International Journal of General Systems
Vol. 00, No. 00, Month 200x, 1–20

Expressiveness of Verifiable Hierarchical Clock Systems

Moon Ho Hwang* and Bernard P. Zeigler

The Department of Electrical and Computer Engineering, the University of Arizona, Tucson, AZ 85721, USA

(*Received 00 Month 200x; in final form 00 Month 200x*)

The modelling and analysis of multi-component discrete event systems is a challenging research area. Over 30 years, modelling and simulation research of discrete event system specification (DEVS) has been developed with (1) dense-time, (2) the I/O concept, and (3) hierarchical model construction. Nevertheless, DEVS model verification research began relatively recently considering the whole DEVS research history. In the meantime, over 15 years, the automata theory has been developed to cover the dense-time behavior verification of discrete event systems. Especially, timed automata (TA) has performed the key role in the field.

This paper builds on the research results that have been achieved from both theories of DEVS and TA. Thus contributions of this paper can be seen from each side. From the viewpoint of the DEVS theory, a finite and nondeterministic DEVS has been found as a verifiable class. From the view point of the TA theory, a TA which is modular and hierarchical as well as verifiable, is proposed. To show the results, this paper uses the top down manner in which a general formalism is defined first and then its sub-classes are introduced.

Keywords: General Dynamic System, Hierarchical I/O Clock System, DEVS, Timed Automata, Verification

1 Introduction

The state of a dynamic system changes over time. Depending on its dynamic features, we can model it as a continuous system, a discrete system, or a hybrid system that is combining both the continuous system and the discrete events system into a unified system. While it is true that the hybrid model provides a broader modeling expressiveness than the other two individual approaches, in practice, however, one way approaches might be enough in the sense that the behavior of target systems doesn't necessarily need both features. As an alternative to the hybrid model, a *timed discrete event system* whose continuous feature is captured by clocks, has been widely used especially for man-made systems (Ho, 1993). Timed Petri Nets (Wang, 1998), timed automata (TA) (Alur and Dill, 1994), and Discrete Event System Specification (DEVS) (Zeigler *et al.*, 2000) are well-known methodologies in this category.

Unlike both timed Petri Nets and timed automata which are extended from their own un-timed ordinary formalisms, DEVS was born with a system theory in 1976. Zeigler designated DEVS to capture the system I/O over real-valued time (Zeigler, 1976). He also made DEVS's dynamics clear by comparing other fundamental dynamic systems such as the differential equation system and the discrete time system using the general system theory in his first book. In the 1980s the *hierarchical and modular modelling functionality* was added to DEVS (Zeigler, 1984) so that the user builds DEVS models in gradually (Sargent *et al.*, 1993) or rapidly (Hwang and Choi, 1999). In early 1990s, a hybrid model, called DEV& DESS, has been proposed as the combined form of DEVS and the differential equation system (Praehofer, 1991; Praehofer *et al.*, 1993). Since 2000, DEVS is showing a promising unified methodology for the hybrid model simulation (Kofman, 2004; Zeigler, 2006).

Until recently, the DEVS formalism was most often studied from the viewpoint of simulation, although there has been some work on verification and related issues (e.g., (Cutler, 1980), (Zeigler and Chi, 1992), (Hong and Kim, 1996), (Hong and Kim, 2005), and (Hocaoglu *et al.*, 2004)). Verification is desirable since

* Corresponding author. Email: mhhwang@ece.arizona.edu. This manuscript is accepted by IJGS.

it can examine, in principle, the complete behavior of a DEVS model while simulation is limited to a finite subset of it. One of the barriers we encounter when trying to describe when trying to describe the behavior of DEVS networks is the infinite-state space problem. To overcome this problem, proposed was a strongly restricted sub-class of DEVS called schedule-preserved DEVS (SP-DEVS) that is closed under the coupling operation called by the relative schedule abstraction (Hwang, 2005b). Relaxing the expressive restriction of SP-DEVS, finite-deterministic DEVS (FD-DEVS) was designated to allows the selective reschedule with external inputs (Hwang and Zeigler, 2006b). The behavior of FD-DEVS networks can be evaluated by a finite reachable graph using the difference bound matrix (Dill, 1989) so modular and hierarchical model checking was possible (Hwang and Zeigler, 2006a). Here, a fundamental question aries.

Q.1 *Does more relaxed but also verifiable DEVS exist?*

One possible answer might be relaxed as timed automata (TA)(Alur and Dill, 1994) that allows us to model nondeterministic transitions. The good properties of TA are its verification capabilities such as checking liveness and fairness (Alur and Dill, 1994), linear temporal logic (Henzinger, 1998), and branching temporal logic (Alur *et al.*, 1993); state minimization (Alur *et al.*, 1992) (Tripakis and Yovine, 2001); and symbolic representation (Bozga *et al.*, 1997; Behrmann *et al.*, 1999). However, from the viewpoint of the modeling aspect, TA seems to be improved.

To construct a complex system network, we use the *product construction* synchronizing transitions of different components if their associate events have an identical name (Alur and Dill, 1994; Alur, 1999). The event synchronization by name can be seen as a non-modular construction approach (Zeigler, 1990; Zeigler *et al.*, 2000), thus in many cases to avoid all undesired synchronization of sub-components, we should modify some events' names as well as associate transitions. This procedure can be a burden to the modeler. Once again, we may ask ourselves a couple of questions about TA.

Q.2 *Can we construct a hierarchical and modular TA network the same as we did in DEVS?*
Q.3 *If yes, can the hierarchical and modular TA still be verifiable?*

To answer above questions, this paper is organized as follows. In Section 2, we first define the general dynamic system and its dynamics in terms of the event trajectory as well as the state trajectory. In Section 3, we introduce an I/O clock system and its network as sub-classes of the general dynamic system so that they provide the foundation of the following two formalism: (1) FD-DEVS and its network in Section 4 and (2) I/O timed automaton and its network in Section 5. The expressiveness of the addressed sub-classes of the I/O clock system is addressed in Section 6. Conclusion and further research directions are given in Section 7.

2 General Dynamic System (GDS)

This section defines a general system. The dynamics of GDS will be examined from two different viewpoints and that will represent the dynamics instead of all its sub-classes.

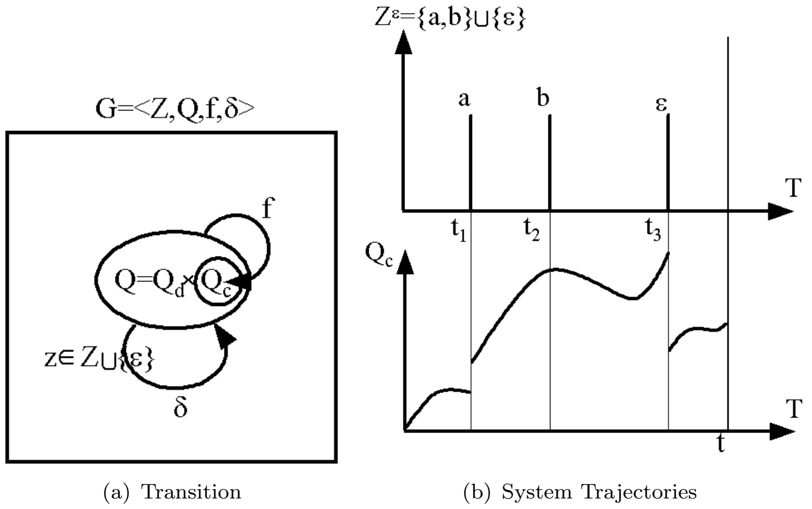
2.1 *Structure*

A general dynamic system $\mathcal{G}$ is a four-tuple

$$\mathcal{G} = < Z, Q, f, \delta >$$

where

- Z is a set of events.
- $Q = Q_d \times Q_c$ is a state set that consists of two disjoint sets: a discrete state set Q_d and a continuous set Q_c.
- $f : Q \rightarrow Q_c$ is the derivative function.



(a) Transition (b) System Trajectories

Figure 1. General Dynamic System

- $\delta : Q \times Z^\epsilon \to 2^Q$ is the *total* discrete transition function where $Z^\epsilon = Z \cup \{\epsilon\}$ where $\epsilon \notin Z$ is the *silent event.* Notice that δ is a total function so that it should define all possible $q \in Q$ and $z \in Z^\epsilon$.

□

As shown in Figure 1(a), the GDS changes its state either continuously or discretely. The dynamics of state transitions with time passage will be represented in the next section.

2.2 *Dynamics*

The time base of the GDS is $\mathbb{R}_{[0,\infty]}$ that denotes the set of non-negative reals with infinity. The dynamics of GDS can be seen from two different view points: one is the event trajectory (shown in the top of Figure 1(b)), the other is the state trajectory (shown in the bottom of Figure 1(b)).

2.2.1 ***Event Trajectory.*** From the time-passage view, an *event trajectory* over an event Z^ϵ can be seen as the function $\omega : \mathbb{R}_{[0,\infty]} \to Z^{\epsilon *}$ indicating that the event string $\bar{z} \in Z^{\epsilon *}$ occurring at $t \in \mathbb{R}_{[0,\infty]}$ where $Z^{\epsilon *}$ is the *Kleene closure* of $(Z \cup \{\epsilon\})$ that is the set of all finite length of strings over $(Z \cup \{\epsilon\})$ (Hopcroft *et al.*, 2000). In addition, we can see the event trajectory as a sequence of $(\bar{z}, t)$ pairs where $\bar{z} \in Z^{\epsilon *}$ and $t \in \mathbb{R}_{[0,\infty]}$.

Let's define ϕ as the *nonevent trajectory* that is distinguished from the silent event ϵ. Therefore, $\Omega_Z = Z^{\epsilon *} \times \mathbb{R}_{[0,\infty]}$ is the *all possible events over* $Z^{\epsilon *}$ and $\mathbb{R}_{[0,\infty]}$. Not only $[0,\infty]$ but also a time interval $[t_l, t_u] \subseteq [0,\infty]$ can be used for the *observation interval* of the event trajectory.[1] such that $\Omega_{Z[t_l,t_u]} = Z^{\epsilon *} \times [t_l, t_u]$. A special notation, $\phi_{[t_l,t_u]}$ denotes an event trajectory ω such that $\omega(t) = \phi$ for $t \in [t_l, t_u]$.

For example, an event trajectory shown in Figure 1(b) $\omega_{[0,t]} = \phi_{[0,t_1]}(a,t_1)\phi_{[t_1,t_2]}(b,t_2)\phi_{[t_2,t_3]}(\epsilon,t_3)\phi_{[t_3,t]}$ over $Z^\epsilon = \{a,b\} \cup \{\epsilon\}$ is $\omega(t) = a$ for $t = t_1$; $\omega(t) = b$ for $t = t_2$; $\omega(t) = \epsilon$ for $t = t_3$; $\omega(t) = \phi$ otherwise.

