Reachability Graph of Finite & Deterministic DEVS Networks

Moon Ho Hwang, *Member, IEEE,* and Bernard P. Zeigler, *Fellow, IEEE,*

***Abstract*— This paper shows how to generate a finite-vertex graph, called a reachability graph for discrete event system specification (DEVS) network. The reachability graph is isomorphic to a given original DEVS network in terms of behavior but the number of vertices as well as the number of edges of the reachability graph are finite.**

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

For abstracting the infinite-state behavior of DEVS network, we use the concept of time zone, invented by Dill [3], that is a conjunction of inequalities of elapsed times. Based-on time zone abstraction, an algorithm for generating the reachability graph of a DEVS network is proposed and its completeness and complexity are addressed.

Questions concerning qualitative properties, for examples, "Does this DEVS network have any possibility to reach a bad situation?" or "Will this system repeat a certain pattern forever?" are open problems for more than 30 years. This paper gives an answer about the above questions for the FD-DEVS subclass of DEVS. A reachability graph-based qualitative verification is exemplified with a modular monorail system, so the reader will find the usefulness of the reachability graph.

***Note to Practitioners*— Modular and hierarchical modeling and analysis becomes more important as systems are increasingly complicated [10]. DEVS formalism is a modular and hierarchical formalism in which the user build a system by connecting system components, and the system can be a component in a bigger system. In addition, the practitioners can use the all source codes of the algorithm and the verification example proposed in this paper which are available at http://xsy-csharp.sourceforge.net/DEVSsharp.**

***Index Terms*— Discrete Event System Specification (DEVS), Finite-Vertex Reachability Graph, Time Abstraction, Verification Qualitative Properties**

I. Introduction

Verification of discrete event systems has been researched based on the assumption that the target system has a finite state space [2]. However, when analyzing a dense-time system in which a state transition is able to occur at any real-valued time instance, we can encounter an infinite state problem [3]. In particular, DEVS [15] [14] has as its time advance function a mapping from states to real numbers as well as the fact that elapsed time is reset whenever any external state occurs. These features cause an infinite state problem, especially when building a network of DEVS models.

Several papers have been published on the problem of obtaining a fini-vertex reachability graph of a DEVS network. Reference [13] developed a symbolic representation of the time advance mechanism of the DEVS formalism and proposed a reachability tree for such symbolic DEVS networks. Reference [5] showed how to get a timed state reachability graph from an ordinary coupled DEVS. Reference [12] used Real-time DEVS (RT-DEVS) [4] whose time advance is a mapping from states to intervals of real numbers. All of the above, however, assume that the target system is a closed system that is not interacting with external influences. Therefore, achieving finiteness with real-valued time is an open problem in DEVS.

Dropping the closed system assumption, schedule-preserving DEVS (SP-DEVS) [9] characterizes an open system whose external state transitions are allowed at any time. One advantage of coupled SP-DEVS is the a finite number of vertices for its reachability can be obtained by a time abstracting technique, called the time-line abstraction. This property enables decidability of qualitative analysis (such as deadlock, livelock, and fairness) as well as quantitative analysis (such as processing time bounds). However, there is also a critical limitation on modeling expressiveness: "once an SP-DEVS model becomes passive, it never returns to become active (OPNA)" [7].

In this paper, we define a class of DEVS, called finite and deterministic DEVS (FD-DEVS). This class is defined by the following: (1) the sets of events and states are finite, (2) the time advance is a mapping from states to non-negative rational numbers, (3) there is no restriction on the occurrences of external events, and (4) an external input event can either reschedule or continue processing. The main restriction imposed here is that there can be no use of the time that has elpased in a state to determine its transition to another state. (A characteristic of the general DEVS formalism is that such elapsed time information can be employed in a unrestricted manner.) This restricted class of DEVS has no OPNA problem and is applicable to a wide variety of control problems as will be illustrated here.

Unfortunately, time-line abstraction is no longer guaranteed to produce a finite reachability graph[7]. Instead of using the time-line abstraction, this paper uses a time zone abstraction that was invented by Dill [3]. Based-on the time zone abstraction, we propose an algorithm for generating finite-vertex reachability graphs of the coupled FD-DEVS models.

This paper is organized as follows: Section II introduces the structure and the behavior of atomic as well as coupled FD-DEVS models. Review of the time zone concept, an algorithm for generating reachability graphs for FD-DEVS coupled

Manuscript received mm dd, 2007; revised mm dd, 200x.

Hwang and Zeigler are with the Department of Electrical and Computer Engineering at the University of Arizona.

models (networks), as well as the algorithm's completeness and complexity are addressed in Section III. Section IV illustrates the expressive power of FD-DEVS and how to use the reachability graph of a FD-DEVS network to analyze the qualitative properties of a modular monorail system. Finally, conclusions are given in Section V.

II. 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.

A. Atomic FD-DEVS

1) Structure of Atomic FD-DEVS:

Definition 1: An *atomic FD-DEVS* is a 7-tuple,

$$M =< X, Y, S, s_0, \tau, \delta_x, \delta_y >$$

where • X (res. Y) is a finite set of *input events* (res. *output events*); • S is a finite set of *states*; • $s_0 \in S$ is the initial state; • $\tau : S \to \mathbb{Q}_{[0,\infty]}$ is the *time advance function* where $\mathbb{Q}_{[0,\infty]}$ is the set of non-negative rational numbers plus infinity. This function is used to determine the lifespan of a state; • $\delta_x : S \times X \to S \times \{0, 1\}$ is the *external state transition function* that defines how an input event changes a state, and whether the internal schedule will be updated or not; • $\delta_y : S \to Y^\phi \times S$ is the *output and internal state transition function* where $Y^\phi = Y \cup \{\phi\}$ and $\phi \notin Y$ denotes the *silent event*. This output and internal state transition function defines how a state generates an output event and, at the same time, how it changes the state internally. This function can be invoked when the elapsed time reaches the lifespan this is scheduled by τ; [1] ■

The behavior of a given FD-DEVS model is defined a set of event sequences which can change the state of the model or can be generated by the model. The next section defines how the atomic FD-DEVS behaves.

2) Behavior of Atomic FD-DEVS: Suppose that $A =< X, Y, S, \tau, \delta_x, \delta_y >$ is an atomic FD-DEVS model. The *time base* of DEVS, denoted by $\mathbb{T}$, is the set of non-negative real numbers such that $\mathbb{T} = [0, \infty)$. An *elapsed time* $t_e \in \mathbb{T}$ is continuously increasing and its value denotes the time passage since $t_e = 0$.

In addition to t_e, we need to consider one more internal state variable, called a *lifetime* or a *schedule time span* which is denoted by $t_s \in \mathbb{Q}_{[0,\infty]}$. When we consider $t \in \mathbb{T} \cup \{\infty\}$ as a upper limit of t_e, the existing range of t_e is defined by the function $tr : \mathbb{T} \cup \{\infty\} \to 2^{\mathbb{T}}$ such that

$$tr(t) = \begin{cases} [0, t] & \text{if } t < \infty \\ [0, \infty) & \text{otherwise} \end{cases} \quad (1)$$

where '[' and ']' are closed boundaries, while ')' is an open boundary. Thus, if $t = \infty$, t_e cannot reach t.

Let $Q_{\mathsf{p}} = \{(s, t_s, t_e) | s \in S, t_s \in \mathbb{Q}_{[0,\infty]}, t_e \in tr(t_s)\}$ be the *set of legal states*, and $Q_{\mathsf{imp}} = \{(\mathsf{imp}, \infty, t_e) | \mathsf{imp} \notin S, t_e \in \mathbb{T}\}$ be the *set of illegal states* s.t. $Q_{\mathsf{p}} \cap Q_{\mathsf{imp}} = \varnothing$. Then *total state set* Q is defined as $Q = Q_{\mathsf{p}} \cup Q_{\mathsf{imp}}$.

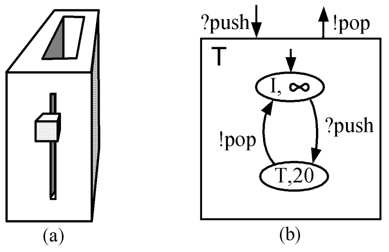

Fig. 1. (a) One-slot Toaster (b) Atomic FD-DEVS Model

Let the *total event set* of A be $Z = X \cup Y^\phi$. Then the *total state function* $\delta : Q \times Z \to Q$ defines state transitions over all combinations of states and events such that for $q = (s, t_s, t_e) \in Q$ and $z \in Z$, $\delta(q, z) =$

$$\begin{cases} (s', \tau(s'), 0) & \text{if } q \in Q_{\mathsf{p}}, z \in X, \delta_x(s, z) = (s', 1) \\ (s', t_s, t_e) & \text{if } q \in Q_{\mathsf{p}}, z \in X, \delta_x(s, z) = (s', 0) \\ (s', \tau(s'), 0) & \text{if } q \in Q_{\mathsf{p}}, z \in Y^\phi, t_e = t_s, \delta_y(s) = (z, s') \\ (\mathsf{imp}, \infty, t_e) & \text{otherwise} \end{cases} \quad (2)$$

Example 1 (Atomic Model for Toaster): Let's consider a toaster as shown in Figure 1(a). This toaster has one slot of bread and a start knob. Initially, the knob is not pushed. But if we push the knob, the toaster starts toasting for 20 seconds, and then it pops up.

