# A Reachable Graph of Finite and Deterministic DEVS Networks

Moon Ho Hwang and Bernard P. Zeigler

Arizona Center for Integrative Modeling and Simulation,
Electrical and Computer Engineering Department,
The University of Arizona, Tucson, AZ 85721, USA
{mhhwang, zeigler}@ece.arizona.edu

**Keywords:** Discrete Event System Specification(DEVS), Finite Reachable Graph, Time Abstraction, Difference Bound Matrix, Verification

**Abstract**

To obtain the finite reachable graph of a Discrete Event System Specification (DEVS) network, this paper uses a subclass of DEVS, called finite and deterministic DEVS. This subclass has been restricted to have (1) finite sets of both events and states, (2) the rational-number time advance function, (3) the time independent external transition, and (4) the selective reschedule functionality.

For abstracting the infinite-state behavior to a finite-state reachable graph, we use the clock zone that is a conjunction of inequalities of clocks. A clock zone-based generating algorithm of the reachable graph of the coupled DEVS is proposed and its completeness and complexity are addressed.

## I. Introduction

Verification of discrete event systems has been researched based on the assumption of the finite state space of the target system [11]. However, when handling a dense-time system in which a state transition is able to occur at any real-valued time, we can encounter the infinite state problem of the system behavior. Since DEVS [15] [16] has its the time advance function as a map from a state to a real number as well as the elapsed time is reset whenever any external input occurs, these features cause the infinite state problem, especially when building a network of DEVS models.

To obtain the finite reachable graph of a DEVS network and to analyze its global timed behavior, several papers have been published. [14] showed a symbolic representation of the time advance mechanism of atomic DEVS and proposed the reachable tree of symbolic DEVS networks. [5] showed how to get a timed state reachability graph from the ordinary coupled DEVS. [13] used a non-deterministic timed DEVS, called real-time DEVS (RT-DEVS) [4] whose time advance is mapping from a state to a real-valued interval rather than a single real number in order to verify a time-critical control problem. All of the above, however, seem to assume that the target system is a closed system that is not interactive with external influences. Therefore, achieving finiteness with the real-valued time looks still to be an open problem in DEVS.

Dropping the input free assumption, schedule-preserved DEVS (SP-DEVS) [9] allows an open system whose external input transitions are allowed at any time. One advantage of SP-DEVS is the finiteness of the reachable graph of coupled SP-DEVS which can be obtained by using a time-abstraction method, called the relative-schedule abstraction [6]. Thus, we can say that SP-DEVS is closed under coupling. In spite of decidability of qualitative analysis (such as deadlock, livelock, and fairness) as well as quantitative analysis (such as Min/Max processing time) [8], there is also a down side. The state space can explode because of self-loop states that are usually used to avoid the phenomenon called “once it becomes passive, it never returns active (OPNA)” [7].

In this paper, we use a class of DEVS, called finite and deterministic DEVS (FD-DEVS) whose (1) sets of events and states are finite, (2) the time advance is a map from a state to a non-negative rational number, (3) the external transition is time-independent, and (4) the reschedule of an external event can be selective. This class of DEVS [1]has no OPNA problem but the relative-schedule abstraction could not get the finite reachable graph when the DEVS network under considering is *partially rescheduled*[7].

Instead of using the relative schedule abstraction, this paper uses the difference bound matrix [3] that is an efficient data structure representing a conjunction of elapsed times of associated atomic models. Based on

[1]In [7], we called it schedule-controllable DEVS but here we newly name it finite and deterministic DEVS.

the difference bound matrix, we propose an algorithm generating the finite reachable graph of the coupled FD-DEVS.

This paper is organized as follows. Section II introduces the definition and the behavior of atomic FD-DEVS as well as coupled FD-DEVS. Review of the difference bound matrix and an algorithm generating reachable graph of coupled FD-DEVS, and its completeness and complexity are addressed in Section III. Section IV gives remarks on the factors increasing the number of vertices in the reachable graph. Finally, conclusions and further research directions are given in Section V.

## II. Finite-State and Deterministic DEVS (FD-DEVS)

In FD-DEVS, the modifier "finite" means that the sets of events and states are finite while "deterministic" indicates that all characteristic functions associated are deterministic. The formal definition and the behavior of FD-DEVS is defined in this section.

### A. Atomic FD-DEVS

*1) Definition of Atomic FD-DEVS:* An *atomic FD-DEVS* specifies the dynamic behavior. An *atomic FD-DEVS* is a 9-tuple,

$$M =< X, Y, S, s_0, \tau, \delta_x, \rho, \delta_\tau, \lambda >$$

where,

- $X$ and $Y$ are *finite sets of input and output events*, respectively such that $X \cap Y = \emptyset$.
- $S$ is a *non-empty and finite state set*.
- $s_0 \in S$ is the *initial state*.
- $\tau : S \rightarrow \mathbb{Q}_{[0,\infty]}$ is the *time advance function* where $\mathbb{Q}_{[0,\infty]}$ denotes a set of non-negative rational numbers with infinity.
- $\delta_x : S \times X \rightarrow S$ is the *external transition function*.
- $\rho : S \times X \rightarrow \mathbb{B}$ is the *reschedule-indicating function* that returns 1 when a reschedule is needed; otherwise, returns 0.
- $\delta_\tau : S \rightarrow S$ is the *internal transition function*.
- $\lambda : S \rightarrow Y \cup \{\epsilon\}$ is the *internal output function* where $\epsilon$ is the *non-event* such that $\epsilon \notin X \cup Y$. □

*2) State Transition of Atomic FD-DEVS:* Given an atomic FD-DEVS $M$, in addition to state $s \in S$, the *total states set* considers the schedule $t_s$ that is the *life time of state $s$* as well as the elapsed time $e$ *since the last time updating $t_s$* such that

$$Q = \{(s, t_s, e) | s \in S, t_s \in \mathbb{Q}_{[0,\infty]}, 0 \leq e \leq t_s\} \quad (1)$$

From the total state set $Q$ and the *total event set* $Z = X \cup Y \cup \{\epsilon\}$, the *total state transition function* $\delta : Q \times Z \rightarrow Q$ maps from one total state to the next. For $(s, t_s, e) \in Q$ where $t_s \in \mathbb{Q}_{[0,\infty]}$ and $z \in Z$,

$$\delta((s, t_s, e), z) = (s', t'_s, e') \quad (2)$$

where
[*External Transition*] for $z \in X$, $(s', t'_s, e') =$

$$\begin{cases} (\delta_x(s,x), \tau(\delta_x(s,x)), 0) & \text{for } \delta_x(s,z) \vdash, \rho(s,z) = 1 \\ (\delta_x(s,x), t_s, e) & \text{for } \delta_x(s,z) \vdash, \rho(s,z) = 0 \\ (s, t_s, e) & \text{otherwise} \end{cases} \quad (2a)$$

[*Internal Transition*] for $z \in Y \cup \{\epsilon\}$, $(s', t'_s, e') =$

$$\begin{cases} (\delta_\tau(s), \tau(\delta_\tau(s)), 0) & \text{for } z = \lambda(s), e = t_s \\ \text{undefined} & \text{otherwise} \end{cases} \quad (2b)$$

We can see that every external event $x \in X$ can occur at any state $s$. Yet because $\delta_x(s, x)$ and $\rho_x(s, x)$ are partial functions, if either $\delta_x(s, x)$ or $\rho_x(s, x)$ is not defined on $s$ and $x$, we assume that there is no change in the state. If they are defined, depending on the value of $\rho(s, x)$, the next state and its schedule change.

