Simulation Processors and Their Collaboration for DEVS Models Integrated with Universal Coupling Specification

Sunwoo Park[1] **Sean HJ Kim**[2] **C. Anthony Hunt**[1,2]
[1]**BioSystems Group, Dept. of Biopharmaceutical Sciences, University of California, San Francisco;**
[2]**Joint Graduate Group in Bioengineering, University of California, Berkeley and San Francisco,**
513 Parnassus Ave., San Francisco, CA 94143-0446, USA
parks@pharmacy.ucsf.edu seanhjk@berkeley.edu a.hunt@ucsf.edu

Abstract

We present a set of simulation processors and describe their temporal collaboration in support of the execution of models specified by an advanced Discrete Event Systems Specification (DEVS) formalism. Our formalism improves the existing parallel DEVS formalism by adopting the universal coupling mechanism. The mechanism enables coupled models to create, permute, and delete their couplings in a proactive or reactive manner. Each coupled model is a decomposable, constructive, multi-scale, discrete-event system that contains at least one component. Proposed simulation processors will allow exploration of spatiotemporal spaces generated from specified models. A simulation protocol is realized as a sequence of well-defined temporal interactions between those processors. It is expected that current modeling and simulation (M&S)-driven systems biology will be empowered by the new DEVS formalism and proposed simulation processors.

1 Introduction

Many complex biological phenomena result from dynamic spatiotemporal interactions between components and subsystems. To cope with the dynamic nature of the interactions, several computational methods have been applied. Process algebra describes an interaction between systems as a set of communication link and its dynamic permutation [1][2][3]. It focuses mainly on relationships between systems rather than on their internal temporal dynamics. Meanwhile, discrete event systems and discrete-event driven M&S focus on the temporal dynamics of systems [4][5][6]. They describe an interaction by message (or event) exchange or propagation links. The message is consumed or produced by a system as the result of a reactive response to incoming messages or its proactive dynamic state transitions. Specifically, agent-based M&S describes system level biological phenomena as resulting from interactions between agents, where some agents represent biological entities and others the environment within which these entities reside [7]. An agent is commonly postulated to be an intelligent and autonomous (discrete event) system. Membrane computing, also known as P-system, provides similar capabilities for dealing with dynamic interactions covered by process algebra from a system-oriented perspective [8]. However, it does not explicitly discuss temporal aspects of the relation. It is desirable to merge the advantages of above approaches to efficiently describe and explore complex dynamic interactions of the type that occur between and inside biological systems.

We have developed an advanced DEVS modeling formalism to describe adaptive spatiotemporal interactions of biological systems and the phenomena they produce [9]. The formalism has been applied to a set of biology problems including DNA formation and synthesis. It demonstrated the power and effectiveness of the new formalism in describing dynamic spatiotemporal interactions and the temporal dynamics that occur between and within biological systems.

In this paper, we present a set of simulation processors and describe their temporal collaboration to execute models described by the new modeling formalism. Proposed simulation processors extend an existing parallel DEVS abstract simulator and coordinators by integrating the universal interaction mechanism [9][10]. The universal interaction specification and revised DEVS model specifications will be briefly described before simulation processors are presented.

2 Universal coupling

A coupling is a directional influence from one entity to another. The influencer and influencee are commonly described as a pair: a model and its property (e.g., communication I/O port). It is classified into three categories: *static*, *dynamic*, and *adaptive*.

A *static* coupling consists of one influencer (m_i, p_i) and one influencee (m_j, p_j). It represents a causal relation from the influencer to the influencee. The coupling is specified by a 4-tuple $\mathcal{C}^s_{i,j} : (m_i, p_i, m_j, p_j)$ where $m_i, m_j \in M$, $p_i, p_j \in P$, M is a set of models, and P is a set of properties.