2.2.2 ***State Trajectory.*** Given a GDS $\mathcal{G} =< Z, Q, f, \delta >$, we would observe the state trajectory up to $t \in \mathbb{R}_{[0,\infty]}$.

To define a state trajectory, we first define the *continuous evolving function* $CONT : Q \times \mathbb{R}_{[0,\infty]} \to Q$. For $q = (q_d, q_c)$ where $q_d \in Q_d, q_c \in Q_c$, since discrete state q_d does not change continuous, we can say $d(q_d, q_c)/dt = (0, f(q_d, q_c))$. Thus, the continuously evolving state of q until $t \in \mathbb{R}_{[0,\infty]}$ is

[1]For simplicity, we focus on the closed boundary interval only.

$$CONT(q,t) = (q_d, q_c + \int_0^t f(q_d, q_c)) \tag{1}$$

It is fact that the empty segment with zero time interval $\phi_{[0,0]}$ cannot change anything, that is $CONT((q_d, q_c), 0) = (q_d, q_c + \int_0^0 f(q_d, q_c)) = (q_d, q_c)$. Recall that a silent event ϵ can change the system state such that $\delta(q, \epsilon) = q'$ and $q \neq q'$ as shown in the bottom of Figure 1(b) at (ϵ, t_3). But it doesn't mean that $\delta(q, z) \neq q$ for $z \in Z \cup \{\epsilon\}$ so it is also possible $\delta(q, z) = q$ as shown in the bottom of Figure 1(b) at (b, t_2).

Now we define *state trajectory* $\Delta : Q \times \Omega_{Z[0,t]} \to 2^Q$ for $\omega \in \Omega_{Z[0,t]}$ and $q = (q_d, q_c) \in Q$.

If $\omega = \phi_{[0,t]}$, then

$$\Delta(q, \omega) = \{CONT(q,t)\} = \{(q_d, q_c + \int_0^t f(q_d, q_c))\}. \tag{2}$$

Equation (2) explains that only the continuous state change is possible with an empty string $\phi_{[0,t]}$.

If we consider an event trajectory that is an empty string [0,t] concatenated with an discrete event or nonevent such as $\omega = \phi_{[0,t]}(z, t)$ where $z \in (Z^\epsilon \cup \{\phi\})$, then the set of state trajectories is defined as the following equation.

$$\Delta(q, \omega) = \delta(\Delta(q, \phi_{[0,t]}), z) = \delta((q_d, q_c + \int_0^t f(q_d, q_c)), z). \tag{3}$$

That is, $\Delta(q, w)$ is the set of states which can be reached by the continuous state change first and then the discrete state transition. Based on Equations (2) and (3), we can extend the set of state trajectories with a general event trajectory as follows.

$$\Delta(q, \omega) = \bigcup_{q' \in \Delta(q, \omega')} \delta(q', z) \text{ for } \omega' \in \Omega_{[0,t]} \text{ and } \omega = \omega'(z, t). \tag{4}$$

In other words, given an event trajectory $\omega = \omega'(z,t)$, all its possible state trajectories are the union of $\delta(q', z)$ where q' is a reachable state from q with ω'.

3 I/O Clock System and Its Hierarchical Network Structure

The continuous dynamics of the I/O clock system is captured by clocks whose speeds are identical to each other's. The definition of I/O clock system and its hierarchical network structure will provide the foundation of its sub-classes which will be introduced in Section 4 and in Section 5.

3.1 *Clock*

For capturing a value of time in $\mathbb{R}_{[0,\infty]}$, we use a clock c whose value is denoted by $v(c)$. Suppose that c is a clock and $t \in \mathbb{R}_{[0,\infty]}$ is a time value. Then comparing c and t is possible with the following operations: "equal to (=)", "less than or equal to ($\leq$)", "less than (<)", "greater than or equal to ($\geq$)" and "greater than (>)". For $\sim \in \{=, \leq, <, >, \geq\}$, $c \sim t$ iff $v(c) \sim t$. In addition, for assigning a clock c by a specific value t, we use $c := t$ in the context of $v(c) := t$.

We can extend this concept to the n-dimensional space of $\mathbb{R}_{[0,\infty]}$, denoted by $\mathbb{R}^n_{[0,\infty]}$. Given a set of clocks $C = \{c_1, \ldots, c_n\}$, the vector of C is defined as $V(C) = \underset{i=1\ldots n}{\times} c_i$. Given a $\mathbf{c} \in V(C)$ where $|C| = n$ and $\mathbf{t} \in \mathbb{R}^n_{[0,\infty]}$, we use the following notations: $\mathbf{c} := \mathbf{t}$ denotes $c_i := t_i$ for $i = 1$ to n (assigning $\mathbf{c}$ by $\mathbf{t}$); $\mathbf{c} + t$

does $\mathbf{c} := \mathbf{c} + t \cdot \mathbf{1}$ (time passage of $\mathbf{c}$ by t) where $\mathbf{1} \in \mathbb{R}^n_{[0,\infty]}$ denotes $\mathbf{t} \in \mathbb{R}^n_{[0,\infty]}$ such that $t_i = 1$ for $i = 1$ to n; $\mathbf{c}[C_R := 0]$ makes that $c := 0$ if $c_i \in C_R$ (reset clocks in $C_R \subseteq C$ to zero).

3.2 *I/O Clock System (IOCS)*

$$\mathcal{C} =< X, Y, Q, \eta, \delta >$$

where

- X and Y are disjoint event sets of input and output, respectively.
- $Q = Q_d \times Q_c$ where Q_d is a discrete state set and Q_c is a nonempty *clock* set.
- $\eta : Q \times Z^\epsilon \to \{0, 1\}$ where $Z^\epsilon = X \cup Y^\epsilon$ is the enabled indicating function that returns 1 when there is a possible transition from $q \in Q$ and $z \in Z^\epsilon$; otherwise, it returns 0.
- $\delta : Q \times Z^\epsilon \to 2^Q$ is the *partial* transition function in which some domain values can be undefined. However, it is assumed that for $q \in Q$ and $z \in Z^\epsilon, \eta(q, z) = 1 \Rightarrow \delta(q, z)$ is defined. Since $Z = X \cup Y^\epsilon$ and $X \cap Y = \varnothing$, the transition with $z \in X$ is an *external transition*, while that with $z \in Y^\epsilon$ is an *internal transition.*

$\square$

The IOCS is a special case of the GDS such that given an IOCS $\mathcal{C} =< X, Y, Q, \eta, \delta >$, its behavior can be described by a GDS $\mathcal{G} =< Z, Q_G, f, \delta_G >$ where

$$Q_G = (Q_d \cup \{\mathsf{error}\}) \times Q_c \tag{5}$$

such that $\mathsf{error} \notin Q_d$. Let $Z = X \cup Y$, $q = (s, \mathbf{c}) \in Q_G$, and $z \in Z^\epsilon$. Then derivative function $f : Q_G \to Q_c$ is defined as

$$f(s, \mathbf{c}) = \mathbf{1} \tag{6}$$

where $|\mathbf{1}| = |Q_c|$ because every clock's speed is equal to 1. Therefore, the continuous evolving function that was defined in Equation (1) can be rewritten for IOCS as

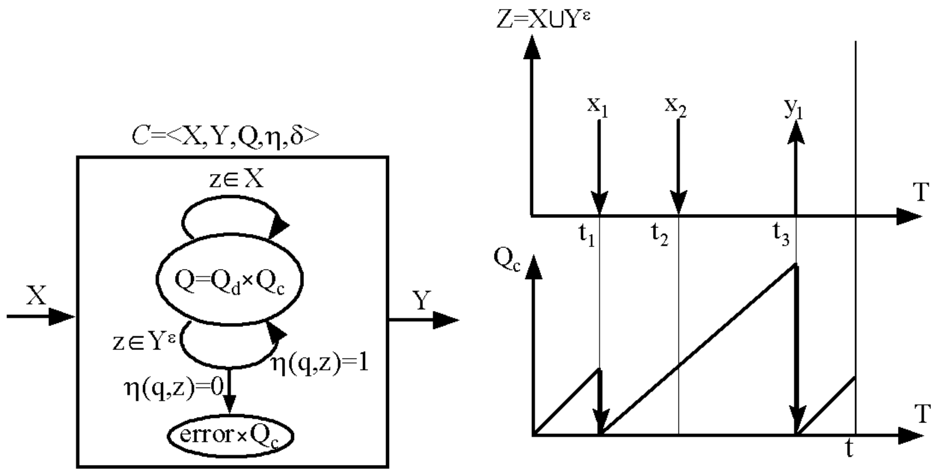
$$CONT(q, t) = (q_d, \mathbf{c} + \int_0^t \mathbf{1}) = (q_d, \mathbf{c} + t). \tag{7}$$

In other words, the interpretation of $CONT(q, t)$ in IOCS is that the *time passage* of t at q. The total discrete transition function $\delta_Q : Q \times Z^\epsilon \to 2^Q$ is defined for $q \in Q$ and $z \in Z^\epsilon$ as

$$\delta_G(q, z) = \begin{cases} \delta(q, z) & \text{for } z \in X, \eta(q, z) = 1 \\ \{q\} & \text{for } z \in X, \eta(q, z) = 0 \\ \delta(q, z) & \text{for } z \in Y^\epsilon, \eta(q, z) = 1 \\ \{(\mathsf{error}, \mathbf{c})\} & \text{for } z \in Y^\epsilon, \eta(q, z) = 0 \end{cases} \tag{8}$$

Equation (8) shows that the different interpretation of enable conditions between the external event and the internal event.

Since an external input $z \in X$ comes from outside, the IOCS allows z at any time. The interpretation is that if z inputs when the system is not enabled, no change occurs and it is not an error. However, an internal event $z \in Y^\epsilon$ is different. Since the internal transition of a system is generated from inside not outside, a regal internal transition is allowed only when it is enabled. Otherwise, this system routes into the error situation, denoted by error.



(a) Transition (b) System Trajectories

Figure 2. I/O Clock System

Figure 2(a) shows the conceptual diagram of this different interpretation in which we can see that IOCS doesn't allow any internal transition except one: it goes into error when it is not enabled ($\eta(q, z) = 0$).

Figure 2(b) conceptually illustrates an example of both an event trajectory (top) in which the event should be one of either an external input or an internal output, and a state trajectory of a clock (bottom) whose derivative is always one.