Unlike to the external input transition, the internal transition from $s$ can be possible only when the elapsed time $e$ reaches the schedule $t_s$, at this time, the associated event is the output of $s$, i.e., $\lambda(s)$. Otherwise, the internal transition is undefined, that means impossible. [2]

*Example 1:* Let's consider a toaster as shown in Figure 1(a). Using atomic FD-DEVS, we can model it as shown in Figure 1(b), that is $X = \{\texttt{push}\}; Y = \{\texttt{pop}\}; S = \{\texttt{E}, \texttt{T}\}; \tau(\texttt{E}) = \infty, \tau(\texttt{T}) = 20; \delta_x(\texttt{E}, \texttt{push}) = \texttt{T}, \delta_x(\texttt{T}, \texttt{push}) = \texttt{T}; \rho(\texttt{E}, \texttt{push}) = 1, \rho(\texttt{T}, \texttt{push}) = 0; \delta_\tau(\texttt{T}) = \texttt{E}; \lambda(\texttt{T}) = \texttt{pop}; s_0 = \texttt{E};$.

The time event $(t, z)$ implies that at time $t$, an event $z$ occurs, so $\omega = (10, \texttt{push})(22, \texttt{push})(30, \texttt{pop})(45, \texttt{push})(58, \texttt{push})(65, \texttt{pop})$ is a sequence of time events. The top and the bottom of Figure 1(c) illustrate a sequence of timed events and its corresponding state trajectory, respectively.

Notice that the external input `push` can be ignored at toasting state `T` (self-looped without any schedule change) and the output, `pop`, is impossible at empty state `E` (so there is no transition arc from `E` with `pop`).

□

### B. Coupled FD-DEVS

*1) Definition of coupled FD-DEVS:* The coupled FD-DEVS defines the structure of a network whose subcomponents are FD-DEVS. A *coupled FD-DEVS* is a 6-tuple,

$$N =< X, Y, D, C_{xx}, C_{yx}, C_{yy} >$$

[2]The elapsed time $e$ is passed continuously. For details of the state trajectory of FD-DEVS associated with a sequence of timed events, the reader can refer to [10].

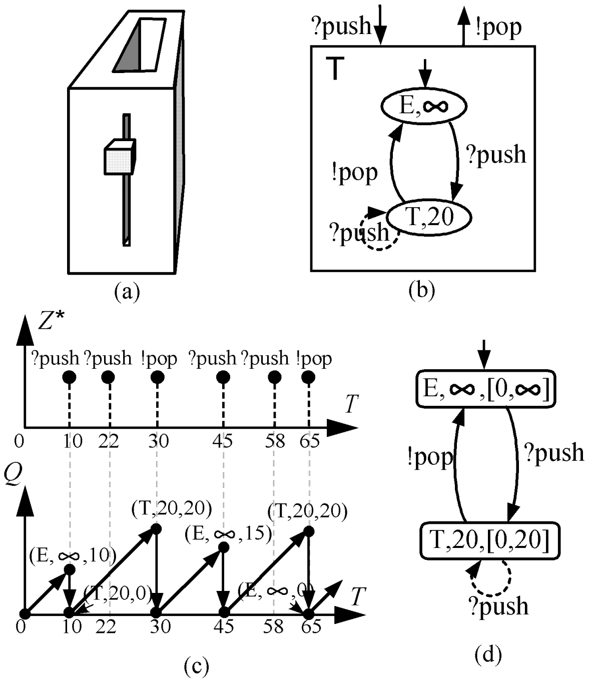

Fig. 1. FD-DEVS Model of Toaster

where

- $X$ and $Y$ are *finite sets of input and output events*, respectively such that $X \cap Y = \emptyset$.
- $D = \{M_i\}$ is the finite set of sub-component FD-DEVSs that are atomic FD-DEVSs. [3]
- $C_{xx} \subseteq X \times \bigcup_{M_i \in D} X_i$ is the *external input coupling relation*.
- $C_{yx} \subseteq \bigcup_{M_i \in D} Y_i \times \bigcup_{M_i \in D} X_i$ is the *internal coupling relation*.
- $C_{yy} = \bigcup_{M_i \in D} Y_i \rightarrow Y \cup \{\epsilon\}$ is the *external output coupling function*. ■

*2) State Transition of Coupled FD-DEVS:* Given $N =< X, Y, D, C_{xx}, C_{yx}, C_{yy} >$, we define $N$'s total state as the combination of sub-components' total states such that

$$Q = \{(\ldots, (s_i, t_{si}, e_i), \ldots) | (s_i, t_{si}, e_i) \in Q_i, M_i \in D\}$$

And we consider its state can change with a triggering event $z \in Z = X \bigcup_{M_i \in D} Y_i \cup \{\epsilon\}$. Thus the *state transition function* $\delta : Q \times Z \rightarrow Q$

$$\delta((\ldots, (s_i, t_{si}, e_i), \ldots), z) = (\ldots, (s'_i, t'_{si}, e'_i), \ldots) \quad (3)$$

can be categorized into two transitions according to the triggering events: [*External Transition Triggering*] For $z \in X$,

$$(s'_i, t'_{si}, e'_i) = \begin{cases} \delta_i((s_i, t_{si}, e_i), x_i) & \text{for } (z, x_i) \in C_{xx} \\ (s_i, t_{si}, e_i) & \text{otherwise} \end{cases} \quad (3a)$$

[*Internal Transition Triggering*] For $z \in \bigcup_{M_i \in D} Y_i \cup \{\epsilon\}$ and $\lambda_{i^*}(s_{i^*}) = z$,

$$(s'_i, t'_{si}, e'_i) = \begin{cases} \delta_i((s_i, t_{si}, t_{si}), z) & \text{for } M_i = M_{i^*} \\ \delta_i((s_i, t_{si}, e_i), x_i) & \text{for } (z, x_i) \in C_{yx} \\ (s_i, t_{si}, e_i) & \text{otherwise} \end{cases} \quad (3b)$$

## III. Finite Reachable Graph of FD-DEVS

If we consider an atomic model $M =< X, Y, S, s_0, \tau, \delta_x, \rho, \delta_\tau, \lambda >$, it is easy to construct its reachable state graph. Let $s$ be a state and $t_s$ be a life time of $s$ since the last schedule. As we mentioned in the previous section, the total state considers additionally the elapsed time $e$ as well as $s$ and $t_s$. Even though instance values of $e$ can be uncountably many because $e \in T$, we can abstract the group of $q = (s, t_s, e)$ where $0 \le e \le t_s$ in to a *zone* that is defined as $z = (s, t_s, 0 \le e \le t_s)$. For example, for the single slot toaster introduced in Example 1, we can draw the reachable state graph as shown in Figure 1(d) whose nodes are zones.

However, if we consider the reachable state graph of a FD-DEVS network, it is not trivial because of the complexity of combination of sub-components' elapsed times. We would abstract uncountable combinations of sub-components' elapsed times into a *clock zone* that can be represented as the inequalities of clocks' boundaries as well as of differences between clocks. This clock zone can be effectively represented with a special data structure, called *difference bound matrix* (DBM) that was originally proposed by Dill [3] that has been used to abstract the behavior of timed automata [2] and [1]. Thus this section starts with a review of DBM first. [4]

### A. Review: Difference Bound Matrix

One strategy to abstract infinite evaluations of clock combinations is to construct convex unions of clock regions. A *clock zone* is a conjunction of inequalities that compare either a clock value or the difference between two clock values to a rational number. [5] For two elapsed time clocks $e_i$ and $e_j$, we allow inequalities of the following types: $e_i \prec d$ (upper bound of $e_i$), $d \prec e_i$

[3]This restriction of only atomic FD-DEVS for sub-components is for the simple explanation. For analysis of hierarchical FD-DEVS networks, we first flatten them, then apply this explanation.

