'92 韓國自動制御學術會議論文集 1992. 10. 19~21

이산현상 시스템을 위한 두개의 입력을 가진 모델

이 준 화 ◦ 권 욱 현

서울대학교 공과대학 제어계측공학과 제어계측신기술 연구센타

Two-Port Machine Model for Discrete Event Dynamic Systems

Joon Hwa Lee and Wook Hyun Kwon

Engineering Research Center for Advanced Contr. and Instr.
Seoul National Univ., San 56-1 Silim-dong Kwanak-gu, Seoul 151-742

Abstract

In this paper, a two ports machine(TPM) model for discrete event dynamic systems(DEDS) is proposed. The proposed model is a finite state machine which has two inputs and two outputs. Inputs and outputs have two components, events and informations. TPM is different from other state machine models, since TPM has symmetric input and output. This symmetry enables the block diagram representation of the DEDS with TPM blocks, summing points, multiplying points, branch points, and connections. The graphical representation of DEDS is analogous to that of control system theory. TPM has a matrix representation of its transition and information map. This matrix repersentation simplifies the analysis of the DEDS.

1 INTRODUCTION

Most of systems can be repersented by their states which characterize their behaviors. The states of a system is finite or infinite according to its modeling objectives. In case controlling the motion of the robot system in Fig.1, the velocity and the position can represent the robot's behavior with infinite state values. However, two states, 'WORK' and 'IDLE' can represent the robot's behavior when scheduling the job of the robot. In any case, a system can be represented by its states. Especially, discrete event dynamic systems(DEDS) can be represented by countable number of states. The dynamics of DEDS is the transition between states. These transitions are triggered by some factors in the system or outside the system. The factors have two components, events and informations. Events can enforce the transition of states due to the the informations which are entered into the system. From the physical limitations, the transition of states takes time. Hence, the time when some event is occurred is different from the time when state is changed. However, we are interested in the logical behavior of the system, we assume that the transition time is negligible, namely the transition time is zero as in Fig2. Events and informations are distinguished by their appearances. Events are appeared instaneously, however informations are maintained between events. Hence, for the given DEDS there exist internal events and informations which is in the system, and external ones which is outside the system. It is noticed that one system's internal events and informations are another system's external ones. States, events, and informations are illustrated in the following examples.

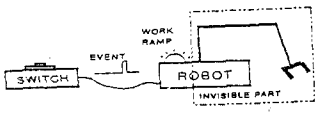


Fig.1

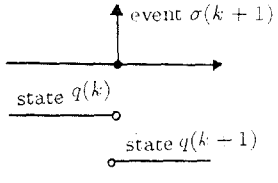


Fig2.

Let's consider a DEDS in Fig1 . The robot starts work when the switch makes a pulse. The operator observes the busy lamp instead of the robot since the robot exists in invisible position. There are two DEDS, the operator-switch system and the robot system. The switch has two states, 'ON' and 'OFF'. The robot has two states, 'IDLE' and 'WORK'. There are four events, 'PUSH BUTTON', 'RELEASE BUTTON', 'STARTS WORK', and 'FINISHES WORK'. The events, 'PUSH BUTTON' and 'RELEASE BUTTON', are internal events which is generated by the human oper-

ator. The event, 'FINISHES WORK', is an internal event which is generated by the robot system. However, 'STARTS WORK' is an external event, since it must be generated by the other systems. There are two informations, 'BUSY LAMP ON' and 'BUSY LAMP OFF'. The state and the events are shown by the state transition diagram in Fig3. The operator pushes the button according to the staus of the busy lamp which provides an information of the robot system. This information is a reflection of the states in the robot system. The switch and the robot are connected by a line which transmit the pulse. This line assigns the event 'PUSH BUTTON' to the event 'STARTS WORK'. These relations are shown by the block diagram in Fig4. From the above analysis of DEDS, we can know the followings: A DEDS is composed of smaller DEDS which have inputs and outputs. Inputs and outputs have two components, events and informations. The inputs and the outputs are connected by some physical media. Hence, we can regard the DEDS as a two inputs and two outputs system whose inputs are external events and informations ,and outputs are internal events and informations as in Fig5. Events can enforces the transition of states due to the status of informations as in the following example.