3.3 *Hierarchical Network Structure (HNS)*

A *hierarchical network structure*, or a *network* is a 7-tuple,

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

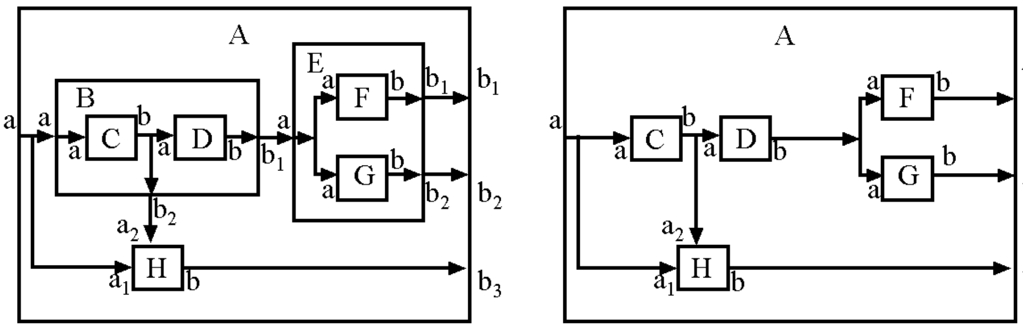
where

- X and Y are disjoint event sets of input and output, respectively.
- D is a non-empty and finite set of sub-components' names.
- $\{M_i\}$ is the non-empty and finite index set for each $i \in D$. The element M_i is either an HNS model or an IOCS model.
- $C_{xx} \subseteq X \times \bigcup_{i \in D} X_i$ is the external input coupling relation where X_i is the input event set of a sub-component whose name is i.
- $C_{yx} \subseteq \bigcup_{i \in D} Y_i \times \bigcup_{i \in D} X_i$ is the internal coupling relation.
- $C_{yy} \subseteq \bigcup_{i \in D} Y_i \times Y$ is the external output coupling relation. □

Notice that the HNS has a hierarchical structure because a sub-component can be an HNS itself. Figure 3(a) shows an HNS in which models A, B and E are HNS models. However, every leaf node in an HNS model is an IOCS that is introduced in Section 2. Given an HNS N, the *set of leaf components*, denoted by LD is a set of all leaves in the hierarchy of N. Consider an HNS A shown in Figure 3(a), then its $LD = \{\mathsf{C}, \mathsf{D}, \mathsf{F}, \mathsf{G}, \mathsf{H}\}$. Generally, LD of N can be achieved with the following function.

$$\mathsf{GetLD}(N, LD) = \begin{cases} \text{for each } i \in D & \\ \quad \text{if } (M_i \text{ is an HNS}) & \text{then } \mathsf{GetLD}(M_i, LD); \\ \quad \text{else} & LD := LD \cup \{i\}; \end{cases} \tag{9}$$

We can also define the set of coupling paths with LD. A *set of input coupling path* P_{xx} is



(a) Hierarchical Network (b) Flattened Network

Figure 3. Hierarchical Network Structures

$$P_{xx} = \{(x, x_i)|x \in X \text{ has a coupling path to } x_i \in X_{LD}\} \tag{10}$$

where $X_{LD} = \bigcup_{i \in LD} X_i$ is a set of input events of models in LD of N. A *set of internal coupling path P_{yx}* is

$$P_{yx} = \{(y_i, x_i)|y_i \in Y_{LD} \text{ has a coupling path to } x_i \in X_{LD}\} \tag{11}$$

where $Y_{LD} = \bigcup_{i \in LD} Y_i$ is a set of output events of models in LD of N. Similarly, a *set of output coupling path P_{yy}* is defined as

$$P_{yy} = \{(y_i, x_i)|y_i \in Y_{LD} \text{ has a coupling path to } x_i \in X\} \tag{12}$$

For example, in Figure 3(a), $P_{xx} = \{(\texttt{A.a}, \texttt{C.a}), (\texttt{A.a}, \texttt{H.a}_1)\}$ where $\texttt{A.a}$ indicates an input event $\texttt{a}$ of a model $\texttt{A}$, while $P_{yx} = \{(\texttt{C.b}, \texttt{D.a}), (\texttt{C.b}, \texttt{H.a}_2), (\texttt{D.b}, \texttt{F.a}), (\texttt{D.b}, \texttt{G.a})\}$.

Given an HNS N, the flattened network of N, denoted by N^F is defined as $N^F =< X, Y, LD, P_{xx}, P_{yx}, P_{yy} >$. For example, Figure 3(b) shows the flattened network N^F of the hierarchical network N shown in Figure 3(a). Notice that we can always get the N^F from an HNS N.

Practically, we can see an event as a pair of (*port*, *value*) and the coupling as a pair of ($port_{source}$, $port_{destination}$) (Zeigler, 1990; Zeigler *et al.*, 2000). The basic assumption of the *port coupling* is that the value of $port_{source}$ is casted to that of $port_{destination}$. We can find that the realistic example of the port coupling in the VHDL language (Skahill, 1996) and the language of programmable logic controller (PLC) (Lewis, 1998).

3.3.1 *Behavior of HNS.* The behavior of a given HNS $N =< X, Y, D, \{M_i\}, C_{xx}, C_{yx}, C_{yy} >$ can be described as an IOCS $\mathcal{C} =< X, Y, Q, \eta, \delta >$ such that $Q = \{(\ldots, q_i, \ldots)|q_i \in Q_{Gi}, i \in LD\}$; Let $Z^\epsilon = X \cup (\bigcup_{i \in LD} Y_i^\epsilon)$,$(\ldots, q_i, \ldots) \in Q$, and $z \in Z^\epsilon$. Then $\eta : Q \times Z^\epsilon \rightarrow \{0, 1\}$ is defined as

$$\eta((\ldots, q_i, \ldots), z) = \begin{cases} 1 & \text{for } z \in X, \exists (z, x_i) \in P_{xx}, \eta_i(q_i, x_i) = 1 \\ 1 & \text{for } z \in \bigcup_{i \in LD} Y_i^\epsilon, \exists i \in LD, \eta_i(q_i, z) = 1 \\ 0 & \text{otherwise} \end{cases} \tag{13}$$

$\delta : Q \times Z^\epsilon \rightarrow 2^Q$ is defined as

$$\delta((\ldots, q_i, \ldots), z) = \{(\ldots, q_i', \ldots) \in Q\} \tag{14}$$

such that

(i) for $z \in X$,

$$q'_i \in \delta_i(q_i, x_i) \text{ for } (z, x_i) \in P_{xx} \tag{14a}$$

(ii) for $z \in \bigcup_{i \in LD} Y_i^{\epsilon}$,

$$q'_i \in \begin{cases} \delta_i(q_i, z) & \text{for } z \in Y_i^{\epsilon} \\ \delta_i(q_i, x_i) & \text{for } (z, x_i) \in P_{yx} \end{cases} \tag{14b}$$

By this definition of HNS's behavior, we can make the following corollary.

Corollary 3.1 *IOCS is closed under coupling.*

Based on IOCS and its HNS addressed in this chapter, we will introduce two sub-classes in Section 4 and 5, respectively, and these are finally compared to each other in terms of expressiveness in Section 6.

4 Finite and Deterministic DEVS (FD-DEVS)

As a sub-class of IOCS and a verifiable class of DEVS, FD-DEVS is defined in this section.

4.1 *FD-DEVS*

An *FD-DEVS* is an 8-tuple,

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

where,

- X and Y are disjoint finite event sets of input and output, respectively.
- S is a non-empty and finite discrete state set.
- $\tau : S \rightarrow \mathbb{Q}_{[0,\infty]}$ is the scheduling function where $\mathbb{Q}_{[0,\infty]}$ denotes the set of non-negative rational numbers with infinity.
- $\delta_x : S \times X \rightarrow S$ is the external transition function.
- $\rho : S \times X \rightarrow \{0, 1\}$ is the reschedule-indicating function that returns 1 when scheduling is needed; otherwise, returns 0.
- $\delta_\tau : S \rightarrow S$ is the internal transition function.
- $\lambda : S \rightarrow Y^{\epsilon}$ is the internal output function. □

We assume that $\delta_x, \rho, \delta_\tau$ and λ are partial functions, but τ is a total function.

4.1.1 ***Behavior of FD-DEVS.***

The behavior of an atomic FD-DEVS $M =< X, Y, S, \tau, \delta_x, \rho, \delta_\tau, \lambda >$ can be described as an IOCS $\mathcal{C} =< X, Y, Q, \eta, \delta >$ such that $Q = \{(s, t_s, c) | s \in S, t_s \in \bigcup_{s \in S} \{\tau(s)\}, 0 \leq c \leq t_s\} \cup \{(\mathsf{error}, \infty, c) | \mathsf{error} \notin S, 0 \leq c < \infty\}$ where c is a clock tracking the elapsed time since the last t_s update; Notice that t_s is not a clock but a discrete state whose number of possible values is bounded by $|S|$. [1]

[1] In view point of $Q = Q_d \times Q_c$ in Section 2, we can say that $Q_d = \{(s, t_s) | s \in S, t_s \in \bigcup_{s \in S} \{\tau(s)\}\} \cup \{(\mathsf{error}, \infty)\}$ and $Q_c = \{c\}$ s.t. $q = (s, t_s, c) \Rightarrow 0 \leq c \leq t_s$.

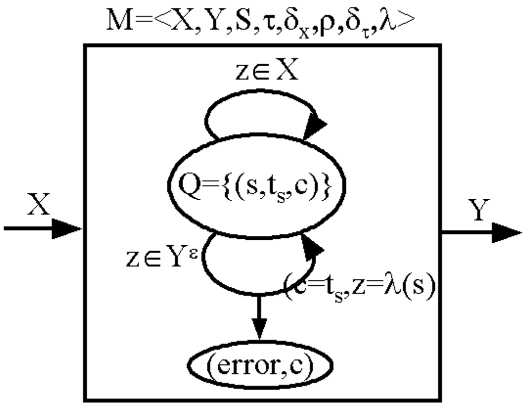


Figure 4. FD-DEVS Transition

Let $Z^\epsilon = X \cup Y^\epsilon$, $z \in Z^\epsilon$ and $(s, t_s, c) \in Q$. Then the enabled indicating function $\eta : Q \times Z^\epsilon \to \{0, 1\}$ is defined as Equation (15).

$$\eta((s,t_s,c),z) = \begin{cases} 1 & \text{for } z \in X, \rho(s,z) \perp, \delta_x(s,z) \perp \\ 1 & \text{for } z \in Y^\epsilon, c = t_s < \infty, z = \lambda(s), \delta_\tau(s) \perp \\ 0 & \text{otherwise} \end{cases} \tag{15}$$

where $\rho(s, z) \perp$ denotes that $\rho(s, z)$ is defined. From now on, given a partial function $f : A \to B$ and a certain domain value $a \in A$, we use the notation $f(a) \perp$ that denotes "$f(a)$ is defined".