We can model the push-knob event as an input event `?push` and the pop-up event as an output event `!pop`. [2] A FD-DEVS model, T, for the one-slot toaster is as follows. $A_{\mathsf{T}} =< X, Y, S, s_0, \tau, \delta_x, \delta_y >$ such that X={`?push`}; Y={`!pop`}; S={`I`, `T`} where `I` and `T` stand for "Idle" and "Toast", respectively; s_0=`I`; $\tau(\texttt{I}) = \infty$, $\tau(\texttt{T}) = 20$; $\delta_x(s, \texttt{?push}) = (\texttt{T}, 1)$, otherwise, $\delta_x(s, \texttt{?push}) = (s, 0)$. $\delta_y(\texttt{T}) = (\texttt{!pop}, \texttt{T})$, $\delta_y(\texttt{I}) = (\epsilon, \texttt{I})$.

Figure 1(b) illustrates an atomic FD-DEVS model for T in which a circle denotes a state s and its lifespan given by $\tau(s)$, and a directed arc indicates a state transition either caused by an input event or accompanied by an output event. For simplicity, the cases of $\delta_x(s, x) = (s, 0)$ such as $\delta(\texttt{T}, \texttt{?push}) = (\texttt{T}, 0)$ and $\delta_y(s)$ where $\tau(s) = \infty$ such as $\delta_x(\texttt{I}) = (\epsilon, \texttt{I})$ are omitted in the diagram. We will use this drawing convention throughout this paper.

Let's take a look at the dynamics of this toaster T. If T has its current total state q=(`I`, ∞, 43), it means T has been idling for 43 seconds, and the remaining time to change to the next state internally is $\infty (= \infty - 43)$. Suppose that we push the knob when the total state is $q = (\texttt{I}, \infty, 43)$. Then the next total state is determined by $\delta((\texttt{I}, \infty, 43), \texttt{?push}) = (\texttt{T}, 0, 0)$ as the first condition of Equation (2). On the other hand, if we push the knob when the total state is $q = (\texttt{T}, 20, 11)$, the next state is δ((`T`, 20, 11),`?push`)=(`T`, 20, 11) as the second condition of Equation (2). If the state is `T` and its elapsed time reaches the lifespan 20, then the output and the next state can

[1] δ_y can be split into two functions: the output function $\lambda : S \to Y$ and the internal transition function $\delta_{int} : S \to S$ as in [14].

[2] To distinguish event types, this paper uses '?' before an input event, and '!' before an output event

be described by $\delta((\texttt{T}, 20, 20), !\texttt{pop}) = (\texttt{I}, \infty, 0)$ as the third condition of Equation (2). Since it is impossible to execute an output and internal state transition when $t_e \neq t_s$, $\delta((\texttt{T}, 20, 14), !\texttt{pop})$=(imp, ∞, 14) as the last condition of Equation (2). ■

We define a state trajectory over an event sequence. Let Z^* be the Kleene closure [3] over a total event set Z, and let $[t_l, t_u] \subseteq \mathbb{T}$ be an *observing time interval*. Then the set of *total event sequences*, denoted by $\Omega_{[t_l,t_u]}$, considers the pair of an event and its time of occurrence such that $\Omega_{[t_l,t_u]} = Z^* \times [t_l, t_u]$. We define the time passage $t \in T$ of $q = (s, t_s, t_e) \in Q$ as

$$q + t = (s, t_s, t_e + t).$$

A *state trajectory* is defined by the function $\Delta : Q \times \Omega_{[t_l,t_u]} \to Q$ such that for $q = (s, t_s, t_e) \in Q$, $t \in [t_l, t_u]$, $\omega, \omega' \in \Omega_{[t_l,t]}$ and $z \in Z$,

$$\Delta(q, \omega) = \begin{cases} q + t_l - t & \text{for } \omega = \epsilon_{[t_l,t]} \\ \delta(\Delta(q, \omega'), z) & \text{for } \omega = \omega'(z, t) \end{cases} \tag{3}$$

where $\epsilon_{[t_l,t]}$ denotes the *null-event sequence* over $[t_l, t]$.

Based on the state trajectory function, the behavior of a given atomic FD-DEVS A is defined as the all possible event sequences which make the state stay in a legal state. Formally, the *behavior or language* of A over an finite observation length $t \in \mathbb{T}$, denoted by $L(A, t)$, is

$$L(A, t) = \{\omega \in \Omega_{[0,t]} | \Delta(q_0, \omega) \in Q_{\mathsf{p}}\} \tag{4}$$

where $q_0 = (s_0, \tau(s_0), 0)$ is the initial total state. The *infinite-length behavior or language* of A, denoted by $L(A)$, is

$$L(A) = \{\omega \in \Omega_{[0,\infty)} | \Delta(q_0, \omega) \in Q_{\mathsf{p}}\}. \tag{5}$$

Given a FD-DEVS model, T, introduced in Example 1, $\omega_{[0,30]} = (?\text{push}, 5)(!\text{pop}, 25)$, $\omega_{[0,30]} \in L(\mathsf{T}, 30)$ because $\Delta((\texttt{I}, \infty, 0), \omega_{[0,30]}) = (\texttt{I}, \infty, 5) \in Q_{\mathsf{p}}$. However, $\omega'_{[0,30]} = (?\text{push}, 5)(!\text{pop}, 28)$, $\omega'_{[0,30]} \notin L(\mathsf{T}, 30)$ because $\Delta((\texttt{I}, \infty, 0), \omega_{[0,30]})' = (\text{imp}, \infty, 2) \in Q_{\mathsf{imp}}$.

B. Coupled FD-DEVS

1) Structure of Coupled FD-DEVS: The coupled FD-DEVS defines a network structure whose sub-components are FD-DEVS.

Definition 2: A *coupled FD-DEVS* is a 5-tuple,

$$N = < X, Y, D, C_x, C_y >$$

where • X and Y are *finite sets of input and output events*, respectively such that $X \cap Y = \varnothing$; • D is a finite set of *names of sub-components*; • $\{M_i\}$ is an index set of *FD-DEVS models* where $i \in D$. M_i can be either an atomic FD-DEVS model or a coupled FD-DEVS model; • $C_x \subseteq X \times \bigcup_{i \in D} X_i$ is an *set of input couplings* where X_i is the set of input events of sub-component $i \in D$; • $C_y \subseteq \bigcup_{i \in D} Y_i \times (\bigcup_{j \in D} X_i \cup Y)$ where $i \neq j$ is an *set of output couplings* where Y_i is the set of output events of sub-component $i \in D$. ■

Notice that the coupled FD-DEVS has a hierarchical structure because its sub-components can be coupled FD-DEVS itself. Since we can get a flatten coupled FD-DEVS model from a hierarchical coupled FD-DEVS model [8], we would restrict each subcomponent $i \in D$ to be an atomic FD-DEVS model without loss of generality.

2) Behavior of Coupled FD-DEVS:

Definition 3 (Behavioral Structure of Coupled FD-DEVS): Given a FD-DEVS network, $N =< X, Y, D, \{M_i\}, C_x, C_y >$, its behavioral structure is defined as a 7-tuple:

$$A_N =< X, \cup Y_i^\phi, S, s_0, \tau, \delta_x, \delta_y >$$

where • X is the input event set of N; • $\cup Y_i^\phi = \bigcup_{i \in D} Y_i^\phi$ is the sum of all output events set (including the silent event) of sub-components; • $S = \underset{i \in D}{\times} Q_{\mathsf{p}i}$ is a set of states where $Q_{\mathsf{p}i}$ is the set of legal states of component i; • $s_0 = (\ldots, (s_{0i}, \tau_i(s_{0i}), 0), \ldots)$ is the initial state; • The time advance function $\tau : S \to \mathbb{T}$ is for $s = (\ldots, (s_i, t_{si}, t_{ei}), \ldots)$, $\tau(s) = \min\{t_{si} - t_{ei} | i \in D\}$. • The external state transition $\delta_x : S \times X \to S \times \{0, 1\}$ is for $s = (\ldots, (s_i, t_{si}, t_{ei}), \ldots)$ and $x \in X$

$$\delta_x(s, x) = ((\ldots, (s'_i, t'_{si}, t'_{ei}), \ldots), \rho) \tag{6}$$

where $(s'_i, t'_{si}, t'_{ei}) =$

$$\begin{cases} (s'_i, \tau_i(s'_i), 0) & \text{if } (x, x_i) \in C_x, \delta_{xi}(s_i, x_i) = (s'_i, 1) \\ (s'_i, t_{si}, t_{ei}) & \text{if } (x, x_i) \in C_x, \delta_{xi}(s_i, x_i) = (s'_i, 0) \\ (s_i, t_{si}, t_{ei}) & \text{otherwise} \end{cases} \tag{6a}$$

and

$$\rho = \begin{cases} 1 & \text{if } \exists i \text{ s.t. } (x, x_i) \in C_x, \delta_{xi}(s_i, x_i) = (s'_i, 1) \\ 0 & \text{otherwise.} \end{cases} \tag{6b}$$

• The output and internal transition function $\delta_y : S \to \cup Y_i^\phi \times S$ is defined as follows. Given $s = (\ldots, (s_i, t_{si}, t_{ei}), \ldots) \in S$, let $IMM(s) = \{i \in D | t_{si} - t_{ei} = \tau(s)\}$ be the set of imminent components of s. If $\exists i^* \in IMM(s)$ s.t. $\delta_{yi^*}(s_{i^*}) = (y_{i^*}, s'_{i^*})$

$$\delta_y(s) = (y_{i^*}, (\ldots, (s'_i, t'_{si}, t'_{ei}), \ldots)) \tag{7}$$

where $(s'_i, t'_{si}, t'_{ei}) =$

$$\begin{cases} (s'_{i^*}, \tau_{i^*}(s'_{i^*}), 0) & \text{if } i = i^* \\ (s'_i, \tau_i(s'_i), 0) & \text{if } (y_{i^*}, x_i) \in C_x, \delta_{xi}(s_i, x_i) = (s'_i, 1) \\ (s'_i, t_{si}, t_{ei}) & \text{if } (y_{i^*}, x_i) \in C_x, \delta_{xi}(s_i, x_i) = (s'_i, 0) \\ (s_i, t_{si}, t_{ei}) & \text{otherwise.} \end{cases} \tag{7a}$$

■

Let $Z = X \cup \bigcup_{i \in D} Y_i^\phi$ be the total event set, $Q_{\mathsf{p}} = \{(s, t_s, t_e) | s \in S, t_s \in \mathbb{T}^\infty, t_e \in tr(t_s)\}$ be the set of legal states, $Q_{\mathsf{imp}} = \{(\ldots, (s_i, t_{si}, t_{ei})), \ldots) | \exists i \in D \text{ s.t. } s_i = \text{imp}_i, t_{si} = \infty, t_{ei} \in tr(\infty)\}$ be the set of illegal states s.t. $Q_{\mathsf{p}} \cap Q_{\mathsf{imp}} = \varnothing$. Then the total state of N is $Q = Q_{\mathsf{p}} \cup Q_{\mathsf{imp}}$.