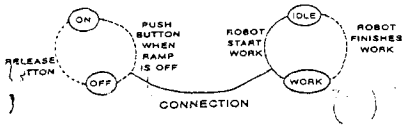


Fig3

Let's consider the FLIP-FLOP system in Fig6 . The state of the FLIP-FLOP is the logical value of the Q-output. The event is the up-edge of the pulse input. The information is the D-input of the FLIP-FLOP. The transition of the state can be occured when the up-edge of the pusle input is appeared. Clearly, the D-input of the FLIP-FLOP controls the transition of the state.

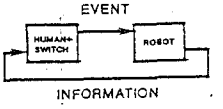


Fig4

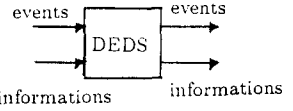


Fig5.

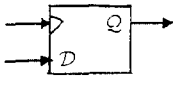


Fig6

In summary, we can treat DEDS as a two ports model(TPM) whose inputs and outputs are composed of events and informations. TPM representation is distinguished from the other finite state machine models since TPM classifies events and informations into external and internal ones. Hence, the basic elements of TPM are states, external events, internal events, external informations, internal informations, a state transition map, and an information map. Using TPM, we can treat DEDS as input/output systems. With the basic assumption that all events cannot occurred simultaneously, we can define the summing point, the multiplying point, and the branch point as block diagram elements. Therefore, we can represent the most DEDS as a block diagram with TPM blocks, summing points, multiplying points, branch points, and connections. This block diagram representation is analogous to the control system theory.

Our model is differs from other models found in the literature [1]-[16]. The usual models for analysing DEDS have states and events but their events are not classified into internal and external ones. The plant and the controller are not described by the same DEDS structure in the usual state machine model[1],[6]-[7],[9]-[12],[14]. For example, in the supervisory control theory[1],[6],[9],[11],[12],[14]. the controlled discrete event process(CDEP) and the supervisor have events and control patterns as inputs and outputs. The events are generated only in the CDEP and the supervisor only accepts the generated events. The control patterns are generated in the supervisor and the CDEP only accepts the control patterns. However, the CDEP and the supervisor are also DEDS. This non-symmetric representation of DEDS limits the connected system representations and the block diagram representations of DEDS. However, TPM has symmetric inputs and outputs. Hence, the connected DEDS block diagrams can be represented by TPM. TPM is a general model in the sense that the DEDS which is modeled by the existing state machine model always be represented by TPM. This is because TPM has the extended elements than the other models for DEDS. Hence, the existing theories for the state machine model can be reformulated for TPM, i.e. the supervisory controller, the output stabilizer [16], and the state observer [10],[15]. can be constructed in the generalized form.

TPM also has matrix representations of its transiton map and information map. This matrix representation is analgous to the time varying discrete time systems. Using the matrix representation of TPM, one can easily analysis the DEDS with the computer. The reachability, the controllability, the stabilizability, and the output stabilizability of TPM are easily checked with simple matrix algebra. The controllers of DEDS are also easily obtained when the DEDS is modeled by TPM with the matrix representation.

Several classes of models have been proposed for describing the behavior of DEDS including, especially Petri nets and state machine(FSM). Automata and formal language models initiated by Ramdge and Wonham [11]-[12], have been successfully used to study the properties of DEDS in a variety of applications. Their formal language model is essentially based on FSM. The Ramadge and Wonham's approach treats the open-loop plant and the controller separately where other approaches require a separate model for each control policy. The Ramadge and Wonham's approach uses different structures for the open-loop plant and the controller unlikely those in control system theory. Due to this unsymmetric representation of plants and controllers, connected DEDS block diagram representations of such model are impossible. And the controller synthesis procedures are complicated since the controllers are represented by a pair of an automaton and a function. However, both plants and controllers are DEDS, hence a unified representation of plants and controllers are expected. One unified representation is our TPM. TPM has symmetric input/output structure, so that plants and controllers can be represented by the same structure and block representation is possible for connected DEDS.

The objective of the Ramadge and Wonham's model is to control DEDS, so that the resulting event sequences lies in a desired set or language of strings of events. In contrast, Özveren and Willsky proposed another objective, controlling the state of DEDS so that it returns regulary to a specified set of states in [15]-[16]. Their objectives can be unified into controlling the DEDS so that the resulting state sequences lies in a desired set of strings of ststes. Hence, controlling the state sequences is a basic control objective of DEDS.