[4]We omit the finite reachable graph of an atomic FD-DEVS because one atomic model can be seen as a coupled FD-DEVS model whose $D$ has the atomic FD-DEVS and $|D| = 1$.

[5]Originally, Dill used the set of integers for clock constraints [3] but here we use the set of rational number for simplification.

(lower bound of $e_i$), $e_i - e_j \prec d$ (upper bound of $e_i - e_j$) and $d \prec e_i - e_j$ (lower bound of $e_i - e_j$) where $\prec$ is $<$ or $\leq$. If there are $n$ clocks, then a clock zone is a convex set in the $n$-dimensional Euclidean space.

Clock zones can be efficiently represented using matrices. Suppose there are $n$ clocks, $e_1, \ldots, e_n$. Then a clock zone is represented by a $(n+1)\times(n+1)$ matrix $\mathcal{D}$. Each entry $\mathcal{D}[i,j]$ has the form $(d_{i,j}, \prec_{i,j})$ and represents the inequality $e_i - e_j \prec_{i,j} d_{i,j}$ where $\prec_{i,j}$ is either $<$ or $\leq$ and $d_{i,j} \in \mathbb{Q} \cup \{\infty, -\infty\}$.

By introducing a special clock $e_0$ that is always 0, for all $i = 1$ to $n$, $\mathcal{D}[i,0]$ and $\mathcal{D}[0,i]$ have special meanings. The first column entry of $i$, $\mathcal{D}[i,0] = (d_{i,0}, \prec)$ shows the upper bound of $e_i$ i.e., $e_i \prec d_{i,0}$, while the first row entry of $i$, $\mathcal{D}[0,i] = (d_{0,i}, \prec)$ means that we have the constraint $0 - e_i \prec d_{0,i}$ or $-d_{0,i} \prec e_i$ that is the lower bound of $e_i$.

Similarly, the upper bound of difference between two clocks $e_i - e_j \prec d$ is $\mathcal{D}[i,j] = (d, \prec)$, while the lower bound of difference between two clocks $-d \prec e_i - e_j$ is $\mathcal{D}[j,i] = (d, \prec)$. And for all $i = 0$ to $n$, $\mathcal{D}[i,i] = (0, \leq)$ if $\mathcal{D}$ is valid. [6]. For example, the two following matrices $\mathcal{D}_a$ and $\mathcal{D}_b$ represent clock zones (a) $0 \leq e_1 \leq 2 \wedge 0 < e_2 \leq 1 \wedge 0 \leq e_1 - e_2 \leq 3$ and (b) $0 \leq e_1 \leq 2 \wedge 0 < e_2 \leq 1 \wedge 0 < e_1 - e_2 \leq 2$, respectively.

| $\mathcal{D}_a$ | | | | $\mathcal{D}_b$ | | | |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | | 0 | 1 | 2 |
| 0 | (0, ≤) | (0, <) | (0, ≤) | 0 | (0, ≤) | (0, <) | (0, ≤) |
| 1 | (2, ≤) | (0, ≤) | (3, ≤) | 1 | (2, ≤) | (0, ≤) | (2, ≤) |
| 2 | (1, ≤) | (0, ≤) | (0, ≤) | 2 | (1, ≤) | (0, <) | (0, ≤) |

*1) Primitive Operations:* Suppose that $(d_1, \prec_1)$ and $(d_2, \prec_2)$ are elements of DBMs. Then $(d_1, \prec_1) = (d_2, \prec_2)$ if $d_1 = d_2$ and $\prec_1 = \prec_2$. And $(d_1, \prec_1) < (d_2, \prec_2)$

1) if $d_1 < d_2$ or
2) if $d_1 = d_2 \wedge (\prec_1 = < \wedge \prec_1 = \leq)$

The addition operation is defined as

$$(d_1, \prec_1) + (d_2, \prec_2) = (d_1 + d_2, \prec)$$

where

$$\prec = \begin{cases} \leq & \text{for } \prec_1 = \leq, \prec_2 = \leq \\ < & \text{otherwise} \end{cases}$$

*2) Equivalence and Inclusion:* Given two DMBs $\mathcal{D}_1$ and $\mathcal{D}_2$, $\mathcal{D}_1 = \mathcal{D}_2$ if $\mathcal{D}_1[i,j] = \mathcal{D}_2[i,j]$ for all $i$ and $j$. And $\mathcal{D}_1 \subseteq \mathcal{D}_2$ if $\mathcal{D}_1[i,j] \leq \mathcal{D}_2[i,j]$ for all $i$ and $j$.

*3) Tightening:* The representation of a clock zone under consideration is not unique because some inequalities might be loose. For example, $e_1 - e_2 \leq 3$ of Figure 2(a) is a loose constraint and it becomes tightened as $e_1 - e_2 \leq 2$ of Figure 2(b) but the movement of inequalities preserves the clock region.

[6]If $\mathcal{D}$ is invalid or empty, there exists $i$ such that $\mathcal{D}[i,i] = (d, \prec)$ and $d < 0$.

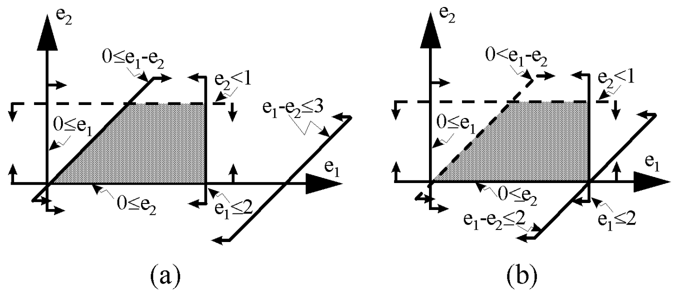

Fig. 2. Canonical DBM (a) a loose DBM (b) a canonical DBM

Dill proposed the shortest method to perform the tightening operations such that

$$\text{if } \mathcal{D}[i,j] + \mathcal{D}[j,k] < \mathcal{D}[i,k] \text{ then } \mathcal{D}[i,k] = \mathcal{D}[i,j] + \mathcal{D}[j,k]$$

The $\mathsf{Tightening}(\mathcal{D})$ operation applies tightening to $\mathcal{D}$ until there is no loose inequality using all-pair shortest path method [12]. The tightening operation produces a *canonical* representation of the clock zone under consideration.

*4) Intersection:* Intersection of two clock zones $\mathcal{D}_1$ and $\mathcal{D}_2$ defines the clock zone that is contained in both $\mathcal{D}_1$ and $\mathcal{D}_2$. This operations is defined as $\mathcal{D} := \mathcal{D}_1 \cap \mathcal{D}_2$ such that

$$\mathcal{D}[i,j] = \begin{cases} \mathcal{D}_1[i,j] & \text{for } \mathcal{D}_1[i,j] \leq \mathcal{D}_2[i,j] \\ \mathcal{D}_2[i,j] & \text{otherwise} \end{cases}$$

*5) Resetting:* The resetting operation resets some clocks in a set $R$ to zero such that $\mathcal{D}' := \mathsf{Resetting}(\mathcal{D}, R)$

$$\mathcal{D}[i,j]' = \begin{cases} (0, \leq) & \text{for } e_i, e_j \in R \\ \mathcal{D}[0,j] & \text{for } e_i \in R, e_j \notin R \\ \mathcal{D}[i,0] & \text{for } e_i \notin R, e_j \in R \\ \mathcal{D}[i,j] & \text{otherwise} \end{cases}$$

*6) Sliding:* The sliding operation moves all upper bounds of each clock, but preserves the rest of the elements. The sliding operation $\mathcal{D}' := \mathsf{Sliding}(\mathcal{D})$ is defined as

$$\mathcal{D}[i,j]' = \begin{cases} (\infty, \leq) & \text{for } i \neq 0, j = 0 \\ \mathcal{D}[i,j] & \text{otherwise} \end{cases}$$

*Remark:* The resulting DBM of all intersection, resetting, and sliding operations cannot be a canonical form, in order to get the canonical DBM, we need to apply the tightening operation after these operations.