From the $\eta((s, t_s, c), z)$ for $z \in Y^\epsilon$, we see that the internal transition can perform when the elapsed time reachs to the schedule time ($c = t_s$) and at this moment the output $z = \lambda(s)$ is transmitted outside. Otherwise, it routes into the error state (see Figure 4). At that time, the next state is determined by the following transition function $\delta : Q \times Z^\epsilon \to 2^Q$ that is

$$\delta((s,t_s,c),z) = \begin{cases} \{(\delta_x(s,z), \tau(\delta_x(s,z)), 0)\} & \text{for } z \in X, \rho(s,z) = 1 \\ \{(\delta_x(s,z), t_s, c)\} & \text{for } z \in X, \rho(s,z) = 0 \\ \{(\delta_\tau(s), \tau(\delta_\tau(s)), 0)\} & \text{for } z \in Y^\epsilon \end{cases} \tag{16}$$

Example 4.1 (FD-DEVS Elevator) Let's consider a one-person capacity elevator that covers 1^{st}, 2^{nd} and 3^{rd} floors as shown in Figure 5. At each floor, we select a different level as our destination. To make the example simple, we assume that the elevator can get starting to serve only after it becomes idle. In other words, the request is ignored while the elevator is in motion.

Figure 6 shows an elevator FD-DEVS model. There are three states `Ii` where $\texttt{i} \in \{1, 2, 3\}$ standing for idle at `i`-th floor. The rest of states in the form (`Mi,j,k`) where $\texttt{i,j,k} \in \{1, 2, 3\}, \texttt{j} \neq \texttt{k}$ denotes a moving state starting from `i`-th floor, serving a request from the `j`-th floor to the `k`-th floor. The life time of state $s \in S$, $\tau(s)$ is denoted at the bottom of a node such that $\tau(s) = \infty$ for $s \in \{\texttt{I1}, \texttt{I2}, \texttt{I3}\}$ while for $s =$ (`Mi,j,k`), τ((`Mi,j,k`))="time to move to requesting floor"+"time to move to destination floor"$=|\texttt{i} - \texttt{j}| + |\texttt{j} - \texttt{k}|$ unit times of moving one floor.

Recall that the external request during moving can be ignored in the FD-DEVS when an undefined external transition such as $\delta_x(s, x)$ for $s =$ (`Mi,j,k`), $x \in X$ doesn't change anything, □

4.2 *FD-DEVS Network*

An FD-DEVS network is an IOCS network such that $N =< X, Y, D, \{M_i\}, C_{xx}, C_{yx}, C_{yy} >$ whose leaf sub-components are FD-DEVS models. Since an FD-DEVS network is a special case of the HNS, the behavior of

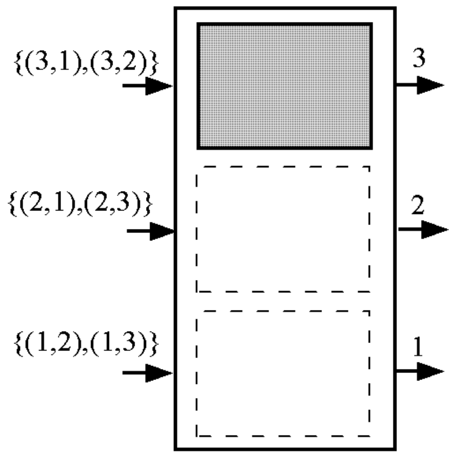


Figure 5. Elevator

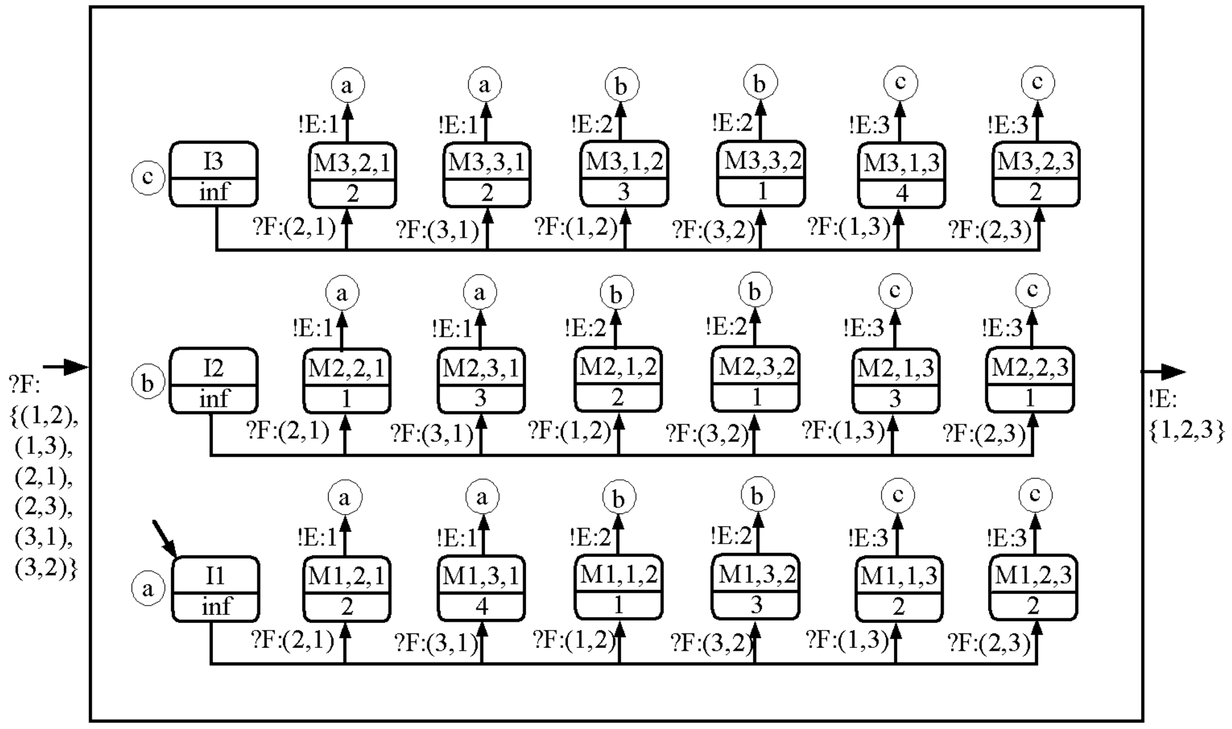
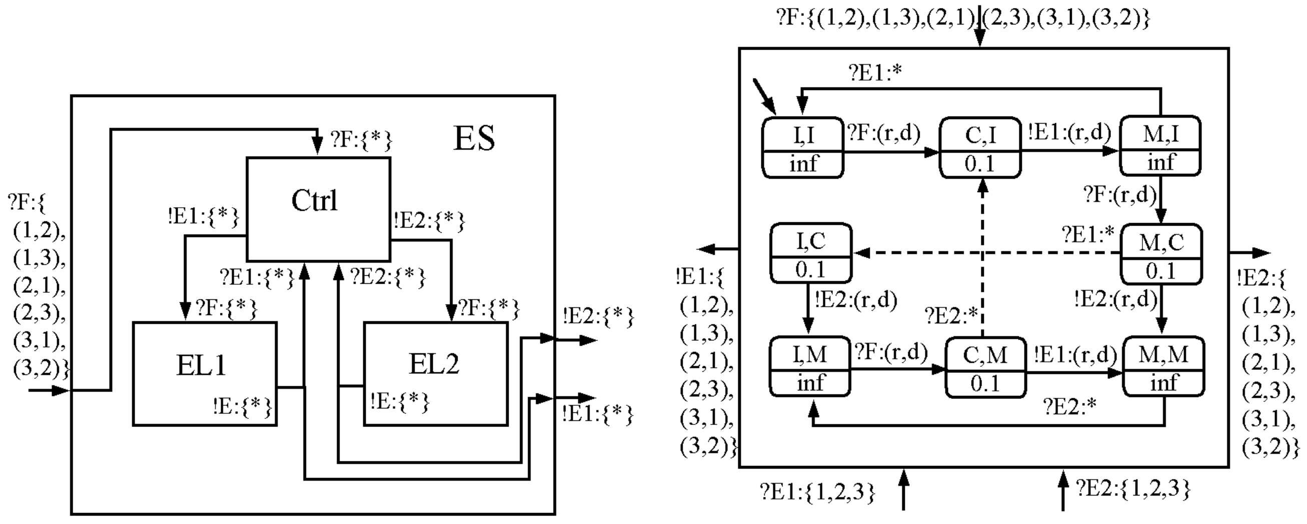


Figure 6. Elevator FD-DEVS Model

an FD-DEVS network is straightforwardly defined by substituting Q_i, η_i, and δ_i in the behavior definition of the HNS (see Section 3.3.1) with Q, η and δ in the behavior definition of FD-DEVS (Section 4.1.1).

Example 4.2 (FD-DEVS Network for Two-Elevator System) Let's consider an elevator system shown in Figure 7(a) that consists of two elevators (`EL1`) and (`EL2`) and one controller (`Ctrl`). For constructing this system using an FD-DEVS network, we use the FD-DEVS elevator introduced in Example 4.1 for `EL1` and `EL2` and one controller `Ctrl`. As we mentioned in Section 3.3, we use ports and their couplings in this example. The elevator system `ES` has an internal port `?F` whose possible values are {(1,2), (1,3), (2,1), (2,3), (3,1), (3,2)}, while its output ports are `!E1` and `!E2`. Since we assume the value of source port is transmitted to that of destination port which is coupled, we just draw the port couplings in Figure 7(a) rather than the event couplings. The port couples in `ES` are $C_{xx} = \{(\texttt{ES.?F}, \texttt{Ctrl.?F})\}$, $C_{yx} = \{(\texttt{Ctrl.!E1}, \texttt{EL1.?F}), (\texttt{Ctrl.!E2}, \texttt{EL2.?F}), (\texttt{EL1.!E1}, \texttt{Ctrl.?E1}), (\texttt{EL1.!E2}, \texttt{Ctr2.?E1})\}$ and $C_{yy} = \{(\texttt{EL1.!E1}, \texttt{ES.?E1}), (\texttt{EL1.!E2}, \texttt{ES.?E1})\}$

The controller `Ctrl` is modelled as shown in Figure 7(b). `Ctrl`'s initial state is (`I,I`) indicating both



(a) Network Model (b) Controller (Ctrl) FD-DEVS

Figure 7. Two-Elevator System using FD-DEVS

elevators are idle. Once it receives a service request `?F:(r,d)` it transmits a service command to `EL1` by generating an output event `!E1:(r,d)` at state (`C,I`). A state (`C,I`) means that `Ctrl` is generating a command for `EL1` during `EL2` `EL1` is idle, while a state (`C,M`) is generating a command for while `EL2` is moving. We can interpret (`M,C`) and (`I,C`) in the same context. All commanding states need the generating time 0.1 unit and the rest states have no time limit to stay, denoted by “inf” in Figure 7(b) unless `?F:(r,d)` arrives.