In the next section, we introduce the mathematical model and assumptions for TPM. The definition of summing poins, multiplying points, strings, and finite systems are described. n Section 3, we present some conclusions.

2 PRELIMINARIES

A. *Mathematical Model of Two-Port Machines*

Two-Port Machines(TPM) are DEDS which is characterized by the following seven-tuples:

$$\mathcal{D} = (\mathcal{Q}, \Sigma_{in}, \Sigma_{out}, \Gamma_{in}, \Gamma_{out}, f, g) \tag{1}$$

where $\mathcal{Q}$ is the countable set of **states**, Σ_{in} is the finite set of **input events** generated externally and Σ_{out} is the finite set of **output events** generated internally. Γ_{in} is the set of all subsets of I_{in} i.e. $\Gamma_{in} = 2^{I_{in}}$ where I_{in} is the set of **input informations**. Γ_{out} is the set of all subsets of I_{out} i.e. $\Gamma_{out} = 2^{I_{out}}$ where I_{out} is the set of the **output informations**. The input sets, Σ_{in}, Γ_{in} and output sets, Σ_{out}, Γ_{out} are disjoint each other, i.e.

$$\Sigma_{in} \cap \Sigma_{out} = \emptyset, \quad \Gamma_{in} \cap \Gamma_{out} = \emptyset. \tag{2}$$

The **system events** set Σ_{sys} is defined by

$$\Sigma_{sys} := \Sigma_{in} \cup \Sigma_{out} \ . \tag{3}$$

The elements $\gamma_i \in \Gamma_{in}$, $\gamma_o \in \Gamma_{out}$ is said to be an **input pattern** and an **output pattern**, respectively. $f \subset \Gamma_{in} \times \Sigma_{sys} \times \mathcal{Q} \times \mathcal{Q}$ is a **transition relation**. $g : \Gamma_{in} \times \mathcal{Q} \rightarrow \Gamma_{out}$ is the **information output function**.

With an abuse of notation we identify the relation f with the point-to-set function $f(\gamma_i, \sigma, q) = \{q' \in \mathcal{Q} | (\gamma_i, \sigma, q, q') \in f\}$ where $q, q' \in \mathcal{Q}$, $\sigma \in \Sigma_{sys}$, ,and $\gamma_i \in \Gamma_{in}$. And we say that $f(\gamma_i, \sigma, q)$ is defined if it is nonempty. If $f(\gamma_i, \sigma, q)$ contains at most one state for every $(\gamma_i, \sigma, q) \in \Gamma_{in} \times \Sigma_{sys} \times \mathcal{Q}$, $\mathcal{D}$ is said to be **deterministic** and otherwise **undeterministic**. f can be extended to set-to-set function $\tilde{f} : \Gamma_{in} \times \Sigma_{sys} \times 2^{\mathcal{Q}} \rightarrow 2^{\mathcal{Q}}$ by $\tilde{f}(\gamma_i, \sigma, \tilde{q}) = \bigcup_{q \in \tilde{q}} f(\gamma_i, \sigma, q)$ where $\tilde{q}$ is a subset of $\mathcal{Q}$, i.e. $\tilde{q} \in 2^{\mathcal{Q}}$. The function $\tilde{f}$ preserves the nature of the function f since $\tilde{f}(\gamma_i, \sigma, \{q\}) = f(\gamma_i, \sigma, q)$ where $q \in \mathcal{Q}$. Hence, we will identify the set-to-set function $\tilde{f}$ with the point-to-set function f and the singleton set $\{q\}$ with the state q.

The function g is a point-to-set function considering the output pattern γ_o as a subset of the information set

I_{out}, and the input pattern γ_i as an element of the input information set I_{in}. The function g can be extended to a set-to-set function $\tilde{g} : \Gamma_{in} \times 2^{\mathcal{Q}} \to 2^{I_{out}}$ by $\tilde{g}(\gamma_i, \tilde{q}) := \bigcup_{q \in \tilde{q}} g(\gamma_i, q)$ where $\tilde{q} \in 2^{\mathcal{Q}}$. We will identify $\tilde{g}$ with g since the set-to-set function $\tilde{g}$ preserves the nature of the point-to-set function g.