## B. DBM Operations for FD-DEVS Behavior

Based on the clock zone, we define a zone as a vector of state and schedule pairs $(\ldots, (s_i, t_{si}), \ldots)$ paired with a clock zone $\mathcal{D}$ i.e., a zone $v = ((\ldots, (s_i, t_{si}), \ldots), \mathcal{D})$.

The clock zone can be interpreted as the *elapsed time zone* in the FD-DEVS application. That is, each clock $e_i$ in the clock zone represents the elapsed time $e_i$ of an atomic FD-DEVS $M_i$.

*1) Widest Clock Zone:* Sometimes we need the *widest* clock zone $\mathcal{D}$ containing all possible clocks in schedule vectors from a given state-schedule vector $(\ldots,(s_i,t_{si}),\ldots)$. For this purpose, we define $\mathsf{MakeWidestClockZone}((\ldots,(s_i,t_{si}),\ldots))$ that returns the widest canonical DMB that can be obtained by $\mathsf{Tightening}(\mathcal{D})$ where $\mathcal{D}$ is as follows

$$\mathcal{D}[i,j] = \begin{cases} (0,\leq) & \text{for } i=j \text{ or } (i=0, j\neq 0) \\ (t_{si},\leq) & \text{for } i\neq 0, j=0 \\ (t_{si},\leq) & \text{for } i<j \\ (t_{sj},\leq) & \text{otherwise} \end{cases}$$

For example $\mathsf{MakeWidestClockZone}(((s_1,2),(s_2,1)))$ returns a canonical $\mathcal{D}$ representing clock zone $0 \leq e_1 \leq 2 \wedge 0 \leq e_2 \leq 1 \wedge -1 \leq e_1 - e_2 \leq 2$.

*2) Time Advancing by Schedule:* Given a zone $v = ((\ldots,(s_i,t_{si}),\ldots),\mathcal{D})$, the clock zone when all clocks in $\mathcal{D}$ elapse until $e_i$ reaches $t_{si}$ is achieved by Algorithm 1, that is (1) setting the lower bound of $e_i$ to its schedule time $t_{si}$ and then (2) tightening all other boundaries.

**Algorithm 1** $\mathsf{MakeClockZoneAt}(t_{si},\uparrow \mathcal{D})$
```
1: $\mathcal{D}[0,i] = (-t_{si},\leq)$;
2: $\mathsf{Tightening}(\mathcal{D})$;
```

For instance, let's consider a state and schedule vector $((s_1,2),(s_2,1))$ and, initially, there is no difference between associated two clocks, i.e., the clock zone $\mathcal{D}$ is $0 \leq e_1 \leq 1 \wedge 0 \leq e_2 \leq 2 \wedge e_1 - e_2 = 0$ (that is before tightening). The resulting matrix $\mathcal{D}' = \mathsf{MakeClockZoneAt}(t_{s2},\mathcal{D})$ where $t_{s2}=1$ is $e_1 = 1 \wedge e_2 = 1 \wedge e_1 - e_2 = 0$.

| $\mathcal{D}$ | | | | $\mathcal{D}'$ | | | |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | | 0 | 1 | 2 |
| 0 | (0, $\leq$) | (0, $\leq$) | (0, $\leq$) | 0 | (0, $\leq$) | (-1, $\leq$) | (-1, $\leq$) |
| 1 | (2, $\leq$) | (0, $\leq$) | (0, $\leq$) | 1 | (1, $\leq$) | (0, $\leq$) | (0, $\leq$) |
| 2 | (1, $\leq$) | (0, $\leq$) | (0, $\leq$) | 2 | (1, $\leq$) | (0, $\leq$) | (0, $\leq$) |

*3) Invariant Clock Zone after a State Transition:* Regardless of internal or external transition, when a state transition occurs, some clocks can be reset as well as states and their schedules can be updated. Thus with the resetting clocks set $R$ and the updated state and schedule vector $(\mathsf{s},\mathsf{t}) = (\ldots,(s_i,t_{si}),\ldots)$, we can calculate the new clock zone as shown in Algorithm 2. The procedure is (1) extending $R$ so that $R$ includes the passive subcomponent whose schedule is infinite; (2) resetting the previous clock zone $\mathcal{D}$ with $R$; (3) getting all possible clock zone $\mathcal{D}'$ of the next state and schedule vectors; (4) taking intersection $\mathcal{D}$ and $\mathcal{D}'$ and sliding the clock up to the boundary of $\mathcal{D}'$. [7]

**Algorithm 2** $\mathsf{MakeInvariantClockZone}(R,(\mathsf{s},\mathsf{t}),\uparrow \mathcal{D})$
```
1: For each if $t_{si}=\infty$, add $M_i$ to $R$;
2: $\mathcal{D}$=$\mathsf{Resetting}(\mathcal{D},R)$;
3: DBM $\mathcal{D}'$ = $\mathsf{MakeWidestClockZone}((\mathsf{s},\mathsf{t}))$;
4: $\mathcal{D} = \mathsf{Sliding}(\mathcal{D}\cap\mathcal{D}')\cap\mathcal{D}'$;
```

For example, let's consider a DBM $\mathcal{D}$ for the clock zone $e_1 = 1 \wedge e_2 = 1 \wedge e_1 - e_2 = 0$ and suppose that the resetting clock set $R=\{e_2\}$ and the next state-schedule vector $(\mathsf{s},\mathsf{t}) = ((s_1,2),(s_2',4))$[8]. Then the consequent matrices are following:

| $\mathcal{D}$ | | | | $\mathcal{D}_1 = \mathsf{Resetting}(\mathcal{D},R)$ | | | |
|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | | 0 | 1 | 2 |
| 0 | (0, $\leq$) | (-1, $\leq$) | (-1, $\leq$) | 0 | (0, $\leq$) | (-1, $\leq$) | (0, $\leq$) |
| 1 | (1, $\leq$) | (0, $\leq$) | (0, $\leq$) | 1 | (1, $\leq$) | (0, $\leq$) | (1, $\leq$) |
| 2 | (1, $\leq$) | (0, $\leq$) | (0, $\leq$) | 2 | (0, $\leq$) | (-1, $\leq$) | (0, $\leq$) |
| $\mathcal{D}_2 = \mathsf{MakeWidestClockZone}$ | | | | $\mathcal{D}_3 = \mathcal{D}_1\cap\mathcal{D}_2$ | | | |
| | 0 | 1 | 2 | | 0 | 1 | 2 |
| 0 | (0, $\leq$) | (0, $\leq$) | (0, $\leq$) | 0 | (0, $\leq$) | (-1, $\leq$) | (0, $\leq$) |
| 1 | (2, $\leq$) | (0, $\leq$) | (2, $\leq$) | 1 | (1, $\leq$) | (0, $\leq$) | (1, $\leq$) |
| 2 | (4, $\leq$) | (4, $\leq$) | (0, $\leq$) | 2 | (0, $\leq$) | (-1, $\leq$) | (0, $\leq$) |
| $\mathcal{D}_4 = \mathsf{Sliding}(\mathcal{D}_3)$ | | | | $\mathcal{D}_5 = \mathcal{D}_4\cap\mathcal{D}_2$ | | | |
| | 0 | 1 | 2 | | 0 | 1 | 2 |
| 0 | (0, $\leq$) | (-1, $\leq$) | (0, $\leq$) | 0 | (0, $\leq$) | (-1, $\leq$) | (0, $\leq$) |
| 1 | ($\infty$, $\leq$) | (0, $\leq$) | (1, $\leq$) | 1 | (2, $\leq$) | (0, $\leq$) | (1, $\leq$) |
| 2 | ($\infty$, $\leq$) | (-1, $\leq$) | (0, $\leq$) | 2 | (1, $\leq$) | (-1, $\leq$) | (0, $\leq$) |