A dashed line transition in Figure 7(b) such as from (`M,C`) to (`I,C`) by `?E1:*` indicates the external transitions without schedule update. Formally, $\rho((\texttt{M,C}), \texttt{?E1:*}) = 0$ and $\delta_x((\texttt{M,C}), \texttt{?E1:*}) = (\texttt{I,C})$. These capture the situation occurring while `Ctrl` is generating a service command to `EL2` within 0.1 unit time, `EL1` finishes its service and returns to the idle state. At this moment, `Ctrl` just marks `EL1`'s state as `M` and then keeps generating the command to `EL2` again.

Notice that the controller `Ctrl` gives a higher service priority to `EL1` when both `EL1` and `EL2` are idle. □

5 I/O Timed Automaton (IOTA)

The ordinary timed automata (TA) of (Alur, 1999) is modified by splitting the event set into input and output so that the coupling operation of TA can be possible.

5.1 *I/O Timed Automaton*

An I/O timed automaton is a 6-tuple

$$A =< X, Y, L, C, I, T >$$

where

- X and Y are disjoint and finite event sets of input and output, respectively.
- L is a nonempty and finite set of locations.
- C is a nonempty and finite set of clocks.
- $I : L \to \Phi(C)$ is a mapping that labels each location l to a clock constraint φ in $\Phi(C)$. A clock constraint φ defined by the grammar

$$\varphi := c \leq u | c < u | u \leq c | u < c | \varphi_1 \wedge \varphi_2$$

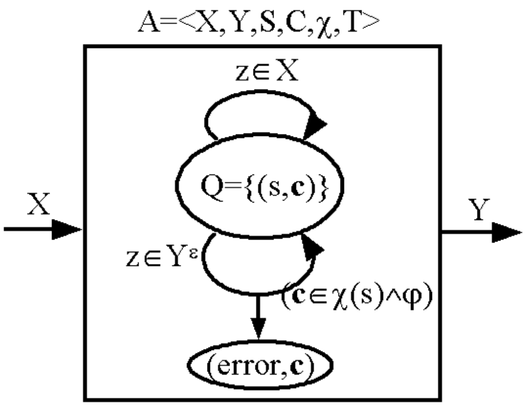


Figure 8. Transition of Timed I/O Automaton

where c is a clock in C and u is a constant in $\mathbb{Q}_{[0,\infty]}$ that is the set of non-negative rational numbers with infinity. A special notation $\varphi =\_$ indicates non-constraint that is $\varphi = 0 \leq c \leq \infty$.

- $T \subseteq L \times Z^{\epsilon} \times \Phi(C) \times 2^{C} \times L$ is a set of transitions where $Z^{\epsilon} = X \cup Y^{\epsilon}$. A transition $(l, z, \varphi, C_R, l') \in T$ represents an edge from location l to another l' on event z. $I(l) \wedge \varphi$ is a clock constraint when the transition is enabled and the set $C_R \subseteq C$ gives the clocks to be reset with this transition.

□

5.1.1 ***Behavior of IOTA.*** Given an IOTA $A =< X, Y, L, C, I, T >$, its state transition is also described by an IOCS $\mathcal{C} =< X, Y, Q, \eta, \delta >$ such that $Q = Q_d \times Q_c$ where $Q_d = L \cup \{\mathsf{error}\}$ ($\mathsf{error} \notin L$) and $Q_c = C$; Let $l \in Q_d$,$\mathbf{c} \in V(C)$ and $z \in Z^{\epsilon}$. A *set of enabled transitions* of l, $\mathbf{c}$ and z is defined as

$$ET(l, \mathbf{c}, z) = \{(l, z, \varphi, C_R, l') \in T | \mathbf{c} \in I(l) \wedge \varphi\} \tag{17}$$

where $\mathbf{c} \in I(l) \wedge \varphi$ denotes that $\forall i, v(c_i)$ satisfies all constraints in $I(l) \wedge \varphi$. The enabled indicating function $\eta : Q \times Z^{\epsilon} \rightarrow \{0, 1\}$ is defined based on $ET(l, \mathbf{c}, z)$ such that

$$\eta((l, \mathbf{c}), z) = \begin{cases} 1 & \text{for } ET(l, \mathbf{c}, z) \neq \varnothing \\ 0 & \text{otherwise} \end{cases} \tag{18}$$

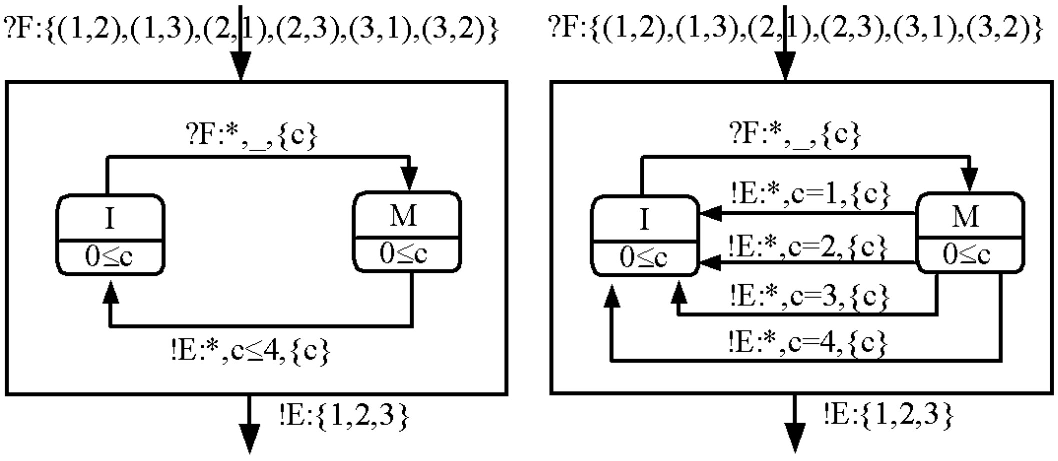
The transition function $\delta : Q \times Z^{\epsilon} \rightarrow 2^{Q}$ is

$$\delta((l, \mathbf{c}), z) = \{(l', \mathbf{c}[C_R := 0]) | (l, z, \varphi, C_R, l') \in ET(l, \mathbf{c}, z)\} \tag{19}$$

In other words, when (l, z, φ, C_R, l') executes, the value of each clock in C_R is reset to 0, as well as the discrete state changes from l to l'. Recall IOTA is a special class of IOCS, thus the legal internal transition of (l, z, φ, C_R, l') is allowed if $ET(l, c, z) \neq \varnothing$ and it means ($\mathbf{c} \in I(l) \wedge \varphi$) (refer to Figure 8).

Example 5.1 (IOTA Elevator) Let's construct an IOTA model for the elevator introduced in Example 4.1. Unlike the FD-DEVS model of Example 4.1, we emphasize the nondeterministic feature provided by IOTA, so we use only one idle state `I` instead of `I1`, `I2`, and `I3`, use only one moving state `M` instead of 18 different moving states.

In Figure 9(a) and (b), a node contains the state $l \in \{\mathtt{I}, \mathtt{M}\}$ at top and its invariant clock constraints $I(l)$ at bottom, for example, $0 \leq c$ for $l = \mathtt{I}$ and $1 \leq c$ for $l = \mathtt{M}$. For each edge $(l, z, \varphi, C_R, l') \in T$, one arc from node l to node l' with z, φ, C_R is drawn in Figures 9(a) and 9(b). For example, $(\mathtt{I}, \mathtt{?F{:}*}, \_, \{c\}, \mathtt{M})$ denotes one edge (1) whose triggering event has a port `?F` with its value (`r,d`) as one of {(1,2), (1,3), (2,1), (2,3), (3,1), (3,2) }; (2) $\varphi = \_$, that is no clock constraint; (3) the set of reset clocks is $C_R = \{c\}$.



(a) $\delta((\mathrm{M},c),!\mathrm{E}:*)$ at $1 \le c \le 4$ (b) $\delta((\mathrm{M},c),!\mathrm{E}:*)$ at $c \in \{1,2,3,4\}$

Figure 9. Elevator IOTA models

Observe that the transition from `M` to `I` is enabled when $c \in [1,4]$ in Figure 9(a), while that transition is enabled when either $c = 1$, $c = 2$, $c = 3$ or $c = 4$ in Figure 9(b). □

5.2 *IOTA Network*

An IOTA network is an IOCS network such that $N =< X, Y, D, \{M_i\}, C_{xx}, C_{yx}, C_{yy} >$ whose leaf sub-components are IOTA models. Since an IOTA network is a special case of the HNS, the behavior of an IOTA network is defined by substituting Q_i, η_i, and δ_i in the behavior definition of the HNS (see Section 3.3.1) with Q, η and δ in the behavior definition of IOTA (Section 5.1.1). Let us omit it here, but we will revisit this when proving Theorem 6.6 in Section 6 later.

Example 5.2 (IOTA Network for Two-Elevator System) Let's reconsider the two-elevator system as shown in Figure 7(a) that consists of two elevators (`EL1`) and (`EL2`) and one controller (`Ctrl`). But now, the IOTA elevator introduced in Example 5.1 is used for two identical elevators `EL1` and `EL2`.

Similarly, we use an IOTA controller shown in Figure 10. We find that there is nondeterministic transition at $l =$ (`I,I`) with input event `?F:(r,d)` such that the resulting state can be either (`C,I`) or (`I,C`). □

6 Expressiveness of IOCS Sub-classes

In this section, we try to compare sub-classes of IOCS in terms of their expressiveness. In this paper *E(formalism)* stands for the expressiveness of *formalism*. By definitions of FD-DEVS and IOTA and any NHS through Sections 3, 4 and 5, we can make the following corollaries.

Corollary 6.1 *E(FD-DEVS)* $\subset$ *E(IOCS).*

Corollary 6.2 *E(IOTA)* $\subset$ *E(IOCS).*

Corollary 6.3 *E(formalism)* $\subseteq$ *E(formalism Network) where formalism* $\in$ *{FD-DEVS, IOTA}.*

Since IOCS is closed under the coupling operation that was proven in Corollary 3.1, we get the following corollary.