The system $\mathcal{D}$ is said to be **finite** if the number of states, informations, and events are finite. The system $\mathcal{D}$ is said to be **proper** if $f(\gamma_i, \sigma, \tilde{q}) \neq \emptyset$ for all $\tilde{q} \neq \emptyset, \sigma \in \Sigma_{in}$, and $\gamma_i \in \Gamma_{in}$. The real systems are always proper since, all external events and informations can be applied to the system at any time. The system $\mathcal{D}$ is said to be **strict** if $g(\gamma_1, q) = g(\gamma_2, q)$ for all $\gamma_1, \gamma_2 \in \Gamma_{in}$ and $q \in \mathcal{Q}$. When a system is strict then the output information is the function of state only, i.e. the output function can be represented by $\gamma_o = g(q)$.

A proper system $\mathcal{D}$ id said to be **striclty proper** when the system is also strict. The system $\mathcal{D}$ is said to be **reasonable** if $f(\gamma_1, \sigma, q) \subset f(\gamma_2, \sigma, q)$ for all $\gamma_1 \subset \gamma_2$. The reasonable system has more action for more information. We will consider the strictly proper and reasonable systems.

B. Assumptions For TPM

For TPM work propoerly, we assume the followings:

ASSUMPTION 1 *All events occur instaneously and asynchronously.*

ASSUMPTION 2 *Two events cannot occur simultaneously and the time between consecutive events is not fixed.*

These assumptions are already accepted by many authors [1],[11],[17], for the formal language or automata model. These assumptions eable counting the events and numbering the events sequentially. Our model TPM has information input/output which is similar to the control patterns of the controlled discrete event system in the formal language model.

With the above assumptions, the operation of TPM is described as follows: TPM is a system such that if it is in state $q \in \mathcal{Q}$ and its input pattern is γ_i, then its output will be $g(\gamma_i, q)$, furthermore if an event $\sigma \in \Sigma_{sys}$ is occurred then it will be in one of the state in $f(\gamma_i, \sigma, q)$. Namely, if the k'th state and input pattern are $q(k)$ and $\gamma_i(k)$ then the k'th output pattern is $\gamma_o(k) = g(\gamma_i(k), q(k))$. If the $(k+1)$'th event $\sigma(k+1)$ occurs, then the $(k+1)$'th state is $q(k+1) \in f(\gamma_i(k), \sigma(k+1), q(k))$.

In other DEDS model including the formal language model, only one DEDS and the associated controller is considered. Hence, these models does not explains the connection of DEDS however TPM explains the connection of DEDS as in Section 2. The following is an assumption that is needed to desrcibe such DEDS.

ASSUMPTION 3 *All DEDS generate different events and informations each other.*

This assumption is very natural so that one may overlook this. But this assumption gives a foundation in order to integrate several DEDS as in Section 2.

C. Symbols and Strings of TPM

We introduce symbol sets and string sets with respect to the system $\mathcal{D}$. Λ_{in}, Λ_{out}, and Λ_{sys} are an **input symbol** set, an **output symbol** set, and an **system symbol** set, respectively, defined by

$$\begin{aligned} \Lambda_{in} &:= \Gamma_{in} \times (\Sigma_{in} \cup \{\epsilon\}) \\ \Lambda_{out} &:= \Gamma_{out} \times (\Sigma_{out} \cup \{\epsilon\}) \\ \Lambda_{sys} &:= \Gamma_{in} \times (\Sigma_{sys} \cup \{\epsilon\}) \end{aligned} \tag{4}$$

where ϵ denotes the null event which cannot transit states, i.e. $f(\gamma_i, \epsilon, \tilde{q}) = \tilde{q}$, for all $\gamma_i \in \Gamma_{in}$ and $\tilde{q} \subset \mathcal{Q}$.

Let X be a symbol set, X^* denote the set of strings of the elements of the symbol set X , including the null symbol ν. Hence, we can define the symbol sets, Λ_{in}^*, Λ_{out}^*, and Λ_{sys}^* . Each string set is a semigroup with the identity ν under concatenation, i.e. each string set is a monoid.