[3] Z^* is the set of a *finite-length* event sequences over Z[6]. For example, if $Z = \{\phi, a, b\}$, then $Z^* = \{\epsilon, \phi, a, b, \phi\phi, aa, bb, \phi a, \phi b, a\phi, ab, b\phi, ba, \ldots\}$ where ϵ is the *null-event* that is indicating nothing happens. Thus ϵ is distinguished from the silent event ϕ which denotes that there is an state transition that generates the unobservable output event.

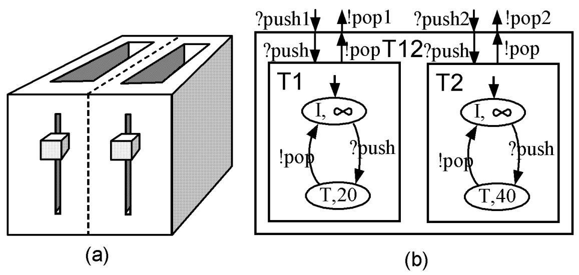

Fig. 2. (a) Tow-Slot Toaster (b) Coupled FD-DEVS

We won't repeat to define then the total state transition function $\delta : Q \times Z \to Q$, the state trajectory function $\Delta : Q \times \Omega_{[t_l,t_u]} \to Q$, and the language of N over an finite observation length, $L(N,t)$ and the infinite length language, $L(N)$ for the coupled FD-DEVS N because their definitions will be the same as Equations (2),(3),(4), and (5), respectively. In order to define the state trajectory function of coupled FD-DEVS N, we need the time passage $t \in \mathbb{T}$ of $q \in Q$ as follows. If $s = (\ldots,(s_i,t_{si},t_{ei}),\ldots) \in S$ and $q = (s,t_s,t_e) \in Q$, their time passage t are defined as

$$s + t = (\ldots,(s_i,t_{si},t_{ei}+t)\ldots)$$

and

$$q + t = (s+t, t_s, t_e + t).$$

Example 2 (Coupled Model for Two-Slot Toaster): Consider a two-slot toaster as shown in Figure 2(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, named T12, as shown in Figure 2(b) such that $N_{\mathsf{T12}} =< X, Y, D, \{M_i\}, C_x, C_y >$ where $X = \{\texttt{?push1}, \texttt{?push2}\}, Y = \{\texttt{!pop1}, \texttt{!pop2}\}, D = \texttt{T1}, \texttt{T2}$, M_{T1} and M_{T2} are drawn in Figure 2(b), $C_x = \{(\texttt{?push1}, \texttt{?T1.push}), (\texttt{?push2}, \texttt{?T2.push})\}, C_y = \{(\texttt{!T1.pop}, \texttt{!pop1}), (\texttt{!T2.pop}, \texttt{!pop2})\}$.

Given a timed event sequence $\omega_{[0,70]} = (\texttt{?push1},5)(\texttt{?push2},20)(\texttt{!pop1},25)(\texttt{!pop2},60)$, $\omega_{[0,70]} \in L(\mathsf{T12},70)$ because $\Delta((((\texttt{I},\infty,0),(\texttt{I},\infty,0)),\infty,0),\omega_{[0,70]}) = (((\texttt{I},\infty,45), (\texttt{I},\infty,10)),\infty,10) \in Q_{\mathsf{p}}$. However, if $\omega'_{[0,70]} = (\texttt{?push1},5)(\texttt{?push2},20)(\texttt{!pop1},30)(\texttt{!pop2},60)$, $\omega'_{[0,70]}$ is not in $L(\mathsf{T12},70)$ because $\Delta((((\texttt{I},\infty,0),(\texttt{I},\infty,0)),\infty,0),\omega'_{[0,70]}) = (((\mathsf{imp},\infty,45), (I,\infty,10)),\infty,10) \in Q_{\mathsf{imp}}$. In other words, it is impossible to occurs $\omega'_{[0,70]}$ in T12. ■

III. Reachablility Graph of FD-DEVS Network

We would abstract uncountably-many combinations of elapsed times into a *time zone* which was introduced by Dill [3], and used for abstracting real-time in Timed Automata [1], [2]. We reviews the time zone and its basic operations first and introduces the operations for tracing the FD-DEVS behavior. Finally, we will see an algorithm constructing a reachability graph of FD-DEVS networks.

We would focus our attention on the coupled FD-DEVS class because an atomic FD-DEVS model can be seen as a coupled FD-DEVS model whose D has only the atomic FD-DEVS model.

A. Review of Time Zone and Its Basic Operations

Recall the elapsed time t_{ei} of $i \in D$ is an non-negative real number. For describing the bounds of t_{ei} in FD-DEVS, we use a real-number interval denoted by $b(i)$, which is bounded by two rational numbers such that if $l \leq t_{ei} \leq u \Leftrightarrow b(i) = [l,u]$ where $l,u \in \mathbb{Q}_{[-\infty,\infty]}$. [4] The difference bound of two t_{ei} and t_{ej} are also defined in the same way such that if $l \leq t_{ei} - t_{ej} \leq u \Leftrightarrow b(i,j) = [l,u]$. By its definition, if $b(i) = [l,u]$ then $-b(i) = [-u,-l]$. Similarly, $b(i,j) = [l,u]$, then $b(j,i) = [-u,-l]$.

If $|D| = n$, we can think n-dimensional Euclidean space of non-negative real numbers plus infinity, denoted by $\mathbb{T}^{\infty,n}$. A *time zone* $\varphi \subseteq \mathbb{T}^{\infty,n}$ is a convex polyhedron that is represented by a conjunctions of bounds of (1) all elapsed times and (2) differences of between elapsed times. Formally,

$$\varphi = \bigwedge_{i\in D} b(i) \ \wedge \bigwedge_{i,j\in D, i\neq j} b(i,j).$$

For example, the initial elapsed time zone of T12 in Example 2 is $\varphi = (0 \leq t_{e1} \leq \infty, 0 \leq t_{e2} \leq \infty, t_{e1} - t_{e2} = 0)$ which means that t_{e1} and t_{e2} keep increasing forever if there is no push events.

In this section we will use $\varphi = \bigwedge_{i\in D} b(i) \ \wedge \bigwedge_{i,j\in D,i\neq j} b(i,j)$ for unary operations, and two time zones $\varphi_1 = \bigwedge_{i\in D} b_1(i) \ \wedge \bigwedge_{i,j\in D,i\neq j} b_1(i,j)$ and $\varphi_2 = \bigwedge_{i\in D} b_2(i) \ \wedge \bigwedge_{i,j\in D,i\neq j} b_2(i,j)$ for binary operations.

1) Tightening: A time zone φ is *tight* if $\forall i,j \in D, i \neq j$

$$[lb(i) - ub(j), ub(i) - lb(j)] = [lb(i,j), ub(i,j)]$$

where $ub(i)$ and $lb(i)$ are the upper bound and the lower bound of t_{ei}, while $ub(i,j)$ and $lb(i,j)$ are the upper bound and the lower bound of $t_{ei} - t_{ej}$, respectively.

The tightening operation of φ, denoted by $\mathsf{Tightening}(\varphi)$, makes all boundary constraints in φ tightened so there is no loose inequalities. Thus $\mathsf{Tightening}(\varphi)$ is defined for all $i,j \in D$, $i \neq j$,

$$\begin{array}{l}
\text{if } lb(i) < ub(j) + lb(i,j) \Rightarrow lb(i) := ub(j) + lb(i,j)\\
\text{if } ub(j) > lb(i) - lb(i,j) \Rightarrow ub(j) := lb(i) - lb(i,j)\\
\text{if } lb(i,j) < lb(i) - ub(j) \Rightarrow lb(i,j) := lb(i) - ub(j)\\
\text{if } ub(i) > lb(j) + ub(i,j) \Rightarrow ub(i) := lb(j) + ub(i,j)\\
\text{if } lb(j) < ub(i) - ub(i,j) \Rightarrow lb(j) := ub(i) - ub(i,j)\\
\text{if } ub(i,j) > ub(i) - lb(j) \Rightarrow ub(i,j) := ub(i) - lb(j).
\end{array} \tag{8}$$

The first, second, and third statements of Equation (8) come from $lb(i) - ub(j) \lesseqgtr lb(i,j)$, while the rest other statements of the equation from $ub(i) - lb(j) \lesseqgtr ub(i,j)$.

[4] The boundary condition can be either open or closed. But for simplicity, we would use the closed conditions here. Thus for $t_e \in [0,\infty)$, we will write $t_e \in [0,\infty]$ instead.