Corollary 6.4 *E(IOCS) = E(IOCS Network).*

Lemma 6.5 *FD-DEVS is not closed under the coupling operation. So E(FD-DEVS)* $\subset$ *E(FD-DEVS Network).*

Proof: Suppose that there is an FD-DEVS network $N =< X_N, Y_N, D, \{M_i\}, C_{xx}, C_{yx}, C_{yy} >$ such that $D = \{a, b\}$ and there exists $x \in X$ such that (1) $(x, x_a) \in C_{xx}, x_a \in X_a, \exists s_a \in S_a : \rho_a(s_a, x_a) =$

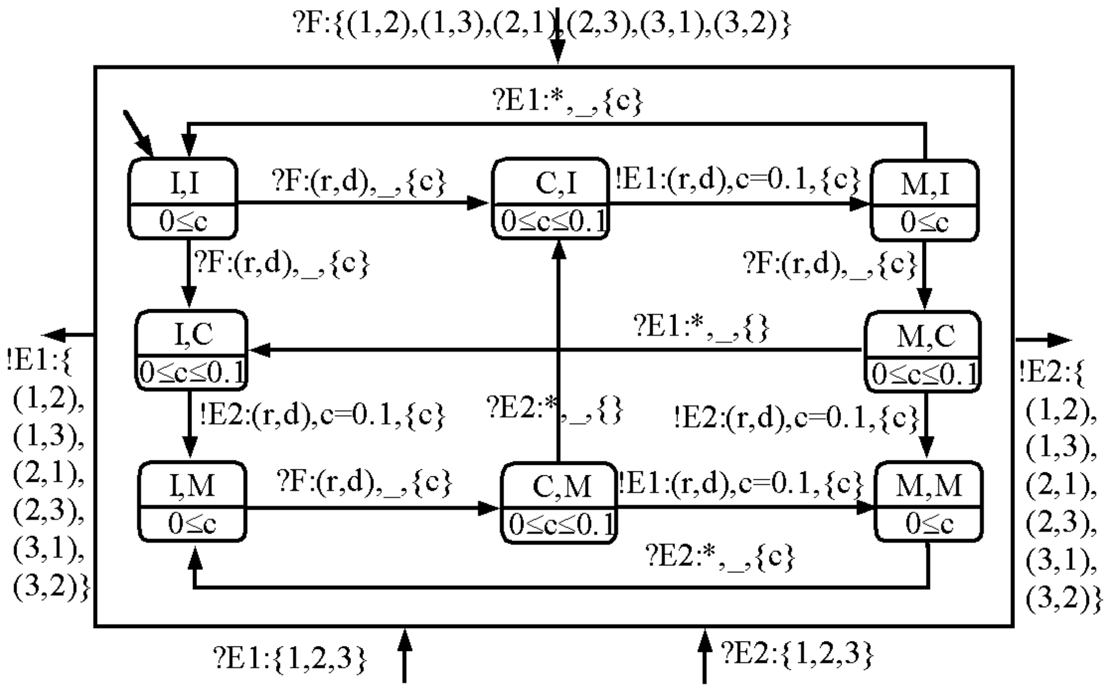


Figure 10. Controller IOTA model

$1, \tau_s(\delta_{xa}(s_a, x_a)) \neq \infty$ and (b) $(x, x_b) \in C_{xx}, x_b \in X_b, \exists s_b \in S_b : \rho_b(s_b, x_a) = 0, \tau_s(s_a) \neq \infty$. Since this is a partially rescheduled DEVS (Hwang, 2005a), there is no atomic FD-DEVS M whose behavior is the same as that of N. ■

THEOREM 6.6 *IOTA is closed under coupling. So E(IOTA) = E(IOTA Network)*

Proof Since an IOTA network $N =< X, Y, D, \{M_i\}, C_{xx}, C_{yx}, C_{yy} >$ is a special case of the IOCS network, the behavior is defined by substituting Q_i, η_i, and δ_i in the behavior definition of the IOCS network (see Section 3.3.1) with Q, η and δ in the behavior definition of IOTA (Section 5.1.1). As a result, the behavior of the IOTA network N is defined as an IOCS $\mathcal{C} =< X, Y, Q, \eta, \delta >$ where $Q = \underset{i\in LD}{\times} Q_i$. Let $Z^\epsilon = X \cup \bigcup_{i\in LD} Y_i^\epsilon$, $(\ldots, q_i, \ldots) \in Q$, $q_i = (l_i, \mathbf{c}_i)$ and $z \in Z^\epsilon$. Then $\eta : Q \times Z^\epsilon \to \{0, 1\}$ is defined as

$$\eta((\ldots, (l_i, \mathbf{c}_i), \ldots), z) = \begin{cases} 1 & \text{for } z \in X, \exists (z, x_i) \in P_{xx}, ET(l_i, \mathbf{c}_i, x_i) \neq \varnothing \\ 1 & \text{for } z \in \bigcup_{i\in LD} Y_i^\epsilon, \exists i \in LD, ET(l_i, \mathbf{c}_i, z) \neq \varnothing \\ 0 & \text{otherwise} \end{cases}$$

In addition, $\delta : Q \times Z^\epsilon \to 2^Q$ is defined as

$$\delta((\ldots, q_i, \ldots), z) = \{(\ldots, q_i', \ldots) \in Q\}$$

such that

(i) for $z \in X$,

$$(l_i', \mathbf{c}_i') \in \delta_i(l_i, \mathbf{c}_i, x_i) \text{ for } (z, x_i) \in P_{xx} \tag{20}$$

(ii) for $z \in \bigcup_{i\in LD} Y_i^\epsilon$,

$$(l'_i, \mathbf{c}'_i) \in \begin{cases} \delta_i(l_i, \mathbf{c}_i, z) & \text{for } z \in Y_i^\epsilon \\ \delta_i(l_i, \mathbf{c}_i, x_i) & \text{for } (z, x_i) \in P_{yx} \end{cases} \tag{21}$$

Recall that $ET(l_i, \mathbf{c}_i, z) = \{(l_i, z, \varphi_i, C_{Ri}, l'_i) \in T_i | \mathbf{c}_i \in I_i(l_i) \wedge \varphi_i\}$ (see Equation (17)) and $\delta_i(l_i, \mathbf{c}_i, z) = \{(l'_i, \mathbf{c}_i[C_{Ri} := 0]) | (l_i, z, \varphi_i, C_{Ri}, l'_i) \in ET(l_i, \mathbf{c}_i, z)\}$ (see Equation (19)).

Let's show that above IOCS $\mathcal{C}$ can be described by an IOTA $A =< X, Y, S, C, I, T >$ such that $S = \underset{i \in D}{\times} S_i$. $C = \bigcup_{i \in D} C_i$. Let $Z = X \cup \bigcup_{i \in LD} Y_i$ and suppose that $\mathbf{s} = (\dots, l_i, \dots) \in S, \mathbf{s}' = (\dots, l'_i, \dots) \in S$ and $z \in Z^\epsilon$. Then, $I : S \to \Phi(C)$ is defined as $I(\mathbf{s}) = \underset{i \in D}{\wedge} I_i(l_i)$. And the transition set $T \subseteq S \times Z^\epsilon \times \Phi(C) \times 2^C \times S$ is defined as

$$((\dots, l_i, \dots), z, \varphi, C_R, (\dots, l'_i, \dots)) \in T \tag{22}$$

(i) If $z \in X$: Let $T_{xx}(l_i, z) = \{(l_i, x_i, \varphi_i, C_{Ri}, l''_i) \in T_i | (z, x_i) \in P_{xx}\}$ be a *set of externally coupled transitions* at l_i by z and $D_{xx}(\mathbf{s}, z) = 2^{\{i | i \in D, T_{xx}(l_i, z) \neq \varnothing\}}$ be a *name set of externally coupled components* at $\mathbf{s}$ by z. Then the conditions that Equation (22) satisfies Equation (20) are:
$\varphi = \bigwedge_{i \in D_{xx}(\mathbf{s}, z)} \varphi_i$; $C_R = \bigcup_{i \in D_{xx}(\mathbf{s}, z)} C_{Ri}$;

$$l'_i = \begin{cases} l''_i & \text{for } i \in D_{xx}(\mathbf{s}, z) \\ l_i & \text{otherwise} \end{cases}$$

(ii) If $\exists i^* \in LD$ s.t. $z \in Y_{i^*}^\epsilon$ and $(l_{i^*}, z, \varphi_{i^*}, C_{Ri^*}, l''_{i^*}) \in T_{i^*}$: Let $T_{yx}(l_i, z) = \{(l_i, z, \varphi_i, C_{Ri}, l''_i) \in T_i | (z, x_i) \in P_{yx}\}$ be a *set of internally coupled transitions* at l_i by z and $D_{yx}(\mathbf{s}, z) = 2^{\{i | i \in D, T_{yx}(l_i, z) \neq \varnothing\}}$ be a *name set of internally coupled components* at $\mathbf{s}$ by z. Then the conditions that Equation (22) satisfies Equation (21) are:

$$\varphi = \varphi_{i^*} \wedge \bigwedge_{i \in D_{yx}(\mathbf{s}, z)} \varphi_i; \; C_R = C_{Ri^*} \cup \bigcup_{i \in D_{yx}(\mathbf{s}, z)} C_{Ri};$$

$$l'_i = \begin{cases} s''_{i^*} & \text{for } i = i^* \\ l''_i & \text{for } i \in D_{yx}(\mathbf{s}, z) \\ l_i & \text{otherwise} \end{cases}$$

□

Theorem 6.7 *E(FD-DEVS) ⊂ E(IOTA).*

Proof Given an FD-DEVS $M =< X, Y, S, \tau, \delta_x, \rho, \delta_\tau, \lambda >$, its behavior can be described by $A =< X, Y, L, C, I, T >$ such that $L = \{(s, t_s) | s \in S, t_s \in \bigcup_{s \in S} \{\tau(s)\}\}$; $C = \{c\}$; Let $Z = X \cup Y$ and suppose that $(s, t_s) \in L$ and $z \in Z^\epsilon$. Then $I : L \to \Phi(C)$ is defined as

$$I((s, t_s)) = \begin{cases} 0 \le c \le t_s & \text{for } t_s < \infty \\ 0 \le c & \text{otherwise.} \end{cases}$$

Let consider $T \subseteq L \times Z^\epsilon \times \Phi(C) \times 2^C \times L$ in the form of $((s, t_s), z, \varphi, C_R, (s', t'_s)) \in T$. It has possibilities depending on z as follows.