A *dynamic coupling* is an extension of the static coupling that enables its dynamic permutation. Dynamic permutation includes the change of influencer or influecee, construction of a new static or dynamic coupling, and destruction of an existing dynamic coupling. It is specified by a 9-tuple $\mathcal{C}^d_{i,j} : (m_i, p_i, m_j, p_j, m_k, p_k, \xi, \rho, l)$ where $m_i, m_j, m_k \in M$, $p_i, p_j, p_k \in P$, ξ is a coupling constraint, $\rho : \mathcal{C}^d \rightarrow \mathcal{C}^d$ is a coupling permutation function, and $l \in Z^+$ is a lifespan variable. It can be alternatively specified by an endomorphic 5-tuple $\mathcal{C}^d_{i,j} : (C^s_{i,j}, m_k, p_k, \xi, \rho, l)$ where $C^s_{i,j}$ is a static coupling, m_k, p_k, ξ, ρ, and l are as same as above. (m_k, p_k) is the candidate that will be promoted to a new influencer or influencee after dynamic permutation. ξ specifies when the permutation occurs. Satisfying ξ triggers ρ. ρ provides details on how the coupling is changed. The dynamic coupling is valid only when $l > 0$. l can be increased or decreased after execution of ρ. An important property of the coupling is its capability to construct, permute, and evolve coupling networks from an existing coupling or coupling network. By tracking changes in network topology, we can trace dynamic coupling permutation between models during their lifetime.

An *adaptive coupling* is an extension of the dynamic coupling that permits meta entities in the coupling specification. The entities are not precisely described in an initial specification. However, they are adaptively resolved and bound to concrete entities at later times. The coupling is mainly used to adaptively create a dynamic coupling in a proactive or reactive maner. It divides into two adaptive couplings - *proactive* and *reactive*. A *proactive coupling* is an extension of the dynamic coupling that contains a meta influencee or influencer $(*, *)$, a binding constraint ψ, and a binding function $\varphi : \mathcal{C}^{a_p} \rightarrow \mathcal{C}^d$. $(*, *)$ is resolved and bound by φ when ψ is satisfied. φ creates a dynamic coupling. The coupling is represented by a 11-tuple $\mathcal{C}^{a_p}_{i,*} : (m_i, p_i, *, *, m_k, p_k, \xi, \rho, l, \psi, \varphi)$ or $\mathcal{C}^{a_p}_{*,j} : (m_*, p_*, m_j, p_j, m_k, p_k, \xi, \rho, l, \psi, \varphi)$, respectively, depending on the existence of the meta influencer or influencee. It is alternatively simplified to an endormorphic 3-tuple - $\mathcal{C}^{a_p}_{i,*} : (\mathcal{C}^d_{i,*}, \psi, \varphi)$ or $\mathcal{C}^{a_p}_{*,j} : (\mathcal{C}^d_{*,j}, \psi, \varphi)$. A *reactive coupling* is an adaptive coupling that contains a meta candidate. Unlike the proactive coupling, the influencer and influencee are "clearly" presented but the candidate (m_k, p_k) is not precisely defined in the coupling. The coupling is denoted as $\mathcal{C}^{a_r}_{i,j} : (m_i, p_i, m_j, p_j, *, *, \xi, \rho, l, \psi, \varphi)$. The binding function $\varphi : \mathcal{C}^{a_r} \rightarrow \mathcal{C}^d$ resolves the meta candidate when ψ is satisfied. $\mathcal{C}^{a_r}_{i,j} : (\mathcal{C}^{d_*}_{i,j}, \psi, \varphi)$ is its endomorphic representation.

A universal coupling $\mathcal{C}^u_{i,j} : \mathcal{C}^s_{i,j} \oplus \mathcal{C}^d_{i,j} \oplus \mathcal{C}^{a_p}_{i,*} \oplus \mathcal{C}^{a_p}_{*,j} \oplus \mathcal{C}^{a_r}_{i,j}$ is the composition of all couplings introduced above. See [4] and [9] for more details on coupling mechanisms.

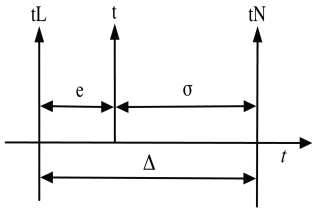
3 DEVS coupled model with universal coupling

Our new DEVS coupled model strengthens the existing coupling specification of Parallel DEVS (P-DEVS) by replacing it with the universal coupling specification [4][9][10]. For the congruent integration of the universal coupling specification with the DEVS formalism, we use causal relation forms instead of tuples. Specifically, $(m_i, p_i) \rightarrow (m_j, p_j)$ for $C^s_{i,j}$, $(m_i, p_i) \xrightarrow{m_k, p_k, \xi, \rho, l} (m_j, p_j)$ for $C^d_{i,j}$, $(m_*, p_*) \xrightarrow{m_k, p_k, \xi, \phi, \rho, l, \psi, \varphi} (m_j, p_j)$ for $C^{a_p}_{*,j}$, $(m_i, p_i) \xrightarrow{m_k, p_k, \xi, \phi, \rho, l, \psi, \varphi} (m_*, p_*)$ for $C^{a_p}_{i,*}$, and $(m_i, p_i) \xrightarrow{m_*, p_*, \xi, \phi, \rho, l, \psi, \varphi} (m_j, p_j)$ for $C^{a_r}_{i,j}$.