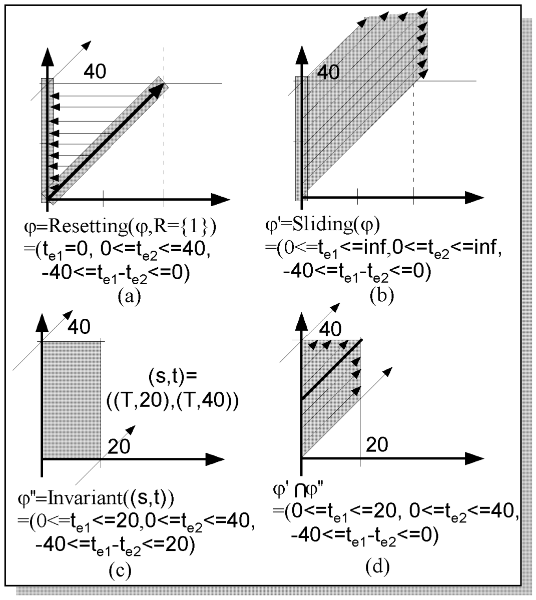

Fig. 3. Successor($\varphi = (0 \le t_{e1} \le 20, 0 \le t_{e2} \le 40, 0 \le t_{e1} - t_{e2} \le 0)$, $R = \{1\}$, (s,t) $= ((\mathrm{T}, 20), (\mathrm{T}, 40)))$

2) Equality: Given two intervals $[l_1, u_1]$ and $[l_2, u_2]$, $[l_1, u_1] = [l_2, u_2]$ if $l_1 = l_2$ and $u_1 = u_2$. Given two time zones φ_1 and φ_2, $\varphi_1 = \varphi_2$ if $b_1(i) = b_2(i)$ for all $i \in D$ and $b_1(i,j) = b_2(i,j)$ for all $i, j \in D, i \neq j$.

3) Resetting: The resetting operation of φ w.r.t a name set $R \subseteq D$ resets t_{ei} to zero for each $i \in R$.

$$\mathsf{Resetting}(\varphi, R) = \mathsf{T}(\bigwedge_{i \in R} [0, 0] \ \wedge \bigwedge_{i \notin R} b(i) \ \wedge \bigwedge_{i,j \in D, i \neq j} b(i,j))$$

where T denotes Tightening operation.

Figure 3(a) shows Resetting(φ, $R = \{1\}$) where $\varphi = (0 \le t_{e1} \le 40, 0 \le t_{e2} \le 40, t_{e1} - t_{e2} = 0)$. The resetting can happen if `?push1` occurs when the first slot is empty and the second slot is toasting in toaster T12.

4) Sliding: The sliding operation is used for letting every t_{ei} pass infinitely long from a given time zone. This operation sets each upper bound of t_{ei} infinity, but preserves the rest other boundaries so

$$\mathsf{Sliding}(\varphi) = \bigwedge_{i \in D} [lb(i), \infty] \ \wedge \bigwedge_{i,j \in D, i \neq j} b(i,j).$$

Figure 3(b) shows Sliding(φ) where $\varphi = (t_{e1} = 0, 0 \le t_{e2} \le 40, -40 \le t_{e1} - t_{e2} \le 0)$.

5) Intersection: The intersection of two intervals $[l_1, u_1] \cap [l_2, u_2] = [\max(l_1, l_2), \min(u_1, u_2)]$. Given two time zone φ_1 and φ_2, the intersection of φ_1 and φ_2 is defined as

$$\varphi_1 \cap \varphi_2 = \bigwedge_{i \in D} b_1(i) \cap b_2(i) \wedge \bigwedge_{i,j \in D, i \neq j} b_1(i,j) \cap b_2(i,j).$$

Figure 3(d) illustrates an intersection between $\varphi' = (0 \le t_{e1} \le \infty, 0 \le t_{e2} \le \infty, -40 \le t_{e1} - t_{e2} \le 0)$ and $\varphi'' = (0 \le t_{e1} \le 20, 0 \le t_{e2} \le 40, -40 \le t_{e1} - t_{e2} \le 20)$.

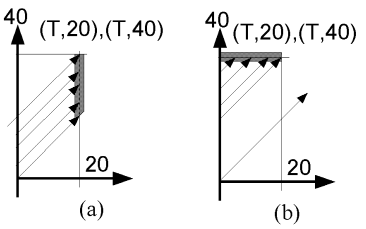

Fig. 4. TimeZoneAt Operation

B. Time Zone Operations for Tracing FD-DEVS Behavior

1) Invariant Time Zone: An *invariant time zone* is a time zone covering all possible elapsed times in a given state-schedule vector (s,t) $= (\ldots, (s_i, t_{si}), \ldots)$ such that

$$\mathsf{Invariant}((\mathsf{s},\mathsf{t})) = \bigwedge_{i \in D} [0, t_{si}] \ \wedge \bigwedge_{i,j \in D, i \neq j} [-t_{sj}, t_{si}].$$

For example, Figure 3(c) shows the invarant time zone of (s,t) $= ((\mathtt{T}, 20), (\mathtt{T}, 40))$ which results in (s,t) $= (0 \le t_{e1} \le 20, 0 \le t_{e2} \le 40, -40 \le t_{e1} - t_{e2} \le 20)$.

Given a coupled FD-DEVS N and its state $q = (\ldots, (s_i, t_{si}, t_{ei}), \ldots)$, the elapsed time vector $(\ldots, t_{ei}, \ldots) \in$ Invariant$((\ldots, (s_i, t_{si}), \ldots))$.

2) Successor Time Zone after a State Transition: An *successor time zone* is defined as φ's successive time zone by resetting each t_{ei} in $R \subseteq D$, and then by letting the time zone grow within an invariant time zone of the next state-schedule vector (s,t). Formally, this operation is defined as

$$\mathsf{Successor}(\varphi, R, (\mathsf{s},\mathsf{t})) = \mathsf{T}(\mathsf{S}(\mathsf{R}(\varphi, R)) \cap \mathsf{I}((\mathsf{s},\mathsf{t})))$$

where R, S, I, and T strand for Resetting, Sliding, Invariant, and Tightening operations, respectively.

Figure 3s show the successor time zone from $((\mathtt{E}, \infty, [0, 40]), (\mathtt{T}, 40, [0, 40])$ triggered by the input event `?push1`. To compute Successor(φ), we need to use Resetting, Sliding, Invariant, and Intersection ($\cap$) as shown in Figure 3's (a), (b), (c) and (d), respectively.

3) Time Zone for Enabling δ_{yi}: As mentioned in Equation (2), an output and internal state transition δ_{yi} can be enabled when $t_{ei} = t_{si}$. Thus we need to compute a time zone when $t_{ei} = t_{si}$ before generating the next zone triggered by δ_{yi}. To get this time zone, all t_{ej} for $j \in D$ are passed during t_{ei} reaches t_{si} as

$$\mathsf{TimeZoneAt}(i, t, \varphi) = \mathsf{T}([t, t] \wedge \bigwedge_{j \neq i} b(j) \ \wedge \bigwedge_{i \neq j} b(i,j))$$

where $i \in D$, $t \in Q_{[0,\infty]}$, and T stands for Tightening operation.

Given $\varphi = (0 \le t_{ei} \le 20, 0 \le t_{e2} \le 40, -40 \le t_{e1} - t_{e2} \le 0$ which is a time zone in Figure 3(d), gray areas in Figure 4(a) and (b) show time zones which are achieved by TimeZoneAt$(1, 20, \varphi))$ and TimeZoneAt$(2, = \ 40, \varphi))$, respectively.

C. Reachability Graph of Coupled FD-DEVS

Definition 4: The reachability graph of a coupled FD-DEVS, N, is defined as

$$RG(N) =< Z, V, v_0, E >$$

where

- $Z = X \cup \bigcup_{i \in D} Y_i^{\phi}$ is the set of triggering events.
- V is a set of zones. A zone $v = ((\ldots, (s_i, t_{si}), \ldots), \varphi)$ is a pair of a state-schedule vector $(\ldots, (s_i, t_{si}), \ldots)$ and a time zone φ.
- $v_0 = ((\ldots (s_{0i}, \tau_i(s_{0i})), \ldots), \varphi_0) \in V$ is the initial zone where $\varphi_0 = \mathsf{Successor}(\varphi, R, (\mathsf{s,t}))$ where $\varphi = \bigwedge_{i \in D} [0,0] \wedge \bigwedge_{i,j \in D, i \neq j} [0,0]$, $R = \varnothing$, $(\mathsf{s,t}) = (\ldots, (s_{0i}, \tau_i(s_{0i})), \ldots)$ is the initial time zone.
- $E \subseteq V \times Z \times 2^D \times V$ is a transition relation. An edge $(v, z, R, v') \in E$ represents that if an event z occurs, and it changes the state-schedule from v to that of v', and reset t_{ei} for each $i \in R$. ■

We will see the behavior of $RG(N)$ in Definition 5 later.

D. Algorithm for Generating Reachability Graph

Given a zone $v = ((\ldots, (s_i, t_{si}), \ldots), \varphi)$, we will use the following notations: $\mathsf{disc}(v) = (\mathsf{s,t}) = (\ldots, (s_i, t_{si}), \ldots)$ for the discrete zone, i.e., the state-schedule vector; $\mathsf{disc}(v)[i] = (s_i, t_{si})$ for the pair of state and schedule of component i; and $\mathsf{clock}(v)[i].ub(i)$ for the upper bound of the elapsed time of component i.