We can extend the set-to-set transition function f so that it maps $\Lambda_{sys}^* \times 2^{\mathcal{Q}}$ into $2^{\mathcal{Q}}$, by the repeated application of the equalities:

$$\begin{aligned} f(\nu, \tilde{q}) &:= \tilde{q} \\ f(\lambda, \tilde{q}) &:= f(\gamma, \sigma, \tilde{q}) \\ f(s\lambda, \tilde{q}) &:= f(\lambda, f(s, \tilde{q})) \end{aligned} \tag{5}$$

where $\tilde{q} \in 2^{\mathcal{Q}}$, $s \in \Lambda_{sys}^*$, and $\lambda = (\gamma, \sigma) \in \Lambda_{sys}$. Using these sysmbols, the system $\mathcal{D}$ can be represented by the block symbol as in figure 2.2.

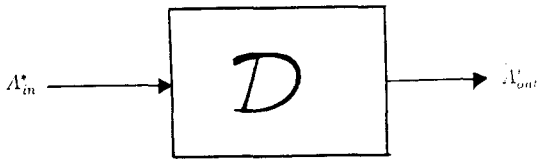


fig 2.2

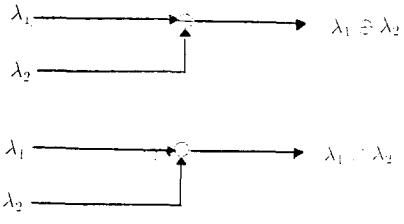
We will define two partial operations, summing and multiplying for a symbol set $\Lambda := \Gamma \times \Sigma$. The set Σ may include the null event ϵ. The summing operator $\oplus : \Lambda \times \Lambda \to \Lambda$ is a partial operator such that

$$\lambda_1 \oplus \lambda_2 = \begin{cases} (\gamma_1 \cup \gamma_2, \sigma_1), & \sigma_2 = \epsilon \\ (\gamma_1 \cup \gamma_2, \sigma_2), & \sigma_1 = \epsilon \\ (\gamma_1 \cup \gamma_2, \sigma_1), & \sigma_1 = \sigma_2 \\ undefined, & otherwise \end{cases} \tag{6}$$

where $\lambda_1, \lambda_2 \in \Lambda$, i.e. $\lambda_1 = (\gamma_1, \sigma_1), \lambda_2 = (\gamma_2, \sigma_2)$ for some $\gamma_1, \gamma_2 \in \Gamma$ and $\sigma_1, \sigma_2 \in \Sigma$. The multiplying operator $\otimes : \Lambda \times \Lambda \to \Lambda$ is a partial operator such that

$$\lambda_1 \otimes \lambda_2 = \begin{cases} (\gamma_1 \cap \gamma_2, \sigma_1), & \sigma_2 = \epsilon \\ (\gamma_1 \cap \gamma_2, \sigma_2), & \sigma_1 = \epsilon \\ (\gamma_1 \cap \gamma_2, \sigma_1), & \sigma_1 = \sigma_2 \\ undefined, & otherwise. \end{cases} \tag{7}$$

These partial operators are represented by graphic symbols as follows:



Using these graphic symbols we can represent the symbol flow of the system. By the assumption 2 in Section 2, the event in the symbol λ_1 cannot be the same event in the symbol λ_2 at any instance. Hence the partial operators, $\oplus$ and $\otimes$, are consistent with the graphical representations.

The string set $\vec{\Lambda}^*_{sys}$, **acts** on the system $\mathcal{D}$, so that $2^{\mathcal{Q}}$ is a **right** $\vec{\Lambda}^*_{sys}$**-set** which means that, we can define $\vec{\Lambda}^*_{sys}$**-action** on $2^{\mathcal{Q}}$ by using the set-to-set function f as

$$2^{\mathcal{Q}} \times \vec{\Lambda}^*_{sys} \to 2^{\mathcal{Q}}, \quad (\tilde{q}, s) \longmapsto \tilde{q}s \tag{8}$$

where $\tilde{q}s = f(s, \tilde{q})$. Clearly, if $s_1, s_2 \in \vec{\Lambda}^*_{sys}$ then $\tilde{q}(s_1 s_2) = (\tilde{q}s_1)s_2$ and $\tilde{q}\nu = \tilde{q}$.

Since $f(\lambda, \{q_1\} \cup \{q_2\}) = f(\lambda, \{q_1\}) \cup f(\lambda, \{q_2\})$ for all $q_1, q_2 \in \mathcal{Q}$, the action can be characterized by the map

$$\mathcal{Q} \times \vec{\Lambda}^*_{sys} \to 2^{\mathcal{Q}}, \quad (q, s) \longmapsto qs \quad . \tag{9}$$

where $qs = f(s, \{q\})$.