The DEVS coupled model with the universal coupling N is $\langle X, Y, D, \{M_d\}, \{I_d\}, \{Z_{d,i}\}\rangle$. X is a set of inputs. Y is a set of outputs. D is a set of components. For each $d \in D$, M_d is a DEVS model, which is either atomic or coupled. For each $d \in D \cup \{self\}$, I_d is the set of influencees of d. $self$ is a reserved reference of N itself. For each $i \in I_d$, $Z_{d,i}$ is a function, d-to-i output translation with $Z_{d,i} \in (Z^s_{d,i} \cup Z^d_{d,i} \cup Z^p_{d,i} \cup Z^r_{d,i})$. $Z^s_{d,i}$ is the static translation function with (i) $Z_{d,i}$: $X \rightarrow X_i$, if d=$self$; (ii) $Z_{d,i}$: $Y_d \rightarrow Y$, if i=$self$; and (iii) $Z_{d,i}$: $Y_d \rightarrow X_i$, if $d \neq self \wedge i \neq self$. $Z^d_{d,i}$ is the dynamic translation function with (i) $Z_{d,i}$: $X \xrightarrow{X_k, \xi, \rho, l} X_i$, if d=$self$; (ii) $Z_{d,i}$: $Y_d \xrightarrow{Y_k, \xi, \rho, l} Y$, if i=$self$; and (iii) $Z_{d,i}$: $Y_d \xrightarrow{X_k, \xi, \rho, l} X_i$, if $d \neq self \wedge i \neq self$. $Z^p_{d,i}$ is the proactive translation function with (i) $Z_{d,i}$: $X \xrightarrow{X_k, \xi, \rho, l, \psi, \varphi} X_*$, if d=$self$; (ii) $Z_{d,i}$: $Y_* \xrightarrow{Y_k, \xi, \rho, l, \psi, \varphi} Y$, if i=$self$; and (iii) $Z_{d,i}$: $Y_* \xrightarrow{X_k, \xi, \rho, l, \psi, \varphi} X_i$ or $Y_d \xrightarrow{X_k, \xi, \rho, l, \psi, \varphi} X_*$, if $d \neq self \wedge i \neq self$, where $* \in D$ is an unspecified component. $Z^r_{d,i}$ is the reactive translation function with (i) $Z_{d,i}$: $X \xrightarrow{X_*, \xi, \rho, l, \psi, \varphi} X_i$, if d=$self$; (ii) $Z_{d,i} : Y_d \xrightarrow{Y_*, \xi, \rho, l, \psi, \varphi} Y$, if i=$self$; and (iii) $Z_{d,i}$: $Y_d \xrightarrow{X_*, \xi, \rho, l, \psi, \varphi} X_i$, if $d \neq self \wedge i \neq self$.

4 Simulation protocol

For simulation time management, we use t_L, t, t_N, e, σ to represent the time the last event occurs, the current simulation time, the time the next event will occur, the elapsed time since t_L, and the time left until t_N, respectively. Specifically, $e = t - t_L$, $\sigma = t_N - t = t_N - (t_L + e)$, and $t_N = t_L + \tau(s) = t + \sigma$ where $t_L \leq t \leq t_N$.



tL: time of the last event t: current time tN: time of the next event
e: elapsed time since the last event σ: remaining time to the next event
Δ: inter-arrival time between two adjacent events

Figure 1: DEVS simulation time

There exist three different simulation processors - *root coordinator*, *coordinator*, and *simulator*. A root coordinator controls the top-most coordinator. Each coordinator controls its components by exchanging simulation control messages with both its parent coordinator and child components. Each simulator executes its associated atomic model whenever it receives a simulation control message from its parent coordinator. Algorithm 1 presents an atomic simulator that refines the original P-DEVS abstract simulator.