Algorithm 1 ReachabilityGraph($N, \uparrow G$)
```
1: $v_0 = ((\ldots, (s_{0i}, \tau_i(s_{0i})), \ldots), \varphi_0)$;
2: $V_T := \varnothing$; Add $v_0$ to $V_T$;
3: while $V_T \neq \varnothing$ do
4:     $v = ((\ldots, (s_i, t_{si}), \ldots), \varphi)$ := pop_front($V_T$);
5:     for all $x \in X$ do
6:         $v_n$ := copy($v$);
7:         WhenReceive-z($N, v_n, x, R^z := \varnothing, V_T, G$);
8:     end for
9:     for all $i \in D$ do
10:        if clock($v$).$ub(i) = t_{si}$ then
11:            $v_n$ := copy($v$);
12:            TimeZoneAt($i, t_{si}$, clock($v_n$));
13:            $\delta_{yi}(s_i) = (y, s_i')$;
14:            disc($v_n$)[$i$] := $(s_i', \tau_i(s_i'))$;
15:            $R^z := \varnothing$; Add $i$ to $R^z$;
16:            WhenReceive-z($N, v_n, y, R^z, V_T, G$);
17:        end if
18:    end for
19: end while
```

Algorithm 1, named ReachabilityGraph, is the main procedure that traces all possible states by triggering an external state transition as well as an output and internal state transition until to-be-tested states in V_T are available. The starting point of this searching is the initial zone v_0 which was defined in Definition 4.

Generating a next reachable zone triggered by each external event $x \in X$ is considered in lines 5 to 8 of Algorithm 1. First, the algorithm copies the candidate of the next vertex v_n from v and calls WhenReceive-z that is Algorithm 2. Let's take a look into Algorithm 2.

Algorithm 2 WhenReceive-z($N, v, z, R^z, \uparrow V_T, \uparrow G$)
```
1: for all $(z, x_i) \in C_y$ or $(z, x_i) \in C_x$ do
2:     $\delta_{x,i}(s_i, x_i) = (s_i', \rho)$;
3:     if $\rho = 1$ then
4:         disc($v_n$)[$i$] := $(s_i', \tau_i(s_i'))$;
5:         Add $i$ to $R$;
6:     else
7:         disc($v_n$)[$i$] := $(s_i', t_{si})$;
8:     end if
9: end for
10: if $R^z \neq \varnothing$ or disc($v$) $\neq$ disc($v_n$) then
11:     For each $i \in D$ if $t_{si} = \infty$ then add $i$ to $R^\infty$;
12:     Successor(clock($v_n$), $R^z \cup R^\infty$, disc($v_n$));
13:     if $\nexists v' \in G.V$ s.t. $v_n = v'$ then
14:         Add $v_n$ to $G.V$ and $V_T$;
15:     Add $(v, z, v_n)$ to $G.E$;
16: end if
```

Algorithm 2 updates an influencee of event z through the coupling (z, x_i) in C_x (at line 1). If there is such a coupling, it computes $\delta_{x,i}(s_i, x_i) = (s_i', \rho)$; if $\rho = 1$, update not only s' but also t_{si} (at line 4), as well as add i to a set R^z (a line 5). If $b = 0$, update only state as s_i' (lines 7).

Algorithm 2 doesn't go further for the case that $R^z = \varnothing$ and $\mathsf{disc}(v) = \mathsf{disc}(v_n)$ which indicates that nothing changes by $z \in X$(line 10). Otherwise, to avoid meaningless time passage of t_{ei} when $t_{si} = \infty$, Algorithm 2 adds i to R^∞ (at line 11) so that t_{ei} will be reset by applying Successor operation with $R = R^z \cup R^\infty$ (at line 12).

If the next zone v_n is newly generated, i.e. $\nexists v' \in G.V$ s.t. $v' = v_n$ [5], Algorithm 2 adds v_n to both the vertex set V of the reachability graph G and the to-be-tested set V_T (lines 13 to 14). In addition, add an edge (v, z, R, v_n) to the edge set E of G except both $R = \varnothing$ and $\mathsf{disc}(v) = \mathsf{disc}(v_n)$ which means there is no change by $z \in X$ (lines 15).

Let's revisit to the output and internal state transition part of Algorithm 1. Unlike to the external transition, recall that the internal transition of component i can be enabled only $t_{ei} = t_{si}$. To check the possibility of $t_{ei} = t_{si}$, Algorithm 1 checks the upper bound of t_{ei} by evaluating $\mathsf{clock}(\mathsf{v}).ub(i) = t_{si}$ at line 6. If this condition is satisfied, Algorithm 1 copies v to v_n, makes the time zone of v_n when t_{ei} reaches t_{si}, calculates the output and the next state of the internal transition as $\delta_{y,i}(s_i) = (y, s_i')$, updates the state and schedule as $(s_i', \tau_i(s_i'))$ of v_n, add i to R^z after making R^z empty, and generates the next reachable zone by calling WhenReceive-z(N, v, y, R^z, V_T, G) as shown in lines 10 to 16 in Algorithm 1.

Example 3 (Reachability Graph of Two-Slot Toaster): Figure 5 illustrates the reachability graph of the two-slot toaster T12 in Example 2. This reachability graph has 8 vertices and 15 edges. Each vertex v shows its state-schedule vector and its time zone drawn in a gray area. For example, the initial zone is $v = ((\mathsf{s,t}), \varphi)$ where $(\mathsf{s,t}) = ((\mathtt{E}, \infty), (\mathtt{E}, \infty))$

[5]We assume that the time zone φ of every zone is tightened. Thus given two zones v_1 and v_2, $v_1 = v_2$ if $\mathsf{disc}(v_1) = \mathsf{disc}(v_2)$ and $\mathsf{clock}(v_1) = \mathsf{clock}(v_2)$.

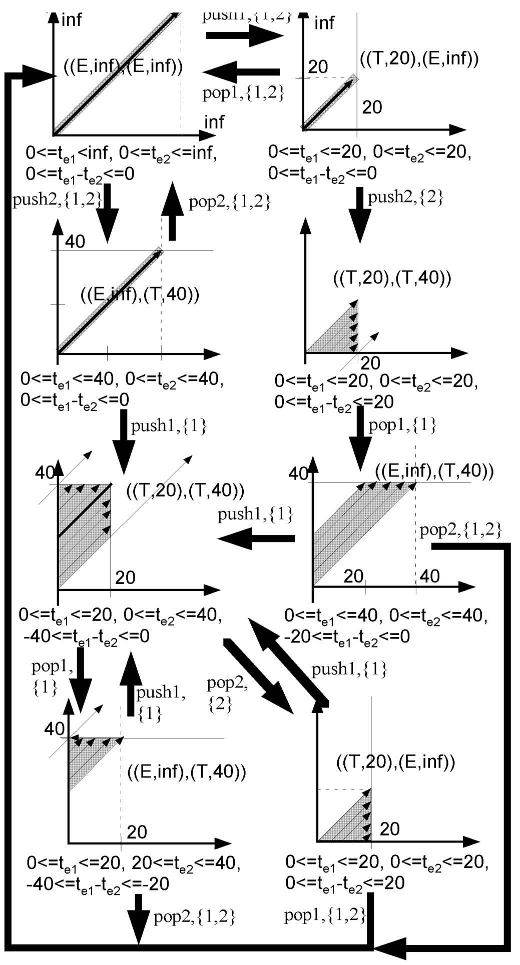

Fig. 5. Reachable Graph of Two-Slot Toaster

and $\varphi = (0 \leq t_{e1} \leq \infty, 0 \leq t_{e2} \leq \infty, 0 \leq t_{e1} - t_{e2} \leq 0)$. Here, each time zone φ is illustrated as the result of $\mathsf{Successor}$ operation, but the steps in $\mathsf{Successor}$ such as shown in Figure 3(a) to (d) are omitted. Each edge $e = (v, z, R, v')$ is illustrated by an directed arc where v and v' are located at the source and the destination of the arc, while the triggering event z and the set of resetting clocks $R = R^z \cup R^\infty$ are augmented around the arc. ■

1) Completeness: To show the completeness of Algorithms 1 and 2, we would show the behavioral equivalence between a given FD-DEVS network N and its reachability graph $RG(N)$. To do this, we first define the behavior of $RG(N)$.

Definition 5 (Behavioral Structure of $RG(N)$): Given a FD-DEVS network, $N =< X, Y, D, \{M_i\}, C_x, C_y >$, and its reachability graph $RG(N)$ is generated, the behavioral structure of $RG(N)$ is defined as a 7-tuple:

$$A_G =< X, \cup Y_i^\phi, S, s_0, \tau, \delta_x, \delta_y >$$

where • X is the input event set of N; • $\cup Y_i^\phi = \bigcup_{i \in D} Y_i^\phi$ is the sum of all output events set (including the silent event) of sub-components; • $S = \{(\dots, (s_i, t_{si}, t_{ei}), \dots) | \mathsf{disc}(v)[i] = (s_i, t_{si}), t_{ei} \in tr(t_{si}), \forall i \in D, v \in V\}$; • $s_0 \in v_0$ s.t. $s_0 = (\dots, (s_{0i}, \tau_i(s_{0i}), 0), \dots)$, $\forall i \in D$ is the initial state; • The time advance function $\tau : S \to \mathbb{T}$ is defined for $s = (\dots, (s_i, t_{si}, t_{ei}), \dots) \in v \in V$, $\tau(s) = \min\{t_{si} - t_{ei} | i \in D\}$; • The external state transition $\delta_x : S \times X \to S \times \{0, 1\}$ is defined as follows. For $x \in X$, if $\exists (v, x, R, v') \in E$, $x \in X$ and $s = (\dots, (s_i, t_{si}, t_{ei}), \dots) \in v$,

$$\delta_x(s, x) = ((\dots, (s'_i, t'_{si}, t'_{ei}), \dots), \rho) \tag{9}$$

where

$$(s'_i, t'_{si}) = \mathsf{disc}(v')[i]; \tag{9a}$$

$$t'_{ei} = 0 \text{ if } i \in R, \quad t'_{ei} = t_{ei} \text{ otherwise}; \tag{9b}$$

$$\rho = 0 \text{ if } R = \varnothing, \quad \rho = 1 \text{ otherwise}. \tag{9c}$$

For $x \in X$, if $\nexists (v, x, R, v') \in E$, then $(s'_i, t'_{si}, t'_{ei}) = (s, t_{si}, t_{ei})$ for all $i \in D$ and $\rho = 0$.