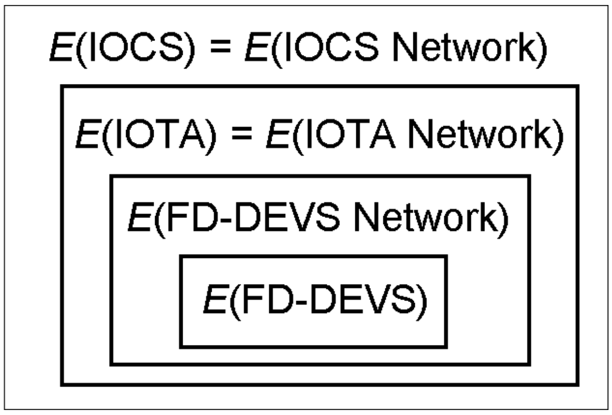


Figure 11. Expressiveness of IOCS Sub-classes

(i) If $z \in X$: Since $I((s, t_s))$ already defines the elapsed time constraints of FD-DEVS, we don't need additional $\varphi =$ _ but updating the next state s' and the schedule t'_s is depending on $\rho(s, x)$ such that
 a) if $\rho(s, x) = 1$ and $\delta_x(s, x) \perp \Rightarrow C_R = \{c\}; s' = \delta_x(s, z), t'_s = \tau(\delta_x(s, z))$.
 b) if $\rho(s, x) = 0$ and $\delta_x(s, x) \perp \Rightarrow C_R = \{\}; s' = \delta_x(s, z), t'_s = t_s$.

(ii) If $z \in Y^\epsilon$: Recall that the internal transition occurs when $c = t_s, z = \lambda(s), \delta_\tau(s) \perp$ as defined in Equation (15). So we need to define an edge for $z = \lambda(s), \delta_\tau(s) \perp \Rightarrow \varphi = (c = t_s)$; $C_R = \{c\}; s' = \delta_\tau(s), t'_s = \tau(\delta_\tau(s))$.

In other words, at least, we can say that E(FD-DEVS) $\subseteq E$(IOTA).

Let's check the reverse way, since the nondeterminism of IOTA and the determinism of FD-DEVS, E(IOTA) $\nsubseteq E$(FD-DEVS). Therefore, E(FD-DEVS) $\subset E$(IOTA). □

Example 6.8 (IOTA equivalent to FD-DEVS) Given the FD-DEVS controller shown in Figure 7(b), the IOTA controller shown in Figure 10 is equivalent to the origin FD-DEVS if there is no such an edge ((`I,I`), `?F:(r,d)`, _, $\{c\}$, (`I,C`)). □

Lemma 6.9 *E(FD-DEVS Network)* $\subset$ *E(IOTA).*

Proof Let's consider an FD-DEVS network whose all leaf nodes are FD-DEVS models. By Theorem 6.7, we can replace the leaf FD-DEVS models with equivalent IOTAs. Now we can have an IOTA network whose behavior is equivalent to the original FD-DEVS network. By Theorem 6.6, we can have an IOTA whose behavior is the same the IOTA network that is also equivalent to the original FD-DEVS network. Thus, we can say that E(FD-DEVS Network) $\subseteq E$(IOTA).

However, the clock constraints containing conditions of strict less than $<$ or strict grater than $>$ cannot be represented by an FD-DEVS network. Hence E(IOTA) $\nsubseteq E$(FD-DEVS Network). Therefore, E(FD-DEVS Network) $\subset E$(IOTA). □

Now to illustrate the expressive relation of IOCS sub-classes, we can draw a Venn diagram as shown in Figure 11.

7 Conclusion and Further Research

Now we can answer the three questions that arose in the introduction as follows.

Q.1 Does more relaxed but also verifiable DEVS exist?

A.1 *As a verifiable nondeterministic DEVS, we can use IOTA introduced in Section 5. To check if IOTA is a sub-class of non-deterministic DEVS, the reader can refer to Appendix.*

Q.2 Can we construct a hierarchical and modular TA network the same as we did in DEVS?

A.2 *For this, we defined first the IOTA as a sub-class of IOCS by splitting the event set of the ordinary TA into an input event set and an output event set, and then, we connected the events (or ports) as a coupling relation (refer to Section 5). It is a special case of the IOCS network introduced in Section 3.*

Q.3 Can the hierarchical and modular TA still be verifiable?

A.3 *Since the behavior of IOTA is the same as the ordinary TA and IOTA is closed under the coupling operation (by Theorem 6.6), we can verify the IOTA network as the ordinary TA.*

We can expect to develop the unified verification method converting to the IOTA from all verifiable sub-classes of IOCA such as an FD-DEVS, an FD-DEVS network and an IOTA network.

Besides, since the nondeterministic model tends to have less states than the deterministic model does, we can also expect to increase scalability by introducing a certain level of nondeterministic behavior as an abstraction. For abstraction or reduction of the system behavior, we might need to show the inclusion relation or equivalence between the detailed model and the abstracted model. For decidability of inclusion and equivalence, the reader can refer to (Alur and Dill, 1994).

Acknowledgment

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

References

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

Alur, R., Courcoubetis, C. and Dill, D., 1993, Model-checking in dense real-time. *Information and Computation*, **104**, 2–34.

Alur, R., Courcoubetis, C., Dill, D., Halbwachs, N. and Wong-Toi, H., 1992, Minimization of timed transition systems. In *Proceedings of the Proceedings of the Third Conference on Concurrency Theory*, 630 of *LNCS* (Springer), pp. 340–354.

Alur, R. and Dill, D., 1994, A theory of timed automata. *Theoretical Computer Science*, **126**, 183–235.

Behrmann, G., Larsen, K., Pearson, J., Weise, C. and Yi, W., 1999, Efficient timed reachability analysis using clock difference diagram. In *Proceedings of the In Computer Aided Verification*, LNCS (Springer).

Bozga, M., Maler, O., Pnueli, A. and Yovine, S., 1997, Some Progress in the symbolic verification of timed automata. In *Proceedings of the In Computer Aided Verification*, 1254 of *LNCS* (Springer), pp. 179–190.

Cutler, M., 1980, Discrete event simulation of stochastic and deterministic sequential machine models. In *Proceedings of the Proc. of Windter Simulation Conference*, Orlando, FL, pp. 83–97.

Dill, D.L., 1989, Timming Assumptions and Verification of Finite-State Concurrent Systems. In *Proceedings of the Proc. of the Workshop on Computer Aided Verification Methods for Finite State Systems*, Grenoble, France, pp. 197–212.

Henzinger, T., 1998, It's about time: Real-time logics reviewed. In *Proceedings of the In CONCUR '98: Ninth International Conference on Concurrency Theory*, 531 of *LNCS*, pp. 439–454.

Ho, Y.C., 1993, Forward to the Special Issue. *Discrete Event Dynamic Systems:Theory and Applications*, p. 111.

Hocaoglu, M.F., Zeigler, C.F.B.P. and Sarjoughian, H.S., 2004, Temporal System Identification and Model Verification. In *Proceedings of the Foundations '04: A Workshop for V& V in the 21st Century*, Oct 13-15 (Tempe, AZ: US Department of Defense).

Hong, G.P. and Kim, T.G., 1996, A Framework for Verifying Discrete Event Models whithin a DEVS-based System Development Methodology. *Transactions of The Society for Computer Simulation*, **13**, 19–34.

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

Hopcroft, J.E., Motwani, R. and Ullman, J.D., 2000, *Introduction to Automata Theory, Languages, and Computation*, second (Addison Wesley).

Hwang, M.H., 2005a, Generating Finite-State Behavior of Reconfigurable Automation Systems: DEVS Approach. In *Proceedings of the Proceed. of 2005 IEEE-CASE*, Edmonton,Canada.

Hwang, M.H., 2005b, Tutorial: Verification of Real-time System Based on Schedule-Preserved DEVS. In *Proceedings of the Proceedings of 2005 Spring Simulation Multi-Conference: Proceedings of 2005 DEVS Symposium*, Apr., San Diego, CA.

Hwang, M.H. and Choi, B.K., 1999, Message-Passing Architecture and its Construction in Object-Oriented Rapid Modeling for Automated Manufacturing System Simulation. *SIMULATION*, **72**, p90–104.

Hwang, M.H. and Zeigler, B.P., 2006a, A Modular Verification Framework using Finite & Deterministic DEVS. In *Proceedings of the Proceedings of 2006 Spring Simulation Multi-Conference: Proceedings of 2006 DEVS Symposium*, Apr. 2-8, Huntsville, AL, pp. 57–65.

Hwang, M.H. and Zeigler, B.P., 2006b, A Reachable Graph of Finite and Deterministic DEVS Networks. In *Proceedings of the Proceedings of 2006 Spring Simulation Multi-Conference: Proceedings of 2006 DEVS Symposium*, Apr. 2-8, Huntsville, AL, pp. 48–56.

Kofman, E., 2004, Discrete event simulation of hybrid systems. *SIAM Journal on Scientific Computing*, **25**, 1771–1797.

Lewis, R.W., 1998, *Programming Industrial Control System Using IEC 1131-3*, Revised (The Institution of Electrical Engineers).

Praehofer, H., 1991, Systems Theoretic Formalisms for Combined Discrete-Continuous System Simulation. *International Journal of General Systems*, **19**, 219–240.

Praehofer, H., Auernig, F. and Reisinger, G., 1993, An Environment for DEVS-Based Multiformalism Simulation in Common Lisp/CLOS. *Discrete Event Dynamic Systems: Theory and Applications*, **3**, 119–149.

Sargent, R.G., Mize, J.H., Withers, D.H. and Zeigler, B.P., 1993, Hierarchical Modelling for Discrete Event Simulation (Panel). In *Proceedings of the Proceedings of the 25th Winter Simulation Conference* (Los Angeles, CA: ACM Press).

Skahill, K., 1996, *VHDL for Programmable Logic* (Prentice Hall).

Tripakis, S. and Yovine, S., 2001, Analysis of Timed Systems Using Time-Abstracting Bisimulations. *Formal Methods in System Design*, **18**, 25–68.

Wang, J., 1998, *Timed Petri Nets: Theory and Application* (Boston: Kluwer Academic Publishers).

Zeigler, B.P., 2006, Embedding DEV&DESS in DEVS: Characteristic Bahaviors of Hybrid Models. In *Proceedings of the Proceedings of 2006 Spring Simulation Multi-Conference: DEVS Integrative M&S Symposium* (Huntsville, AL, USA: SCS), pp. 125–132.

Zeigler, B.P. and Chi, S., 1992, Symbolic Discrete Event System Specification. *IEEE Transactions on Systems, Man, and Cybernetics*, **22**, 1428–1443.

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

Zeigler, B.P., 1976, *Theory of Modelling and Simulation*, First (New York: Wiley Interscience).