Algorithm 1 Simulator for an atomic model

1: $t_L := e := 0$; $t_N := \tau(s)$; $x^b = \phi$
2: **when** $(\%, t)$ message arrives from parent
3: send $(\%, \tau(s)$ - $(t$ - $t_L))$ to parent
4: **end when**
5: **when** $(@, t)$ message arrives from parent
6: send $(y, ((t = t_N)\ ?\ \lambda(s) : \phi), t)$ to parent
7: **end when**
8: **when** (x, x^b_d, t) message arrives from parent
9: $x^b := x^b \oplus x^b_d$
10: send $(done, t)$ to parent
11: **end when**
12: **when** $(*, t)$ message arrives from parent
13: **if** $t < t_L \vee t > t_N$ **then** timing synchronization error
14: **else**
15: $e := t$ - t_L
16: **if** $t_L \leq t \leq t_N$ **then**
17: $s := (x^b \neq \phi)\ ?\ \delta_{ext}(s, e, x^b) : s$
18: **else if** $t = t_N$ **then**
19: $s := (x^b = \phi)\ ?\ \delta_{int}(s) : \delta_{conf}(s, e, x^b)$
20: **else** raise error
21: **end if**
22: $t_L := t$; $t_N := t_L + \tau(s)$; $x^b := \phi$
23: send $(done, t_N)$ to parent
24: **end if**
25: **end when**

A coordinator advances the simulation time by the amount of σ_{min}, which is $min\{\sigma_d \,|\, d \in D\}$ where $\sigma_d = \tau(s_d) - e_d$. s_d, e_d, $\tau(s_d)$, and σ_d are s, e, $\tau(s)$, and $\sigma(s)$ of $M_d \in M$, respectively. IMM is the set of imminent components that produce outputs, $\{d \,|\, \sigma_d = \tau(s_d) \wedge \lambda^b_d(s_d) \neq \phi\}$, at time $t_L + \sigma_{min}$. $\lambda^b_d(s_d)$ is the bag of output messages produced by M_d. INF is the set of components influenced by $d \in IMM$, $\{i \,|\, i \in I_d, d \in IMM \wedge x^b_i \neq \phi\}$ where $x^b_i = \{Z_{d,i}(\lambda^b_d(s_d)) \,|\, d \in IMM \cap I_d\}$. It is equivalent to $\{d \,|\, d \in D \wedge x^b_d \neq \phi\}$ in the algorithm we present here because $\lambda^b_d(s_d)$ produced by each $d \in IMM$ is translated to $\{x^b_i\}_{i \in D}$ through $Z_{d,i}(\lambda^b_d(s_d))$ based on I_d before INF is computed. Dynamic or adaptive permutation is managed when the coordinator receives the message $(!, t)$ from its parent coordinator. A collection of Algorithm 2 and 3 represents a coordinator that refines and extends the P-DEVS coordinator with the integration of the universal coupling specification.

Algorithm 2 Coordinator for a coupled model: Part I

1: $t_L := t_N := e := 0$; $sync := \phi$
2: **when** $(\#, t)$ message arrives from parent
3: $sync := D$
4: **for each** $d \in D$ **do** send $(\%, t)$ to child d **end for**
5: **end when**
6: **when** $(\%, t)$ message arrives from child d
7: $\sigma_d := t$; $sync := sync \backslash d$
8: **if** $|sync| = 0$ **then**
9: $t_N := t_L + min\{\sigma_d\}$
10: send $(done, t_N)$ to parent
11: **end if**
12: **end when**
13: **when** $(@, t)$ message arrives from parent
14: $IMM := INF := \phi$; $sync := D$
15: **for each** $d \in D$ **do**
16: $x^b_d := \phi$
17: send $(@, t)$ to child d
18: **end for**
19: **end when**
20: **when** (y, y^b, t) message arrives from child d
21: $sync := sync \backslash d$
22: **if** $y^b \neq \phi$ **then**
23: $IMM := IMM \oplus d$
24: **for each** $i \in I_d \backslash self$ **do** $x^b_i := x^b_i \oplus z_{d,i}(y^b)$ **end for**
25: **if** $self \in I_d$ **then** $y^b_{self} := y^b_{self} \oplus z_{d,self}(y^b)$ **end if**
26: **end if**
27: **if** $|sync| = 0$ **then**
28: send (y, y^b_{self}, t) to parent
29: $y^b_{self} := \phi$
30: **end if**
31: **end when**
32: **when** (x, x^b, t) message is received from parent
33: **for each** $i \in I_{self} \wedge x^b \neq \phi$ **do**
34: $x^b_i := x^b_i \oplus z_{self,i}(x^b)$
35: **end for**
36: **for each** $d \in D \wedge x^b_d \neq \phi$ **do**
37: $INF := INF \oplus d$
38: **end for**
39: $sync := INF$
40: **for each** $d \in INF$ **do** send (x, x^b_d, t) to child d **end for**
41: **end when**