• The output and internal transition function $\delta_y : S \to \bigcup_{i \in D} Y_i^\phi \times S$ is defined as follows. For $z \in \bigcup_{i \in D} Y_i^\phi$, if $\exists (v, z, R, v') \in E$, s.t. $z \in Y_{i*}^\phi$ and $s = (\dots, (s_i, t_{si}, t_{ei}), \dots) \in v$,

$$\delta_y(s) = (z, (\dots, (s'_i, t'_{si}, t'_{ei}), \dots)) \tag{10}$$

where (s'_i, t'_{si}) and t'_{ei} are determined by Equations (9a) and (9b).

For $z \in \bigcup_{i \in D} Y_i^\phi$, if $\nexists (v, z, R, v') \in E$, then $(s'_i, t'_{si}, t'_{ei}) = (\mathsf{imp}_i, \infty, t_{ei})$. ■

The total event set Z and the total state set Q are defined as the same as those of coupled FD-DEVS N. The total state transition δ is defined identically as Equation (3) but by using δ_x in Equation (9) and δ_y in Equation (10).

The state trajectory function Δ, and the language of $RG(N)$ over an finite observation length, denoted by $L(RG(N), t)$, and the infinite length language of $RG(N)$, denoted by $L(RG(N))$ are defined as the same as Equations (3), (4), and (5), respectively.

Lemma 1: $v \in V$ of $RG(N)$, $\mathsf{clock}(v).ub(i) = t_{si} \Leftrightarrow \exists s \in v$ s.t. $i \in IMM(s)$.

Proof: Recall that $IMM(s) = \{i \in D | t_{si} - t_{ei} = \tau(s)\}$.

($\Rightarrow$ case:) Assume that $\mathsf{clock}(v).ub(i) = t_{si}$. That means there is a convex sub-area $a \subseteq v$ such that $\exists s = (\dots, (s_i, t_{si}, t_{ei}), \dots) \in a$, $t_{ei} = t_{si}$. Thus we can say $\exists s \in v$ s.t. $i \in IMM(s)$.

($\Leftarrow$ case:) Suppose that $\exists s$ s.t. $i \in IMM(s)$ but $\mathsf{clock}(v).ub(i) < t_{si}$. Since $\mathsf{clock}(v).ub(i) < t_{si}$, t_{ei} can not be equal to t_{si} because before t_{ei} becomes t_{si} other t_{ej} where $j \neq i$ reaches t_{sj} all the time in v. But it is contradict to $\exists s$ s.t. $i \in IMM(s)$. Lets check the possibility of $\mathsf{clock}(v).ub(i) > t_{si}$. By the $\mathsf{Invariance((s,t))}$ which is used when Algorithm 2 computes $\mathsf{Successor}(\varphi, R,$ (s,t)), there is no possibility $\mathsf{clock}(v).ub(i) > t_{si}$. Thus if $\exists s \in v$ s.t. $i \in IMM(s)$ then $\mathsf{clock}(v).ub(i) = t_{si}$. ■

Definition 6: Let $s = (\dots, (s_i, t_{si}, t_{ei}), \dots), s' = (\dots, (s'_i, t'_{si}, t'_{ei}), \dots) \in S$.Then s is a *schedule equivalent* of s', denoted by $s \cong s'$, if for all $i \in D$, $t_{si} - t_{ei} = t'_{si} - t'_{ei}$. $q =$

$(s,t_s,t_e) \in Q$ is a *schedule equivalent* of $q' = (s',t'_s,t'_e) \in Q$ if $s \cong s'$.

Theorem 1: Let q and q' be the total states of N and $RG(N)$, respectively and $q \cong q'$. Then for $z \in Z$, $\delta(q,z) \cong \delta(q',z)$.

Proof: Let $q = (s^q, t^q_s, t^q_e), s^q = (\ldots,(s^q_i,t^q_{si},t^q_{ei}),\ldots)$, $q' = (s^{q\prime},t^{q\prime}_s,t^{q\prime}_e), s^{q\prime} = (\ldots,(s^{q\prime}_i,t^{q}_{si}{}',t^{q}_{ei}{}'),\ldots)$, and $\exists v^q \in V$ s.t. $s^{q\prime} \in v^q$.

First of all, let's assume that $q,q' \in Q_{\mathsf{imp}}$. Then $\delta(q,z) = q \cong q' = \delta(q',z)$ for any $z \in Z$.

Let's check $q \notin Q_{\mathsf{imp}}$ and $q' \notin Q_{\mathsf{imp}}$. We will investigate $z \in Z$ into three cases (i) $z \in X$, (ii) $z \in Y^\phi_i, t^q_s = t^q_e$, (iii) otherwise.

(i) Thus $z \in X, \delta_x(s^q,z) = (s^r,\rho)$ in N s.t. $s^q = s^r$ and $\rho = 0 \Leftrightarrow \delta_x(s^{q\prime},z) = (s^{r\prime},\rho')$ in $RG(N)$ s.t. $s^{q\prime} = s^{r\prime}$ and $\rho' = 0$ because $\nexists e = (v^q,z,R,v^r) \in E$ s.t. $s^{q\prime} \in v^q \in V$.

If $\delta_x(s^q,z) = (s^r,\rho)$ in N s.t. $s^q \neq s^r \Leftrightarrow \delta_x(s^{q\prime},z) = (s^{r\prime},\rho')$ in $RG(N)$ and $s^{q\prime} \neq s^{r\prime}$ and $\exists e = (v^q,z,R,v^r)$. Since Algorithm 1 copies the next zone v_n from the current zone v, and Algorithm 2 considers the influence of z only when $(z,x_i) \in C_x$. (Case x1:) If $(z,x_i) \notin C_x \Rightarrow ((s^r_i = s^q_i, t^r_{si} = t^q_{si}, t^r_{ei} = t^q_{ei}) \wedge (s^{r\prime}_i = s^{q\prime}_i, t^r_{si}{}' = t^q_{si}{}', t^r_{ei}{}' = t^q_{ei}{}') \Rightarrow t^r_{si} - t^r_{ei} = t^r_{si}{}' - t^r_{ei}{}'$. (Case x2:) If $\exists(z,x_i) \in C_x$ s.t. $\delta_{xi}(s^q_i,x_i) = (s^r_i,1)$ so $r^r_i = (s^r_i,t^r_{si},t^r_{ei})$ where $t^r_{si} = \tau_i(s^r_i)$ and $t^r_{ei} = 0$. At this time $i \in R$ so $r^{r\prime}_i = (s^{r\prime}_i, t^r_{si}{}', t^r_{ei}{}')$ where $t^r_{si} = \tau_i(s^r_i)$ and $t^r_{ei} = 0$. Thus $t^r_{si} - t^r_{ei} = t^r_{si}{}' - t^r_{ei}{}'$. (Case x3:) If $\exists(z,x_i) \in C_x$ s.t. $\delta_{xi}(s^q_i,x_i) = (s^r_i,0)$ so $r^r_i = (s^r_i,t^r_{si},t^r_{ei})$ where $t^r_{si} = t^q_{si}$ and $t^r_{ei} = t^q_{ei} < \infty$. At this time $i \notin R$ so $r^{r\prime}_i = (s^{r\prime}_i, t^r_{si}{}', t^r_{ei}{}')$ where $t^r_{si}{}' = t^q_{si}{}'$ and $t^r_{ei}{}' = t^q_{ei}{}'$. Since $t^q_{si} - t^q_{ei} = t^q_{si}{}' - t^q_{ei}{}'$, $t^r_{si} - t^r_{ei} = t^r_{si}{}' - t^r_{ei}{}'$ too. (Case x4:) If $\exists(z,x_i) \in C_x$ s.t. $\delta_{xi}(s^q_i,x_i) = (s^r_i,0)$ so $r^r_i = (s^r_i,t^r_{si},t^r_{ei})$ where $t^r_{si} = t^q_{si}$ and $t^r_{ei} = t^q_{ei} = \infty$. At this time $i \in R$ so $r^{r\prime}_i = (s^{r\prime}_i, t^r_{si}{}', t^r_{ei}{}')$ where $t^r_{si}{}' = t^q_{si}{}' = \infty$ and $t^r_{ei}{}' = 0$. Since regardless of the value of t^r_{ei}, $t^r_{si} - t^r_{ei} = \infty$, and $t^r_{si}{}' - t^r_{ei}{}' = \infty - 0 = \infty$. Thus $t^r_{si} - t^r_{ei} = t^r_{si}{}' - t^r_{ei}{}'$ too. By Cases x1 to x4, $s^r \cong s^{r\prime}$.

(ii) Suppose that $z \in Y^\phi_i$ and $t^q_{ei} = t^q_{si}$. Observe that $t^q_{ei}{}' = t^q_{si}{}'$ because $t^q_{si} - t^q_{ei} = t^q_{si}{}' - t^q_{ei}{}'$ for $s^q \cong s^{q\prime}$. By Lemma 1, $i \in IMM(s^q) \Leftrightarrow \mathsf{clock}(v^q).ub(i) = t^q_{si}{}'$. By $\mathsf{TimeZoneAt}(i, t_{si}, \mathsf{clock}(v_n))$ at line 12 in Algorithm 1, $t^q_{ei} = t^q_{si} \Leftrightarrow t^q_{ei}{}' = t^q_{si}{}'$.

$\delta_y(s^q) = (z,s^r) \Leftrightarrow \exists(v,z,R,v')$ s.t. $\delta_y(s^{q\prime}) = (z,s^{r\prime})$ and $s^{r\prime} \in v'$. $r^r_i = (s^r_i,t^r_{si},t^r_{ei})$ where $t^r_{si} = \tau_i(s^r_i)$ and $t^r_{ei} = 0 \Leftrightarrow i \in R$ so $r^{r\prime}_i = (s^{r\prime}_i, t^r_{si}{}', t^r_{ei}{}')$ where $s^{r\prime}_i = s^r_i$, $t^r_{si}{}' = \tau_i(s^{r\prime}_i)$ and $t^r_{ei}{}' = 0$. Above Cases x1 to x4 can be applied for calculates the influencees of $z \in Y^\phi_i$. Thus $s^r \cong s^{r\prime}$.