Let $T_{\mathcal{Q}}$ be a set of all transition map of $\mathcal{Q}$, i.e. $T_{\mathcal{Q}} = \{t \,|\, t : \mathcal{Q} \to 2^{\mathcal{Q}}\}$, then there is a homomorphism

$$\psi_{sys} : \vec{\Lambda}^*_{sys} \to T_{\mathcal{Q}} \quad s \longmapsto t \tag{10}$$

such that $t(q) = qs$ for all $q \in \mathcal{Q}$. Using this homomorphism ψ_{sys}, we will define the **unacceptable string set**

$$\vec{\Lambda}^*_{uacc} = \psi^{-1}_{sys}(0) \tag{11}$$

where 0 is the zero transition map which maps all state to the empty set $\emptyset$. The **acceptable string set** is defined by

$$\vec{\Lambda}^*_{acc} = \vec{\Lambda}^*_{sys} - \vec{\Lambda}^*_{uacc} \tag{12}$$

The string set $\vec{\Lambda}^*_{acc}$ represents all possible strings which can occur in the system $\mathcal{D}$. $Im\psi_{sys}$ represents the possible state transition map due to strings. $Ker\psi_{sys}$ represents the set of strings corresponding to the identity transition. Each $t \in T_{\mathcal{Q}}$ can be extended to set-to-set function $\tilde{t} : 2^{\mathcal{Q}} \to 2^{\mathcal{Q}}$ by

$$\tilde{t}(\tilde{q}) := \bigcup_{q \in \tilde{q}} t(q) \quad . \tag{13}$$

We will identify the point-to-set function t with the set-to-set function $\tilde{t}$.

D. Finite Systems

When the system $\mathcal{D}$ is finite, by numbering the state, we can represent the subset $\tilde{q}$ of $\mathcal{Q}$ as a **Boolean vector** $\underline{q}$. The transitions can be represented by Boolean matrices M whose elements are 0 or 1, as in Figure 2.3. Since we identify state $q_i \in \mathcal{Q}$ with the singleton set $\{q_i\}$, each state q_i can also be identified with the Boolean vector $\underline{q}_i$ whose elements are zero except i'th element.

$$\mathrm{M} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Fig 2.3.

The empty state set is represented by the Boolean vector $\underline{0}_{n\times 1}$ where n is the number of states in $\mathcal{Q}$. The set $\mathcal{Q}$ is represented by the Boolean vector $\underline{1}_{n\times 1}$. In the Boolean matrix M, each column represents the starting state and each row represents the endding state. If the element m_{ij} of the transition matrix M is 1 then the transition from q_j to q_i is possible, otherwise impossible. Let $M_{\mathcal{Q}}$ be the set of Boolean transition matrices of $\mathcal{Q}$. The operator $\vee, \wedge$, **and** $\odot$ is defined for the matrices. The set $M_{\mathcal{Q}}$ is a monoid under the operator $\odot$. If the current state is in the set $\underline{q}$, then the next possible states set can be represented by $M \odot \underline{q}$.

When the number of output information set I_{out} is finite, by numbering the output information, the output pattern γ_o can be represented by a Boolean vector. We will identify the output pattern with the Boolean vector when the number of information is finite. In this case, the output information function g can be represented by a Boolean matrix G so that $\gamma_o = G \odot \underline{q}$.

Clearly there exists an isomorphism between the set of transition map, $T_{\mathcal{Q}}$ and the set of Boolean matrices $M_{\mathcal{Q}}$. The composition of the isomorphism and the map ψ_{sys} maps Λ_{sys}^* into $M_{\mathcal{Q}}$, i.e. there exist a homomorphism

$$F : \Lambda_{sys}^* \to M_{\mathcal{Q}}, \quad s \longmapsto F(s) \tag{14}$$

where $F(s_1 s_2) = F(s_2) \odot F(s_1)$. This homomorphism also provides a map

$$\Lambda_{sys} \to M_{\mathcal{Q}}, \quad \lambda \longmapsto F(\lambda) \tag{15}$$

Similary, for each $\gamma_i \in \Gamma_{in}$, we can assign a Boolean matrix such that $\gamma_o = g(\gamma_i, q) = G(\gamma_i) \odot q$.