Algorithm 3 Coordinator for a coupled model: Part II

```
42: when (!, t) message is received from parent
43:     for each z in Z do
44:         if z ∈ (I_a^{a_p} ⊕ I_a^{a_r} ⊕ I_e^{a_p} ⊕ I_e^{a_r}) then
45:             if z.ψ is true then execute z.φ end if
46:         else if z ∈ (I_a^d ⊕ I_e^d) then
47:             if z.ξ is true then execute z.ρ end if
48:         end if
49:     end for
50: end when
51: when (∗, t) message is received from parent
52:     e := t - t_L
53:     if t_L ≤ t ≤ t_N then
54:         sync := IMM ⊕ INF
55:         for each d ∈ IMM ⊕ INF do
56:             send (∗, t) to child d end for
57:         end for
58:     else raise an error
59:     end if
60:     t_L := t
61: end when
62: when (done, t) message arrives from child d
63:     sync := sync\d
64:     if |sync| = 0 then send (done, t) to parent end if
65: end when
```

The root coordinator initiates the whole simulation cycle by sending the $(\#, t_N)$ message to the top-most coordinator and terminates it if t_N is ∞.

Algorithm 4 Root coordinator

```
1: t_N := 0
2: while t_N ≠ ∞ do
3:     send (#, t_N) to the top coordinator
4:     wait for (done, t_N) from the top coordinator
5:     send (@, t_N) to the top coordinator
6:     wait for (y, y^b, t_N) from the top coordinator
7:     send (x, x^b, t_N) to the top coordinator
8:     wait for (done, t_N) from the top coordinator
9:     send (!, t_N) to the top coordinator
10:    wait for (done, t_N) from the top coordinator
11:    send (∗, t_N) to the top coordinator
12:    wait for (done, t_N) from the top coordinator
13: end while
```

5 Conclusion

We present a set of three simulation processors to execute and control models specified using the advanced DEVS modeling formalism. The formalism defines coupling relations between components of a coupled model with the universal coupling specification. The formalism enables dynamic creation, permutation, and removal of couplings in a proactive or reactive manner. With this feature, the coupling networks between components in a coupled model can be evolved during simulation. The evolution can be traced by tracking changes in the coupling network topology. Proposed simulation processors enable exploration of spatiotemporal spaces generated by dynamic construction and permutation of interactions between biological system components. Our expectation is that these processors and the revised formalism will facilitate progress in how the hierarchical, multi-scale mechanistic details of biological systems can be modeled, coupled, and simulated.

Acknowledgements

This research was funded in part by CDH Research Foundation, Postdoctoral Fellowship provided to SP by CDH R.F., and Graduate Fellowship to SHJK from IFER. We thank Prof. Bernard P. Zeigler and other members of the BioSystems Group for helpful discussion and commentary.

References

[1] R. Miller, *Communicating and mobile systems: the π-calculus*. Cambridge University Press, 1999.

[2] C. Priami, "Stochastic π-calculus," *The Computer Journal*, vol. 38, no. 7, pp. 578–589, 1995.

[3] A. Regev, E. M. Panina, W. Silverman, L. Cardelli, and E. Shapiro, "Bioambients: an abstraction for biological compartments," *Theoretical Computer Science*, vol. 325, no. 1, pp. 141–167, 2004.

[4] B. P. Zeigler, T. G. Kim, and H. Praehofer, *Theory of Modeling and Simulation*, ser. 2nd edition. Academic Press, 2000.

[5] J. Banks, I. John S. Carson, and B. L. Nelson, *Discrete-Event System Simulation*. Prentice-Hall, 1996.

[6] R. Y. Rubinstein and B. Melamed, *Modern Simulation and Modeling*. Willey, 1998.

[7] P. Davidsson, B. Loga, and K. Takadama, Eds., *Multi-Agent and Multi-Agent-Based Simulation: Joint Workshop MABS 2004*, ser. Lecture Notes in Computer Science, vol. 3415. Springer, 2005.

[8] G. Paun, "Computing with membranes," *Journal of Computer and System Sciences*, vol. 61, pp. 108–143, 2000.

[9] S. Park and C. A. Hunt, "Coupling permutation and model migration based on dynamic and adaptive coupling mechanisms," in *The Proceedings of the 2006 DEVS Symposium*, 2006.

[10] A. C. Chow and B. P. Zeigler, "Parallel DEVS : A parallel, hierarchical, modular modeling formalism," in *Proceedings of the 1994 Winter Simulation Conference*, 1994.