Zeigler, B.P., 1984, *Multifacetted Modeling and Discrete Event Simulation*, First (London,Orlando: Academic Press).

Zeigler, B.P., 1990, *Object-oriented Simulation with Hierarchical, Modular Models:Intelligent Agents and Endomorphic Systems*, second (Boston, Mass. and San Diego, Calif.: Academic Press).

Appendix A: Expressiveness of nondeterministic DEVS and IOTA

This appendix shows the expressiveness between nondeterministic DEVS (N-DEVS) and IOTA. N-DEVS M is a 7-tuple

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

where

- X and Y are disjoint sets of input and output, respectively.
- S is a set of states.
- $\tau : S \rightarrow 2^{R_{[0,\infty]}}$ is the time advance function.
- $\delta_x : Q \times X \rightarrow 2^S$ is the external transition function where $Q = \{(s, c_e)|s \in S, 0 \le c_e \le \tau(s)\}$ is the total state set and c_e is a clock tracking the elapsed time since the *last event.*
- $\delta_\tau : S \rightarrow 2^S$ is the internal transition function.
- $\lambda : S \rightarrow 2^{Y^\epsilon}$ is the output transition function.

N-DEVS is a special class of IOCA so its behavior can be described as an IOCS $\mathcal{C} =< X, Y, Q, \eta, \delta >$ $Q = \{(s, c_e)|s \in S, 0 \le c_e \le t_s \in \tau(s)\}$. Let $Z^\epsilon = X \cup Y^\epsilon$ and suppose $(s, c_e) \in Q$ and $z \in Z^\epsilon$. Then the enabled indicating function $\eta : Q \times Z^\epsilon \rightarrow \{0, 1\}$ is defined as

$$\eta((s, c_e), z) = \begin{cases} 1 & \text{for } z \in X \\ 1 & \text{for } \exists z \in \lambda(s) \text{ s.t. } c_e = t_s \in \tau(s), \delta_\tau(s) \perp \\ 0 & \text{otherwise} \end{cases}$$

That is the enable condition of the external event $z \in X$ is always true, while that of the internal event $z \in Y^\epsilon$ is only true when the elapsed time reaches one of the possible schedule time candidates $(c_e = t_s \in \tau(s))$, the associate event is one of the possible outputs $(z \in \lambda(s))$, and the internal transition function is defined $(\delta_\tau(s) \perp)$. When this condition holds, the transition function $\delta : Q \times Z^\epsilon \rightarrow 2^Q$ is performed. This function is defined as

$$\delta((s, c_e), z) = \begin{cases} \{(s, 0)|s \in \delta_x((s, c_e), z)\} & \text{for } z \in X, \delta_x((s, c_e), z) \perp \\ \{(s, 0)\} & \text{for } z \in X, \delta_x((s, c_e), z) \not\perp \\ \{(s, 0)|s \in \delta_\tau(s)\} & \text{for } z \in Y \end{cases}$$

Notice that the elapsed time clock c_e is reset to 0 at any time an external event $x \in X$ happens regardless of whether $\delta_x(q, x)$ is defined or not (Zeigler, 1976; Zeigler *et al.*, 2000). As we defined the behavior of IOCS using GDS in Section 3.2, we can rewrite the discrete state transition of N-DEVS using GDS $G =< X, Y, Q, f, \delta_G >$ such that the derivative function f is the same as the IOCA which is defined as Equation (6) while the discrete state transition function is as follows.

$$\delta_G((s, c_e), z) = \begin{cases} \{(s, 0)|s \in \delta_x((s, c_e), z)\} & \text{for } z \in X, \delta_x((s, c_e), z) \perp \\ \{(s, 0)\} & \text{for } z \in X, \delta_x((s, c_e), z) \not\perp \\ \{(s, 0)|s \in \delta_\tau(s)\} & \text{for } \exists z \in \lambda(s) \text{ s.t. } c_e = t_s \in \tau(s), \delta_\tau(s) \perp \\ \{(\mathsf{error}, 0)\} & \text{otherwise} \end{cases} \tag{A1}$$

Theorem A.1 *E(IOTA) ⊂ E(N-DEVS).*

Proof We first show $E(\text{IOTA}) \subseteq E(\text{N-DEVS})$. The behavior of an IOTA $A =< X, Y, L, C, I, T >$ defined in the form of IOCS can be rewritten by GDS $\mathcal{G} =< X, Y, Q, f, \delta_G >$ such that $Q = \{(l, \mathbf{c})|l \in L, \mathbf{c} \in V(C)\}$. The derivative function f is the same as the IOCA which is defined as Equation (6). But the discrete state transition function is that for $(l, \mathbf{c}) \in Q$ and $z \in Z^\epsilon = X \cup Y^\epsilon$,

$$\delta_G((l, \mathbf{c}), z) = \begin{cases} \{(l', \mathbf{c}[C_R := 0]))|(l, z, \varphi, C_R, l') \in ET(l, \mathbf{c}, z)\} & \text{for } ET(l, \mathbf{c}, z) \neq \varnothing \\ \{(l, \mathbf{c})\} & \text{for } z \in X, ET(l, \mathbf{c}, z) = \varnothing \\ \{(\mathsf{error}, \mathbf{c})\} & \text{for } z \in Y^\epsilon, ET(l, \mathbf{c}, z) = \varnothing \end{cases} \tag{A2}$$

where $ET(l, \mathbf{c}, z) = \{(l, z, \varphi, C_R, l') \in T|\mathbf{c} \in I(l) \wedge \varphi\}$ (see Equation (17)).

We can build N-DEVS $M =< X, Y, S, \tau, \delta_x, \delta_\tau, \lambda >$ whose behavior is equivalent to the original IOTA A as follows. Let $Z^\epsilon = X \cup Y^\epsilon$, $S = L \times V(C)$ so that the total state set $Q = \{((l, \mathbf{c}), c_e)|(l, \mathbf{c}) \in S, 0 \leq c_e = t_s \leq \tau((l, \mathbf{c}))\}$.

Given a clock constraint $\varphi \in \Phi(C)$, the *clock constraint for* $c_i \in C$ is

$$\varphi(c_i) = t_l \prec_l c_i \prec_u t_u$$

where $t_l, t_u \in Q_{[0,\infty]}$ s.t. $t_l \leq t_u$ and $\prec_l, \prec_u \in \{<, \leq\}$. The *remaining time constraint* of c_i is defined as

$$\varphi^r(c_i) = \begin{cases} t_l - c_i \prec_l \sigma \prec_u t_u - c_i & \text{for } t_l \not\prec_l \ c_i \\ 0 \leq \sigma \prec_u t_u - c_i & \text{for } t_l \prec_l c_i \prec_u t_u. \end{cases}$$

The *remaining time constraint* of all $c_i \in C$ is $\varphi^r(\mathbf{c}) = \bigwedge_{c_i \in C} \varphi^r(c_i)$. Let a set of internal transitions at a location l be $IT(l) = \{(l, y, \varphi, C_R, l') \in T|y \in Y^\epsilon\}$. Based on $IT(l)$ and $\varphi^r(\mathbf{c})$, the time advance function $\tau : S \to 2^{\mathbb{R}_{[0,\infty]}}$ defines the *possible schedule zone of internal transitions* as

$$\tau(l, \mathbf{c}) = \begin{cases} \bigcup_{(l,y,\varphi,C_R,l') \in IT(l)} \{t|t \in \varphi^r(\mathbf{c})\} & \text{for } IT(l) \neq \varnothing \\ \{\infty\} & \text{otherwise.} \end{cases} \tag{A3}$$

For $x \in X$,

$$\delta_x((l, \mathbf{c}, c_e), x) = \{(l', \mathbf{c}[C_R := 0])|(l, x, \varphi, C_R, l') \in ET(l, \mathbf{c}, x)\} \tag{A4}$$

$$\delta_\tau((l, \mathbf{c})) = \{(l', \mathbf{c}[C_R := 0])|(l, y, \varphi, C_R, l') \in ET(l, \mathbf{c}, y), y \in Y^\epsilon\} \tag{A5}$$

$$\lambda((l, \mathbf{c})) = \{y|(l, y, \varphi, C_R, l') \in ET(l, \mathbf{c}, y), y \in Y^\epsilon\} \tag{A6}$$

Let's substitute $\tau, \delta_x, \delta_\tau$ and λ in δ_G in Equations (A1) describing the behavior of N-DEVS, by these of Equations (A3), (A4), (A5) and (A6).

$\delta_G((l, \mathbf{c}, c_e), z) =$

$$\begin{cases} \{(l', \mathbf{c}[C_R := 0], 0)|(l, z, \varphi, C_R, l') \in ET(l, \mathbf{c}, z)\} & \text{for } z \in X, ET(l, \mathbf{c}, z) \neq \varnothing \\ \{(l, \mathbf{c}, 0)\} & \text{for } z \in X, ET(l, \mathbf{c}, z) = \varnothing \\ \{(l', \mathbf{c}[C_R := 0], 0)|(l, z, \varphi, C_R, l') \in ET(l, \mathbf{c}, z)\} & \text{for } z \in \lambda(l, \mathbf{c}), c_e = t_s \in \tau(l, \mathbf{c}), ET(l, \mathbf{c}, z) \neq \varnothing \\ \{(\mathsf{error}, \mathbf{c}, 0)\} & \text{otherwise} \end{cases} \tag{A7}$$

Let's check the condition of the third line of the above equation. The first part $z \in \lambda(l, \mathbf{c})$ can be rewritten by Equation (A6) as $z \in \{y|(l, y, \varphi, C_R, l') \in ET(l, \mathbf{c}, y), y \in Y^\epsilon\}$; The second part $c_s = t_s \in \tau(l, c)$ can be re-interpreted by Equation (A3) as $ET(l, \mathbf{c}, z) \neq \varnothing$; Thus, the we can write the whole third condition of Equation (A7) as $y \in Y^\epsilon, ET(l, \mathbf{c}, z) \neq \varnothing$. Then, we can get Equation (A7) that as Equation (A2). Now we can say that E(IOTA) $\subseteq$ E(N-DEVS).

Let's check if E(N-DEVS) $\subseteq$ E(IOTA). Since the time advance function $\tau : S \to 2^{\mathbb{R}_{[0,\infty]}}$ of N-DEVS allows an irrational number. For example $\tau(s) = \sqrt{5} < c_1 \leq \sqrt{7}$. This schedule region of s cannot be expressed by IOTA whose clock constraints do not allow any irrational number. So E(N-DEVS) $\not\subseteq$ E(IOTA). Therefore, we can say that E(IOTA) $\subset$ E(N-DEVS). □

Corollary A.2 *We can simulate the IOTA using N-DEVS.*