(iii) If $z \in Y^\phi_i$ but $i \notin IMM(s^q)$ then $r^r_i = (\mathsf{imp}_i,\infty,t^q_{ei})$, at the same time, $r^{r\prime}_i = (\mathsf{imp}_i,\infty,t^q_{ei}{}')$ because $\mathsf{clock}(v^q).ub(i) < t^q_{si}{}'$. If $z \in Y^\phi_i$ but $\delta_{yi}(s^q_i) = (y_i,s^r_i), y_i \neq z$ then $r^r_i = (\mathsf{imp}_i,\infty,t^q_{ei})$, at the same time, $r^{r\prime}_i = (\mathsf{imp}_i,\infty,t^q_{ei}{}')$ because $\nexists e = (v^q,z,R,v^r) \in E$. Thus, $s^r \cong s^{r\prime}$. ∎

Lemma 2: If $q = (s,t_s,t_e), q' = (s',t'_s,t'_e) \in Q$ s.t. $q \cong q'$, then $q + t \cong q' + t$ for $t \in tr(\tau(s))$.

Proof: If $\forall i \in D$, $t_{si} - t_{ei} = t'_{si} - t'_{ei} \Rightarrow \forall i \in D$, $t_{si} - (t_{ei}+t) = t'_{si} - (t'_{ei}+t)$. Thus $s \cong s'$, then $s+t \cong s+t$ for $t \in \mathbb{T}$. So $q+t \cong q'+t$ by Definition 6. ∎

Theorem 2: Let q_0 and q'_0 be the initial total states of N and $RG(N)$, respectively. Then for $\omega \in \Omega_{[0,t]}$ where $t \in \mathbb{T}$, $\Delta(q_0,\omega) \cong \Delta(q'_0,\omega)$.

Proof: By Definitions 3 and 5, the initial states of both N and $RG(N)$ as defined as $s_0 = (\ldots,(s_{0i},\tau_i(s_{0i}),0),\ldots)$. $s_0 \cong s'_0$, so $q_0 \cong q'_0$.

Let's assume that $q \cong q'$ and check if $\Delta(q,\omega) \cong \Delta(q',\omega)$ where $\omega \in \Omega_{[0,t]}$ and $t \in tr(\tau(s^{q\prime}))$. [6]

Suppose that $\omega = \epsilon_{[0,t]}$. Then $\Delta(q,\omega) = q+t \cong \Delta(q',\omega) = q'+t$ (by Lemma 2).

Suppose that $\omega = \epsilon_{[0,t]}(z,t)$. Recall that $\Delta(q,\omega) = \delta(q+t,z)$ and $\Delta(q',\omega) = \delta(q'+t,z)$. By Theorem 1, $\delta(q+t,z) \cong \delta(q'+t,z)$ so $\Delta(q,\omega) \cong \Delta(q',\omega)$.

By induction, we can say $\Delta(q_0,\omega) \cong \Delta(q'_0,\omega)$ where $\omega \in \Omega_{[0,t]}, t \in \mathbb{T}$. ∎

Corollary 1: Given $t \in \mathbb{T}, L(N,t) = L(RG(N),t)$. $L(N) = L(RG(N))$.

Proof: By Theorem 2. ∎

2) Complexity and Termination: In the main procedure, $\mathsf{ReachabilityGraph}$, 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. We investigate the bound of the numbers of generating zone as the the complexity of $\mathsf{ReachabilityGraph}$ algorithm in this paper.

For each component 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 predecessors of s_i for which $\delta_{xi}(s_j,x) = (s_i,0)$. Since we have a set of subcomponents D, $\prod_{i \in D} |S_i|^2$.

Let's check the upper bound of the number of possible different time zone φ for a give zone $v = ((\ldots,(s_i,t_{si}),\ldots),\varphi)$. First of all, let $g \in \mathbb{Q}_{[0,\infty)}$ be the *greatest common divisor* of $\tau_i(s_i)$ for all $s_i \in \bigcup_{i \in D} S_i$ where $\tau_i(s_i) \neq 0$ nor $\tau_i(s_i) \neq \infty$. Let's consider the state-schedule of component i as (s_i,t_{si}). Then, there are t_{si}/g different values can be possible for lower bound $lb(i)$, as well as upper bounds $ub(i)$ w.r.t $lb(i) \leq ub(i)$. However, to find the bound, we would introduce a number, t_{max}, such that is $t_{si} \leq t_{max}$ for all i. Let $FS(i) = \{t_{si} | t_{si} < \infty, i \in D\}$ be the set of finite schedule. And $t_{max} := \infty$ if $FS(i) = \varnothing$; otherwise, $t_{max} := \max(FS(i))$. Let's define $k = \infty$ if $t_{max} = \infty$; $k = 0$ if $t_{max} = 0$; otherwise $k = t_{max}/g$ by as a natural number.

Let $\#b(i)$ be *number of possible bounds of* i and $\#b(i,j)$ be the *number of possible difference bounds between i and j.*

Suppose that $k = \infty$ that means $t_{si} = \infty$ for all i. Then $b(i) = [0,\infty]$ so $\#b(i) = 1$. Since Algorithm 2 adds all i to the set R if $t_{si} = \infty$, $b(i,j) = [0,0]$ and $\#b(i,j) = 1$. In this case, the number of possible φ is 1. If $k = 0$ i.e. $t_{si} = 0$ for all i. In this case only possible $b(i) = [0,0]$ and $b(i,j) = [0,0]$. Thus the number of possible φ is 1 again.

Let's consider the case that $0 < t_{max} < \infty$. Recall that there is the constraints $lb(i) \leq ub(i)$. Thus if $lb(i) = k$, only one possible case for $ub(i) = k$; If $lb(i) = k-1 \geq 0$, two possible cases for $ub(i) \in \{k, k-1\}$. Likewise, if $lb(i) = 0$, $ub(i)$ has

[6] Why we can check $\Omega_{[0,t]}$ rather than $\Omega_{[t_l, l+t]}$ where $t_l \in \mathbf{T}$ is that DEVS is a time invariant system, so translating t_l to 0 doesn't make any difference in the DEVS behavior [15],[14].

$k+1$ number of possibilities such as $\{k, k-1, \ldots 1, 0\}$. Thus $\#b(i) = 1+2+\ldots+(k+1) = \sum_{k=1}^{k+1} k = \frac{k^2+3k+2}{2}$.

Unlikely $b(i)$ in which its lower bound cannot less than 0, the lower bound of $b(i,j)$ can reach $-k$ and then the number of all possible combinations of lower and upper bound for $b(i,j)$ is $\#b(i,j) = 1+2+\ldots+(2k+1) = \frac{(2k)^2+3(2k)+2}{2}$. Therefore, in the case $0 < t_{max} < \infty$. If the number of sub-components is $n = |D|$, the number of pairs of i, j is $\frac{n(n-1)}{2}$. Thus, the number of possible different time zone for a state-schedule $(\ldots, (s_i, t_{si}), \ldots)$ is bounded by $\#b(i)^n \times \#b(i,j)^{\frac{n(n-1)}{2}}$.

Therefore the number of vertices $|V|$ is bounded by $\prod_{i \in D} {S_i}^2 \times \#b(i)^n \times \#b(i,j)^{\frac{n(n-1)}{2}}$.

IV. Checking Qualitative Properties by using Reachability Graph

Even though the number of vertices of a reachability graph can be blown up exponentially in the worst case, we would like to show the practical examples so that this reachability graph-based system analysis is not just theoretical.

This section introduces an experimental result to show how to use the reachability graph for checking qualitative properties: safety and liveness. All source codes used here are available in DEVS# version 1.2.3 and later at http://xsy-csharp.sourceforge.net/DEVSsharp/. The time zone was implemented as a matrix form, called a *difference bound matrix* [3]. The hardware platform used was Presario X1000 Laptop with 1.3 GHz CPU and 1.0 GByte RAM.

A. Problem Definitions

Suppose that $N =< X, Y, D, \{M_i\}, C_x, C_y >$ is a coupled FD-DEVS model and Q is its total state set. The *discrete state vector set* of N is defined as $P = \underset{i \in D}{\times} S_i$. Then the discrete state vector function $\mathsf{ds} : Q \rightarrow P$ is defined for $q = (\ldots, (s_i, t_{si}, t_{ei}), \ldots) \in Q$, $\mathsf{ds}(q) = (\ldots, s_i, \ldots)$. Given a discrete state vector $p = (\ldots, s_i, \ldots) \in P$, the discrete state of component i is denoted by $p[i]$ s.t. $p[i] = s_i$.

1) Checking Safety: The safety of a system is defined as the property that all components of the system are guaranteed not go into a unsafe status. Let $S_{\mathsf{u},i} \subseteq S_i$ be the set of unsafe states in component i. Then N is said to be safe if

$$\forall \omega \in L(N), \forall i \in D, \mathsf{ds}(\Delta(q_0, \omega))[i] \cap S_{\mathsf{u},i} = \varnothing.$$

In other words, there is no way to reach an unsafe state in $S_{\mathsf{u},i}$ for each component $i \in D$.

Checking the safety of N can be investigated by checking if each vertex $v \in V$ of $RG(N)$ contains an unsafe state $s_i \in S_{\mathsf{u},i}$ of component i. Notice that the computation of searching a vertex is proportion to $|V|$.