Here, $\mathcal{D}_1$ is the DBM for $e_1 = 1, e_2 = 0, e_1 - e_2 = 1$; $\mathcal{D}_2$ for $0 \leq e_1 \leq 2, 0 \leq e_2 \leq 4, -4 \leq e_1 - e_2 \leq 2$; $\mathcal{D}_3$ for $e_1 = 1, e_2 = 0, e_1 - e_2 = 1$; $\mathcal{D}_4$ for $1 \leq e_1 \leq \infty, 0 \leq e_2 \leq \infty, e_1 - e_2 = 1$; $\mathcal{D}_5$ for $1 \leq e_1 \leq 2, 0 \leq e_2 \leq 1, e_1 - e_2 = 1$;

## C. Reachable Graph of Coupled FD-DEVS

A zone is a node of a reachable graph of a coupled FD-DEVS model, while its edges are state transitions. Formally, given a coupled FD-DEVS, $N =< X, Y, D, C_{xx}, C_{yx}, C_{yy} >$, the *reachable graph* of $N$ is

$$RG(N) =< Z, V, v_0, E > \tag{4}$$

where

- $Z = X \bigcup_{M_i \in D} Y_i \cup \{\epsilon\}$ is the set of triggering events.
- $V$ is a set of zones. A zone $v = ((\ldots,(s_i,t_{si}),\ldots),\mathcal{D})$ consists of a state-scheduler vector and a clock zone.
- $v_0 \in V$ is the initial zone such that $v = ((\ldots,(s_{0i},\tau_i(s_{0i})),\ldots),\mathcal{D}^0)$ where $\mathcal{D}^0$ is achieved

[7]For the canonical form, we need apply Tightening operation but we omit here.

[8]There is no additional inclusion of $R$ for the infinite schedule. We will take a look at it in Example 2.

by

$$\mathcal{D}^0[i,j] = \begin{cases} (t_{si}, \leq) & \text{for } i \neq 0, j = 0 \\ (0, \leq) & \text{otherwise} \end{cases}$$

and Tightening($\mathcal{D}^0$). Notice that initially there is no difference between any two clocks.

- $E \subseteq V \times Z \times V$ is a transition relation that satisfies the following property: For $q = (\ldots, (s_i, t_{si}, e_i), \ldots)$, $q' = (\ldots, (s'_i, t'_{si}, e'_i), \ldots)$ and $z \in Z$,
$$\delta(q, z) = q' \Leftrightarrow (v, z, v') \in E$$
such that $v = ((\ldots, (s_i, t_{si}), \ldots), \mathcal{D}), \forall e_i \in \mathcal{D}$ and $v' = ((\ldots, (s'_i, t'_{si}), \ldots), \mathcal{D}'), \forall e'_i \in \mathcal{D}'$.

## D. Algorithm of Reachable Graph Generation

Simply speaking, the overall procedure of generating a reachable graph for a coupled FD-DEVS $N$ can be stated as: find a set of next possible zones created by state transitions.

Algorithm 3 shows the main procedure of generating $RG =< Z, V, v_0, E >$ from $N =< X, Y, D, C_{xx}, C_{yx}, C_{yy} >$. Given a zone $v = ((\ldots, (s_i, t_{si}), \ldots), \mathcal{D})$, we indicate the state-schedule vector and the clock zone by using two functions:

- disc$(v) = (\ldots, (s_i, t_{si}), \ldots)$ for the discrete zone, i.e., the state-schedule vector, and
- clock$(v) = \mathcal{D}$ for the clock zone represented as a DBM.

Algorithm 4 explains the procedure handling an event $z$ that can be an output event of a sub-component $i$ or an external input event of $x$ of $N$.

**Algorithm 3** GeneratingReachableGraph($N, \uparrow RG$)

```
1: $v_0 = ((\ldots, (s_{0i}, \tau_i(s_{0i})), \ldots), \mathcal{D}^0)$;
2: $V_T := \emptyset$; Add $v_0$ to $V_T$;
3: while $V_T \neq \emptyset$ do
4:     $v$ = pop_front($V_T$);
5:     for all $i \in D$ do
6:         if clock$(v)[i,0] = (t_{si}, \leq)$ and $t_{si} \neq \infty$ then
7:             $v_n$ := copy($v$);
8:             $y = \lambda_i(s_i)$;
9:             MakeClockZoneAt($t_{si}$, clock($v_n$));
10:            disc$(v_n)[i] := (\delta_{\tau,i}(s_i), \tau_i(\delta_{\tau,i}(s_i)))$;
11:            $R := \emptyset$; Add $i$ to $R$;
12:            WhenReceive-z($N, v_n, y, R, V_T, RG$);
13:        end if
14:    end for
15:    for all $x \in X$ do
16:        $v_n$ := copy($v$);
17:        WhenReceive-z($N, v_n, x, R := \emptyset, V_T, RG$);
18:    end for
19: end while
```

As we mentioned in Section II-B.2, the state transitions of a coupled FD-DEVS $N$ is categorized into two cases: internal transition triggering or external transition triggering. So the procedure of GeneratingReachableGraph (Algorithm 3) searches all possible states by triggering an enable internal transition (in lines 5 to 14) as well as an external transition (in lines 15 to 18) until state are no longer generated by checking if $V_T$ is empty (line 3) where $V_T$ contains a set of zones which are needed to be tested. Of course, the starting point of this searching is the initial zone $v_0$ (lines 1 and 2).

**Algorithm 4** WhenReceive-z($N, v, z, R, \uparrow V_T, \uparrow RG$)

```
1: for all $(z, x_i) \in C_{yx}$ or $(z, x_i) \in C_{xx}$ do
2:     if $\delta_{x,i}(s_i, x_i)$ is defined then
3:         if $\rho_i(s_i, x_i) = 1$ then
4:             disc$(v_n)[i] := (\delta_{x,i}(s_i, x_i), \tau_i(\delta_{x,i}(s_i, x_i)))$;
5:             Add $i$ to $R$;
6:         else
7:             disc$(v_n)[i] := (\delta_{x,i}(s_i, x_i), t_{si})$;
8:         end if
9:     end if
10: end for
11: MakeInvariantClockZone($R$, disc$(v_n)$, clock$(v_n)$);
12: if $\nexists v' \in RG.V$ s.t. disc$(v_n)$ = disc$(v')$ $\wedge$ clock$(v_n) \subseteq$ clock$(v')$ then
13:     Add $v_n$ to $RG.V$ and $V_T$.
14: end if
15: Add $(v, x, v_n)$ to $RG.E$;
```

While $V_T$ is not empty, we pick the first stored zone $v$ (pop_front($V_T$) function (line 4) returns the first zone and remove it from $V_T$) and then attempt to generate reachable zones by testing the internal transitions (lines 5 to 14) and the external transitions (lines 15 to 18).

