\epsilon_5 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
\epsilon_7 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\
\epsilon_9 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\
\epsilon_8 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
\epsilon_{10} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\epsilon_{11} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\epsilon_{13} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}.
\]

For the reachable token distribution \(d_1\),

\[
B_f(I_{|E|} + \Omega)(d_1 - d(0)) = 0
\]

holds. On the other hand, for the token distribution \(d_2\),

\[
B_f(I_{|E|} + \Omega)(d_2 - d(0)) = (0, 0, 0, -1)^T \neq 0
\]

holds, and hence \(d_2\) is not reachable from \(d(0)\).

6. Conclusion

We have proposed a state equation of program nets and its application to reachability analysis for SWITCH-less nets.

Firstly we have proposed a matrix representation of program nets. The matrix representation of a program net can uniquely determine the structure of the program net. Further, it enables us to calculate the executable vector of a program net from its token distribution. Next we have proposed a state equation of program nets using the matrix representation. The state equation of program nets is more complicated than the state equation of Petri nets. This complexity results from non-deterministic behaviors of OR-nodes and SWITCH-nodes. Although the state equation is non-linear, we have given a linear variation of the state equation.
As an application of our proposed state equation, we have analyzed reachability of SWITCH-less nets by using the linear variation of the state equation. As a result, we gave a necessary condition of reachability for SWITCH-less nets. The contrapositive of the condition provides a sufficient verification condition of non-reachability. This sufficient condition is considered useful. For example, the final marking of an acyclic SWITCH-less net with token self-cleanness must be $0$. Therefore, for some acyclic SWITCH-less nets, if the token distribution $0$ is not reachable from the initial token distribution $d(0)$, then we can say the net is not token self-cleanness. As a future work, we are to develop verification methods for various properties of program nets by using the state equation.

References


Shingo Yamaguchi received the B.E., M.E., and D.E. degrees from Yamaguchi University, Japan, in 1992, 1994 and 2002, respectively. He is currently a Research Associate in the Faculty of Engineering, Yamaguchi University. His research interests are in the area of net theory and its applications.

Kousuke Yamada received the B.E. degree from Yamaguchi University, Japan in 2003. He is currently a graduate student of Graduate School of Yamaguchi University. His research interest includes program net theory.

Qi-Wei Ge received the B.E. from Fudan University, the People’s Republic of China, in 1983, M.E. and Ph.D. from Hiroshima University, Japan, in 1987 and 1991, respectively. He was with Fujitsu Ten Limited from 1991 to 1993. Since 1993 he has been with Yamaguchi University, Japan. Currently he is a Professor at Yamaguchi University, Japan. His research interest includes Petri nets, program net theory and combinatorics.

Minoru Tanaka received the B.E., M.E., and Ph.D. from Osaka University, Japan, in 1969, 1973, and 1976, respectively. He was a Research Associate in the Faculty of Engineering Science, Osaka University, Japan, and then he was an Associate Professor in the Faculty of Engineering, Hiroshima University, Japan. Since 1991, he has been a Professor in the Faculty of Engineering, Yamaguchi University, Japan. His research interests include knowledge engineering, pictorial information processing, and software engineering.