2) Checking Liveness: A liveness of a system is defined as the property that all components of the system are guaranteed to repeat a *work infinitely-many times*. The requirement 'work' can be done as a set of states $S_{\mathsf{w},i} \subseteq S_i$.

Let $c(s_i, \omega)$ be the number of visiting to s_i by counting the number of visiting s_i along $\Delta(q_0, \omega)$ for $\omega \in L(N)$. Then N is said to be alive if

$$\forall \omega \in L(N), \forall i \in D, \forall s_i, \in S_{\mathsf{w},i}, c(s_i, \Delta(q_0, \omega)) = \infty.$$

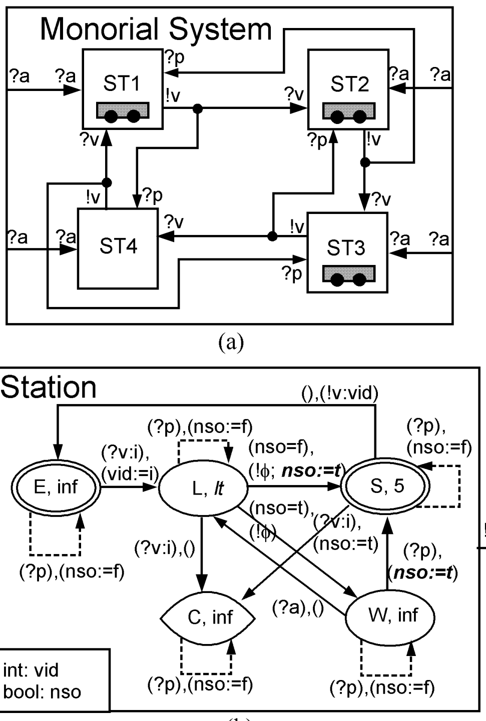

Fig. 6. Monorail System

To check the infinite repeating behavior, we need to build the kernel directed acyclic graph of $RG(N)$, denoted $K(RG(N))$ whose node is a set of strong components of $RG(N))$ [11]. A vertex C of $K(RG(N))$ is said to be *permanent* if C contains more than one vertices of $RG(N)$, or if C has a self loop.

Checking if N is alive can be decided by checking if for each permanent node C of $K(RG(N))$, $S_{w,i} \subseteq s(C,i)$ where $s(C,i) = \{s_i | \mathsf{disc}(v)[i] = (s_i, t_{si}), v \in C\}$. Notice that computing $K(RG(N))$ is bounded by $O(|V| + |E|)$ where $|V|$ and $|E|$ are the number of vertices and the number of edges of $RG(N)$, respectively [11].

B. Experiment using Monorail Systems

Let's consider a monorail system in which stations are connected in a circular railroad so vehicles can circulate along the rail. The number of stations can be varied depending on the size of area the system serves so we will study the safety and the liveness of systems by varying the number of stations from four to eight. Figure 6(a) illustrates an instance of the configurations, which has four stations.

`Station` is a controller modeled by an atomic FD-DEVS whose structure is drawn in Figure 6(b). In the `Station` model, `?v`, `?p`, `?a` stand for input events of *vehicle*, *pull-signal*, *additional-loading*, respectively, while `!v` stands for the *output vehicle* event. There are three state variables: `phase` $\in$ {Empty (`E`), Loading (`L`), Sending (`S`), Waiting (`W`), Collided (`C`)}; `vid`, tracking the vehicle identification; and `nso` $\in$ {false(`f`), true(`t`)} indicating "next station is occupied". To avoid collisions that can occur when more than one vehicle attempts to occupy a station (let's call it A) at the same time, the station prior to A (let's call it B) should dispatch the vehicle ONLY when B's `nso = f`.

In Figure 6(b), an arc is augmented by *(pre-condition),(post-condition)*. For example, when a station receives `?p` at `phase=E`, it makes `nso=f` but its phase doesn't change; After staying in `phase=L` for lt seconds, if `nso=f`, it changes into `phase=S` internally without producing any output indicated by $!\phi$, if `nso=t`, it changes into `phase=W`. A dashed line indicates an external state transition in which $\delta_x(s,x) = (s',0)$ so that the state s change to s' but the schedule t_s and the elapsed time t_e are preserved. All obvious transitions such as $\delta_x(s,x) = (s,0)$ are omitted in Figure 6(b).

The loading time lt is assigned as $lt = 20$ for odd numbered stations, $lt = 40$ for even numbered stations. There are three vehicles that are operated in all configurations. Let n be the number of stations in the configuration. Then the initial state $s_{0i} = (\texttt{phase}, \texttt{id}, \texttt{nso})$ for each station ST_i is determined by $i \in \{1, \ldots, n\}$ where $n \in \{4, 5, 6, 7\}$ as

$$s_{0i} = (\texttt{phase}, \texttt{vid}, \texttt{nso}) = \begin{cases} (\texttt{L}, i, \texttt{t}) & \text{if } i < 3 \\ (\texttt{L}, i, \texttt{f}) & \text{if } i = 3 \\ (\texttt{E}, 0, \texttt{f}) & \text{if } 3 < i < n \\ (\texttt{E}, 0, \texttt{t}) & \text{if } 3 < i = n \end{cases}$$

For each $i \in D$, the set of unsafe states and the set of working states are defined by $S_{\mathsf{u},i} = \{(\texttt{phase}, \texttt{vid}, \texttt{nso}) | \texttt{phase} = \texttt{C}\}$ and $S_{\mathsf{w},i} = \{(\texttt{phase}, \texttt{vid}, \texttt{nso}) | \texttt{phase} = \texttt{E}, \texttt{phase} = \texttt{S}\}$, respectively. Applying the proposed reachability analysis, we found that the monorail systems in all configurations are safe and alive.

To double check that the proposed procedure correctly predicts the system safety, we changed slightly a `Stations`'s state transitions to miss the assignment `nso:= t` at arcs from `L` to `S` and `W` to `S` in Figure 6(b). Under these conditions, the procedure successfully found a Collision state in Station1. For double checking liveness, we disconnected the pull-signal coupling from Station 3 to Station 2. The algorithm correctly found that Station 2 is not alive (this is so, since it stays at `W` and can not repeat visit `E` and `S`).

Table I summarizes the performance from generating the reachability graph of each configuration, to checking safety and liveness. In this experimental study, the time to generate the reachability graph is longer than the time to check safety and liveness. As mentioned in Section III-D.2, the number of vertices of the reachability graph increases with the number of components involved in the system.

TABLE I
PERFORMANCE FOR CHECKING SAFETY AND LIVENESS OF MONORAIL SYSTEMS

n	$\|V\|$	$\|E\|$	Mem.	T_G	T_S	T_L
4	638	1,425	2.6	5.1	0.0	0.2
5	1,574	3,460	3.9	19.8	0.0	0.4
6	3,314	7,276	7.0	1:02.6	0.1	1.2
*6	4,027	7,708	7.7	1:09.2	0.0	1.3
†6	195	478	2.2	2.5	0.0	0.1
7	6,233	13,369	12.2	2:45.8	0.1	3.3
8	10,370	22,264	23.8	6:16.1	0.2	9.5

n: the number of stations used; $|V|$: The number of vertices in $G = RG(N)$; $|E|$: The number of edges in G; Mem: Peak memory required to generate G in M bytes; T_G: CPU time to generate $RG(N)$ in the format, `mm:ss.s`, where `mm` is minutes and `ss.s` is seconds;T_S: CPU time to check the safety;T_L: CPU time to check the liveness;*: the post condition `nso:=t` was missed; †: the coupling between Station 3 and 2 for the pull-signal was disconnected;

V. CONCLUSIONS

This paper proposed a subclass of finite and deterministic DEVS, called FD-DEVS. Compared to its superclass DEVS, FD-DEVS has less expressive power. However, it has the advantage of supporting generation of a finite-vertex reachability graph. The key idea is that infinitely-many instances of elapsed times of components in a FD-DEVS network can be abstracted by a time zone equivalence proposed by Dill [3]. Based on the time zone abstraction technique, a algorithm to generate a finite-vertex reachability graph was introduced and their completeness and complexity were investigated. In addition, a modular monorail system was defined within FD-DEVS and its safety and liveness were analyzed based on its reachability graph. Safety and liveness checking for complex systems of this kind have been resistant to solution with previous approaches using timed-automata and other discrete event formalism.

REFERENCES

[1] R. Alur. Timed Automata. *11th International Conference on Computer-Aided Verification, LNCS*, 1633:8–22, 1999.
[2] E. M. Clarke Jr., O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.
[3] David L. Dill. Timing 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] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, second edition, 2000.
[7] M. H. Hwang. Generating Finite-State Behavior of Reconfigurable Automation Systems: DEVS Approach. In *Proceed. of 2005 IEEE-CASE*, Edmonton,Canada, 2005. IEEE.
[8] M. H. Hwang and B.P. Zeigler. Expressiveness of Verifiable Hierarchical Clock Systems. *International Journal of General Systems*, 2006. to appear, available at http://acims.arizon.edu/.
[9] M. H. Hwang, S.K. Cho, B.P. Zeigler, F. Lin. Processing Time Bounds of Schedule-Preserving DEVS. *Techinical Report-2007-1*, 2007. available at http://acims.arizon.edu/.
[10] R. G. Sargent, J. H. Mize, D. H. Withers, and B. P. Zeigler. Hierarchical Modelling for Discrete Event Simulation (Panel). In *Proceedings of the 25th Winter Simulation Conference*, Los Angeles, CA, 1993. ACM Press.
[11] R. Sedgewick. *Algorithms in C++, Part 5 Graph Algorithm.* Addison Wesley, Boston, third edition, 2002.
[12] 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.
[13] 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.
[14] 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.
[15] Bernard P. Zeigler. *Theory of Modelling and Simulation.* Wiley Interscience, New York, first edition, 1976.