The enable condition of the internal transition of the $i$-th component is checked by testing if the elapsed time of $i$ in the clock zone of $v$, which is represented as clock$(v)[i,0]$, is able to reach at its schedule time $t_{si}$ that is less than $\infty$ (line 6 of Algorithm 3). If the enable condition is satisfied, the followings are performed: copying $v$ to $v_n$ this is the candidate of the next vertex (at line 7); storing the output event generated by $\lambda_i(s_i)$ to $y$ (at line 8); making the clock zone of $v_n$ when $e_i$ reach at $t_{si}$ by calling MakeClockZoneAt (at line 9); updating $(s_i, t_{si})$ of $v_n$ (that is represented as disc$(v_n)[i]$) (line 10); gathering $i$ into the set of resetting clocks, $R$ (line 11); generating the next reachable zone by calling WhenReceive-z($N, v, y, R, V_T, RG$) (line 12). We will look inside of WhenReceive-z later.

The procedures generating a reachable zone triggered by each external event $x$ is considered in lines 16 to 18. Unlikely to the internal transition, we don't need to consider generating output and updating the discrete stat and the clock zone by the enabled schedule. Thus what we should do here are to copy $v_n$ from $v$ (line 16) and to call WhenReceive-z($N, v, x, R, V_T, RG$) (line 17).

In WhenReceive-z$(N, v, z, R, V_T, RG)$ updates the discrete parts of zone $v_n$ if $i$ is an influencee of event $z$ (lines 1 and 2 of Algorithm 4). Here, updating the schedule $t_{si}$ is only performed when a reschedule is required, that is checked by $\rho_i(s_i, x_i) = 1$. If it is needed, it will be also an element of the resting clock set, $R$. With considering of the resetting clock set $R$ and the newly updated discrete vector of $v_n$, the clock zone of $z_n$ is updated by calling MakeInvariantClockZone (line 11).

If there is no zone $z'$ having the same discrete state as that of $v_n$ such that $\text{disc}(v') = \text{disc}(v_n)$ as well as containing the clock zone of $v_n$ such that $\text{clock}(v_n) \subseteq \text{clock}(v')$ (line 12), the reachable region of $v_n$ has not been visited nor tested. Thus, we add $v_n$ to both of the reachable zone set, $RG.V$, and the testing candidate set, $V_T$. (lines 13). Finally, the transition from $v$ to $v_n$ by triggering event $z$ is added to the set of transitions $RG.E$ (line 15).

*1) Completeness:*

*Lemma 1:* Given a coupled FD-DEVS model $N$, GeneratingReachableGraph generates $RG(N)$.

*Proof:* By Definition of the state transition of coupled FD-DEVS (see Section II-B.2) and Definition of its reachable state graph (see Section III-C). ■

*2) Complexity and Termination:* In the main procedure, GeneratingReachableGraph, the *while* loop continues until there is no further new zone. Thus the complexity is strongly related to the number of all possible zones generated.

Let's check the complexity of the discrete part of zones. For one atomic FD-DEVS $M_i$, the number of possible combinations of state and schedule $(s_i, t_{si})$ is bounded to $|S_i| \times |S_i|$ because the schedule $t_{si} = \tau_i(s_i)$ applies not only to $s_i$ but also to successors of $s_i$ for which a continue holds, i.e., $\forall s_j \in S_i$ such that there is an incoming external transition to $s_i = \delta_{xi}(s_j, x)$ with $\rho_i(s_j, x) = 0$. Thus if we have a set of atomic FD-DEVS models under consideration in $D$, $\prod_{M_i \in D} |S_i|^2$.

Let's check the upper bound of the number of possible different clock zone $\mathcal{D}$ for a give zone $v = ((\ldots, (s_i, t_{si}), \ldots), \mathcal{D})$. First of all, let $g \in \mathbb{Q}_{[0,\infty)}$ be the *greatest common divisor* such that $g * n_{si} = \tau_i(s_i)$ for all $s_i \in \bigcup_{M_i \in D} S_i$ if $\tau_i(s_i) \neq 0$ nor $\tau_i(s_i) \neq \infty$ where $n_{si} \in \mathbb{N}$ (natural numbers).

Then, a number of $t_{si}/g$ different lower bounds are possible. The same possibilities are applied to the upper bound. In other words, a number of $(t_{si}/g)^2$ possible intervals are possible for $t_{si}$. Moreover, given two clocks $e_i$ and $e_j$ whose corresponding schedule are $t_{si}$ and $t_{sj}$, respectively, the number of all possible combinations of lower and upper bound for $e_i - e_j$ is $(t_{si}/g + t_{sj}/g)^2$.

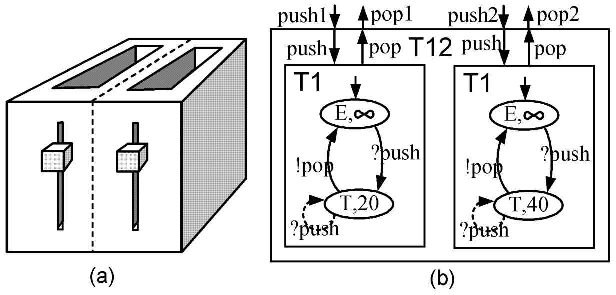

Fig. 3. (a) Tow-Slot Toaster (b) Coupled FD-DEVS

Generally, given $v = ((\ldots, (s_i, t_{si}), \ldots), \mathcal{D})$, the number of possible different clock zones is bounded by $\prod_{M_i \in D} (t_{si}/g)^2 \times \prod_{M_i, M_j \in D} (t_{si}/g + t_{sj}/g)^2$. Thus the number of zones of $RG(N)$ is bounded by $\prod_{i \in D} |S_i|^2 \times \prod_{M_i \in D} (t_{si}/g)^2 \times \prod_{M_i, M_j \in D} (t_{si}/g + t_{sj}/g)^2$. [9]

Since the number of zones is bounded, GeneratingReachableGraph's iteration which tests every single zone until a new zone is generated is terminated.

*Example 2 (Generating Reachable Graph):* Now we consider a two-slot toaster as shown in Figure 3(a) whose first slot has its 20 sec. toasting time while the second slot has the time as 40 sec. We can build a coupled FD-DEVS as shown in Figure 3(b) for the two-slot toaster. Since at initial discrete state $((\texttt{E}, \infty)(\texttt{E}, \infty))$, both $e_1$ and $e_2$ start from 0 and can reach up to $\infty$ without difference $e_1 - e_2 = 0$, the initial zone is the same as (1) in Figure 4.