Hence, the dynamics of a finite system can be represented by the following equations:

$$\begin{aligned} q(k+1) &\in F(\lambda(k+1)) \odot q(k) \\ \gamma_o(k) &= G(\gamma_i(k)) \odot q(k) \end{aligned} \tag{16}$$

where $\lambda(k) = (\gamma_i(k), \sigma(k+1)) \in \Lambda_{sys}$. When the finite system is deterministic, the dynamics can be represented by the following equations:

$$\begin{aligned} q(k+1) &= F(\lambda(k+1)) \odot q(k) \\ \gamma_o(k) &= G(\gamma_i(k)) \odot q(k) \quad . \end{aligned} \tag{17}$$

When the system is finite the reachability is easily checked by the Boolean matrix operation as follows.

Theorem 2.1: Let $F_{sys} := F(\Lambda_{sys})$. The state q_i is reachable from the state q_j if and only if there exist a sequence of matrices $F_k, F_{k-1}, ..., F_1 \in F_{sys}$ such that

$$q_i^T \odot (F_k \odot F_{k-1} \odot ... \odot F_1) \odot q_j \neq 0 \tag{18}$$

3 CONCLUSION

In this paper, TPM model for DEDS is proposed. The proposed model has symmetric input and output, hence the interconnection between the DEDS block is possible. This symmetry of TPM is similar to the C/E system in [17]. The C/E system is general than our TPM model. Because of this generality , the analysis of the C/E system is somewhat complicated. TPM is a finite state machine which has the symmetric input and output. The analysis of TPM is easy due to the specified structure. TPM is a logical model where C/E system is timed model.

TPM is an extended model of the system used in the supervisory control theory and stabilizability theory [1],[2],[15],[16]. Hence, it is expected that the supervisory control theory and the stabilizability theory are extended for TPM. The synthesis of TPM controller is future researches.

References

[1] R. Cieslak. C. Desclaux. A. Fawaz, and P. Varaiya, "Supervisory control of discrete event processes with partial observations," *IEEE Trans. Automat. Contr.*, vol. AC-33, no. 3, pp. 249-260, 1988.

[2] K. Inan and P. Varaiya ," Finitely recursive process models for discrete event systems ," *IEEE Trans. Automat. Contr.*, vol. AC-33, no. 7, pp. 626-639, 1988.

[3] G.Cohen, D.Dubois, J. P. Quadrat, and M. Voit, "A linear system-theoretic view of discrete event processes and its use for the performance evaluation in manufacturing," *IEEE Trans. Automat. Contr.*, vol. AC-30, no. 3, pp. 210-220, 1985.

[4] C. H. Golaszewski and P. J. Ramadge. " Mutual exclusion problems for discrete event systems with shared events ," in *Proc. 27th IEEE Conf. Decision and Control(Austin, TX)*, pp. 234-239, Dec. 1988.

[5] S. Lafortune and E. Wong ," A state model for the concurrency control problem in data base management systems ," *IEEE Trans. Automat. Contr.*, vol. AC-33, no. 5, pp. 439-447, 1988.

[6] F Lin and W. M. Wonham ," Decentralized control and coordination of discrete-event systems ," in *Proc. 27th IEEE Conf. Decision and Control(Austin, TX)*, pp. 1125-1130, Dec. 1988.

[7] F Lin and W. M. Wonham ," On obvervability of discrete-event systems ," in *Inform. Sci.*, vol. 44, pp. 173-198, 1988.

[8] J. L. Peterson , *Petri Net Theory and the Modelling of Systems.* , Englewood Cliffs, NJ: Prentice-Hall, 1981

[9] P. J. Ramadge. " Some tractable supervisory control problems for discrete event systems described by Büchi automata ," in *IEEE Trans. Automat. Contr.*, vol. AC-34, no. 1, pp. 10-19, 1989.

[10] P. J. Ramadge ," Observability of discrete event systems ," in *Proc. 25th IEEE Conf. Decision and Control(Athens, Greece)*, pp. 1108-1112, Dec. 1986.

[11] P. J. Ramadge and W.M. Wonham ," Supervisory control of a class of discrete-event processes ," in *SIAM J. Contr. Optimization*, vol. 25, no. 1, pp. 206-230, 1987.

[12] P. J. Ramadge and W.M. Wonham ," Modular feedback logic for discrete event systems ," in *SIAM J. Contr. Optimization*, vol. 25, no. 5, pp. 1202-1218, 1987