Since at the initial zone, both slots `T1` and `T2` have no enable internal transition so there is no state generation of the internal transition. However, an external transition, $\delta_2((\texttt{E}, \infty), e), \texttt{push2})$ is able to occur when one pushes the second slot. This external transition generates a next state whose discrete state vector is $((\texttt{E}, \infty)(\texttt{T}, 40))$, and its clock zone $\mathcal{D}$ is can be built by Algorithm MakeInvariantClockZone($\{\texttt{T2}\}$,$((\texttt{E}, \infty)(\texttt{T}, 40))$,$\mathcal{D}$) as follows: (1) $R$ includes $\{\texttt{T1}\}$ because of `T1`'s infinite schedule so $R$ becomes $\{\texttt{T1, T2}\}$; (2) Resetting($\{\texttt{T1, T2}\}$,$\mathcal{D}$) updates $\mathcal{D}$ as $0 \leq e_1 \leq 0, 0 \leq e_2 \leq 0, 0 \leq e_1 - e_2 \leq 0$, i.e. $e_1 = e_2 = e_1 - e_2 = 0$. MakeWidestClockZone($((\texttt{E}, \infty)(\texttt{T}, 40))$) generates $\mathcal{D}'$ as $0 \leq e_1 \leq \infty, 0 \leq e_2 \leq 40, -40 \leq e_1 - e_2 \leq \infty$. $\mathcal{D} \cap \mathcal{D}'$ is $0 \leq e_1 \leq 0, 0 \leq e_2 \leq 0, 0 \leq e_1 - e_2 \leq 0$ again. Slide($\mathcal{D} \cap \mathcal{D}'$) makes the upper bounds of all clocks infinite but preserves the differences of clocks such as $0 \leq e_1 \leq \infty, 0 \leq e_2 \leq \infty, 0 \leq e_1 - e_2 \leq 0$. The final clock zone is calculated by Slide($\mathcal{D} \cap \mathcal{D}'$) $\cap \mathcal{D}'$ as

[9] The number of edges, $|E|$ is is bounded by $|V| \times |Z| \times |V|$.

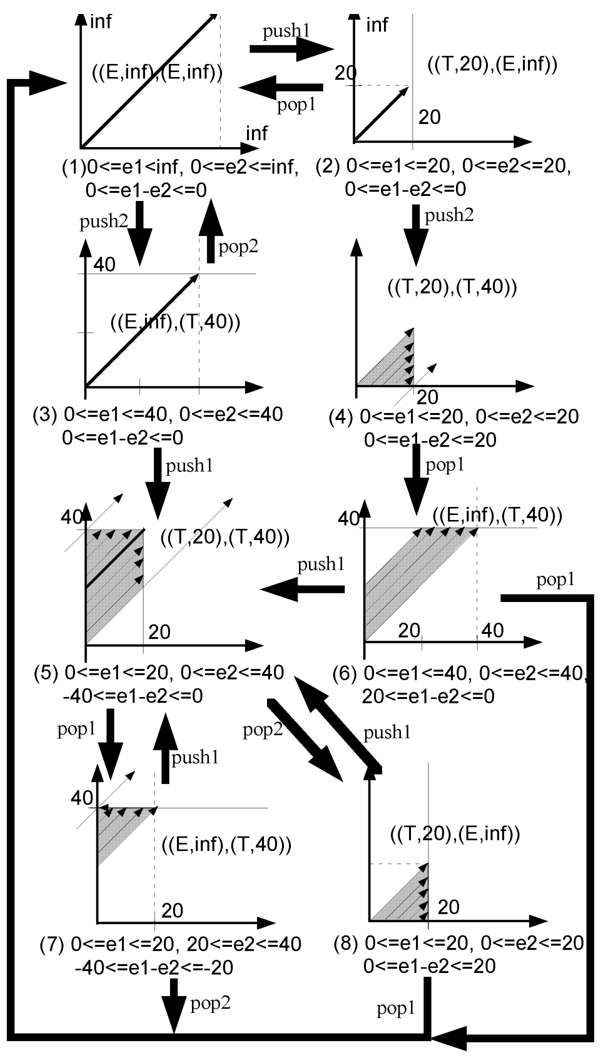

Fig. 4. Reachable Graph of Two-Slot Toaster

$0 \le e_1 \le 40, 0 \le e_2 \le 40, 0 \le e_1 - e_2 \le 0$ in the canonical form. Thus the consequent zone is zone (3) in Figure 4.

If one pushes the first slot at this zone (3), the discrete state changes into $((\mathtt{T}, 20)(\mathtt{T}, 40))$ and its clock zone is calculated by MakeInvariantClockZone($\{\mathtt{T1}\}$,$((\mathtt{T}, 20)(\mathtt{T}, 40))$,$\mathcal{D}$): (1) $R$ remains $\{\mathtt{T1}\}$ because $t_{s2} = 40 < \infty$. (2) Resetting($\{\mathtt{T1}\}$,$\mathcal{D}$) updates $\mathcal{D}$ as $0 \le e_1 \le 0, 0 \le e_2 \le 40, -40 \le e_1 - e_2 \le 0$. MakeWidestClockZone($((\mathtt{T}, 20)(\mathtt{T}, 40))$) generates $\mathcal{D}'$ as $0 \le e_1 \le 20, 0 \le e_2 \le 40, -40 \le e_1 - e_2 \le 20$. $\mathcal{D} \cap \mathcal{D}'$ is $0 \le e_1 \le 0, 0 \le e_2 \le 40, -40 \le e_1 - e_2 \le 0$ again. Slide($\mathcal{D} \cap \mathcal{D}'$) is $0 \le e_1 \le \infty, 0 \le e_2 \le \infty, -40 \le e_1 - e_2 \le 0$. Thus, the final clock zone Slide($\mathcal{D} \cap \mathcal{D}'$)$\cap \mathcal{D}'$ is $0 \le e_1 \le 20, 0 \le e_2 \le 40, -40 \le e_1 - e_2 \le 0$ and it is zone (5) in Figure 4.

Let's take a look at the internal transition case. At the discrete state of $((\mathtt{T}, 20)(\mathtt{T}, 40))$ there are two internal events `pop1` and `pop2` can be enabled. We will investigate when the first slot pops `pop2` occurs. Unlike the external transition, we need to move the clock zone at the time when the enabled schedule is firing. To do this, we use MakeClockZoneAt($t_{s1} = 20, \mathcal{D}$) that changes $\mathcal{D}$ from $0 \le e_1 \le 20, 0 \le e_2 \le 40, -40 \le e_1 - e_2 \le 0$ to $20 \le e_1 \le 20, 20 \le e_2 \le 40, -20 \le e_1 - e_2 \le 0$. In addition, the discrete state gets into $((\mathtt{E}, \infty)(\mathtt{T2}, 40))$ and the resetting sets $R$ is updated as $R = \{\mathtt{T1}\}$. Once again we calculate the next zone caused by `pop1` using MakeReceive-z. Since these two slots toast independently, there is no influencee from `pop1` so only thing we need is to update the clock zone by MakeInvariantClockZone. The next clock zone is calculated as follows: (1) $R$ remains $\{\mathtt{T1}\}$ because $t_{s2} = 40 < \infty$. (2) Resetting($\{\mathtt{T1}\}$,$\mathcal{D}$) updates $\mathcal{D}$ as $0 \le e_1 \le 0, 20 \le e_2 \le 40, -40 \le e_1 - e_2 \le -20$. MakeWidestClockZone($((\mathtt{E}, \infty)(\mathtt{T}, 40))$) generates $\mathcal{D}'$ as $0 \le e_1 \le \infty, 0 \le e_2 \le 40, -40 \le e_1 - e_2 \le \infty$. $\mathcal{D} \cap \mathcal{D}'$ is $0 \le e_1 \le 0, 20 \le e_2 \le 40, -40 \le e_1 - e_2 \le -20$. Slide($\mathcal{D} \cap \mathcal{D}'$) is $0 \le e_1 \le \infty, 0 \le e_2 \le \infty, -40 \le e_1 - e_2 \le -20$. The final clock zone Slide($\mathcal{D} \cap \mathcal{D}'$)$\cap \mathcal{D}'$ is $0 \le e_1 \le 20, 20 \le e_2 \le 40, -40 \le e_1 - e_2 \le -20$ and it is illustrated as zone (7) in Figure 4.

We can apply this procedure until no new state is generated and we get the reachable state graph as Figure 4. □

## IV. Remarks on the Complexity of $RG(N)$

Let's take a close look at factors that can increase or decrease the complexity of $RG(N)$ in terms of the number of zones $|V|$.

- As we can see the previous section, $|V|$ generally increases when the number of subcomponent $|D|$ and states $|S_i|$ of each $M_i \in D$, and the ratio of $\tau_i(s_i)/g$ for each $s_i$ are increasing.
- If the states of all sub-components of $N$ are changed simultaneously, $N$ is said to be *synchronized*. Figure 5(a) and Figure 5(b) show an asynchronized system and a synchronized system, respectively, while their corresponding reachable graph are Figure 5(c) and Figure 5(d), respectively. We can imagine that if we modify $\tau_2(A) = \tau_2(B) = 0.1$ then the number of zones of Figure 5(c) can increase to 400, while that is constant in Figure 5(d). Thus, generally, asynchronized systems generate more zones than synchronized systems.
- If $N$ has no external transition by any $x \in X$, $N$ is said to be *closed*. Generally, the interaction with external influences causes the increasing number of zones and the non-determinism. The non-

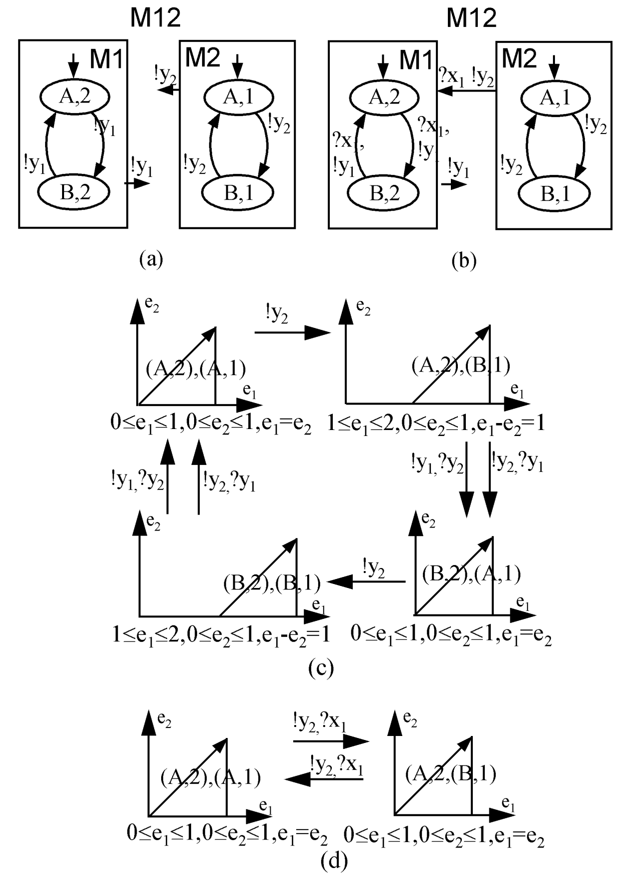

Fig. 5. (a) Asynchronized System, (b) Synchronized System, (c) RG of (a), (d) RG of (b)

determinism observed in the two-slot toaster of Example 2 are two things: multiple internal transitions (and outputs) and multiple values of time advance. For example, zone (5) of Figure 4 has tow possible internal transitions with outputs as `pop1` and `pop2`, respectively. In addition, a gray zone of Figure 4 has multiple values of its life time, for instance, zone (4) can stay there 0 to 20 sec.

- Depending on the order of testing transitions, the different reachable graph can be generated. For example, if we trace zone (8) of Figure 4 earlier than zone (2), the generation of zone (2) can be omitted.

## V. Conclusion and Further Research

This paper proposed a subclass of DEVS, called FD-DEVS. Comparing to the ordinary DEVS, FD-DEVS might have less expressive power, but has an advantage of the achievability of the finite reachable graph. To present the reachable graph, we used the difference bound matrix that had been originally proposed by Dill [3] to represent the clock zone that is a conjunction of inequalities of clocks.

Based-on the clock zone, we proposed a generating algorithm for the reachable graph of a coupled FD-DEVS. The completeness and the complexity of the algorithm were addressed. In addition, the factor increasing the number of zones in a reachable graph was also investigated.

This reachable graph is expected to check the qualitative properties such as safety, liveness, and fairness properties as well as quantitative property such as the time critical response property.

## Acknowledgment

This work was supported by the Korea Research Foundation Grant (No: M01-2004-000-20045-0).

## References

[1] R. Alur. Timed Automata. *11th International Conference on Computer-Aided Verification, LNCS*, 1633:8–22, 1999.

[2] R. Alur, C. Courcoubetis, D.L. Dill, N. Halbwachs, and H. Wong-Toi. An implementation of three algorithms for timing verfication based on automata emptiness. In *Proceedings of the 13th IEEE Real-Time Systems Symposium*, pages 157–166, 1992.

[3] David L. Dill. Timming Assumptions and Verification of Finite-State Concurrent Systems. In *Proc. of the Workshop on Computer Aided Verification Methods for Finite State Systems*, pages 197–212, Grenoble, France, 1989.

[4] J.S. Hong, H.S. Song, T.G. Kim, and K.H. Park. RT-DEVS Executive: A Seamless Realtime Software Development Framework. *Discrete Event Dyanmic Systems*, 7:355–375, 1997.

[5] K.J. Hong and T.G. Kim. Timed I/O Test Sequences for Discrete Event Model Verification. In *13th International Conference on AI, Simulation, and Planning in High Autonomy Systems*, volume 3397 of *LNCS*, pages 257–284. Springer, 2005.

[6] M.H. Hwang. Generating Behavior Model of Coupled SP-DEVS. In *Proceedings of 2005 DEVS Integrative M & S Symposium*, pages 90–97, San Diego, CA, April 2005. SCS.

[7] M.H. Hwang. Generating Finite-State Behavior of Reconfigurable Automation Systems: DEVS Approach. In *Proceed. of 2005 IEEE-CASE*, pages Edmonton,Canada. IEEE, 2005.

[8] M.H. Hwang. Tutorial: Verification of Real-time System Based on Schedule-Preserved DEVS. In *Proceedings of 2005 DEVS Symposium*, San Diego, CA, Apr. 2-8 2005. SCS.

[9] M.H. Hwang and S.K. Cho. Timed Analysis of Schedule Preserved DEVS. In A.G. Bruzzone and E. Williams, editors, *2004 Summer Computer Simulation Conference*, pages 173–178, San Jose, CA, 2004. SCS.

[10] M.H. Hwang and B.P. Zeigler. A Modular Verification Framework using Finite & Deterministic DEVS. In *Proceedings of 2006 DEVS Symposium*. SCS, http://www.u.arizona.edu/∼mhhwang, 2006. Submitted.

[11] E.M. Clarke Jr., O.Grumberg, and D.A. Peled. *Model Checking*. MIT Press, first edition, 1999.

[12] R. Sedgewick. *Algorithms in C++, Part 5 Graph Algorithm*. Addison Wesley, Boston, third edition, 2002.

[13] H.S. Song and T.G. Kim. Application of Real-Time DEVS to Analysis of Safety-Critical Embedded Control Systems: Railroad Crossing Control Example. *SIMULATION*, 81(2):119–136, Feb. 2005.

[14] B. P. Zeigler and S.D. Chi. Symbolic Discrete Event System Specification. *IEEE Transactions on Systems, Man, and Cybernetics*, 22(6):1428–1443, Nov./Dec. 1992.

[15] Bernard P. Zeigler. *Theory of Modelling and Simulation*. Wiley Interscience, New York, first edition, 1976.

[16] B.P. Zeigler, H.Praehofer, and T.G. Kim. *Theory of Modelling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems*. Academic Press, London, second edition, 2000.