Proceedings of the 2012 Winter Simulation Conference
C. Laroque, J. Himmelspach, R. Pasupathy, O. Rose, and A. M. Uhrmacher, eds.

A COMPOSITIONAL APPROACH FOR MODELING AND SIMULATION OF BIO-MOLECULAR SYSTEMS

Fernando J. Barros

Universidade de Coimbra
Departamento de Engenharia Informática
PT-3030 Coimbra, PORTUGAL

ABSTRACT

The simulation of biological systems is a challenge to modeling methodologies. Living entities are supported by a complex hierarchical network of chemical reactions. The accurate representation of organisms require the use of stochastic chemical equations (SCE) organized into compartments in order to reflect the organization into cells, tissues and organs. In this paper we introduce a modeling and simulation framework based on the Heterogeneous Flow System Specification formalism (HFSS). HFSS provides an hybrid representation of dynamic systems, being able to describe sampled and discrete event systems. These features enable a modular representation of stochastic chemical reactions. In particular, we show that SCE require only two types of HFSS models defined in this paper: molecule holders, and chemical reactors. We provide a description of a HFSS implementation based on pluggable software units (PUs), a component-based framework that supports the development of SCE libraries.

1 INTRODUCTION

The simulation of biological systems requires the ability to model at different levels of abstraction. At the basic level, biological systems are commonly represented by a network of chemical reactions organized into compartments. Since these compartments have a small number of molecules, the accurate representation of chemical reactions require the ability to describe each reaction individually, involving the numerical solution of stochastic chemical equations (SCEs) (Gillespie 1977). Living organisms, like cells, exhibit a dynamic life-cycle that includes reproduction and death. To faithfully represent this behavior, a sound modeling and simulation formalism needs to have the ability to modify model topology with the creation and destruction of components. The ability to represent the activation/inhibition of chemical reactions in cells can also be achieved with changes of topology.

The Heterogeneous Flow System Specification (HFSS) is a modeling formalism supporting the constructs to support the representation of hybrid modular systems with a dynamic topology. In this paper we exploit HFSS capabilities to develop a modular representation of SCEs organized in a network of hierarchical components with dynamic topology. Modularity is achieved by HFSS capability to represent dynamic systems based on the combination of discrete events (Zeigler 1976) and generalized sampling. These features enable a modular representation of molecule holders/bins and chemical reactions. We show that these two types of components can be employed to define arbitrary networks of chemical stochastic reactions.

We provide here the HFSS description of these two elements along with a formal definition of a small network of chemical reactions given by Paulsson, Berg, and Ehrenberg (2000).

The HFSS formalism is implemented in the JUSE (Barros 2011), a programming language supporting reusable components based on independent and pluggable software units (PUs). We describe model implementation in JUSE and we present preliminary simulation results.

978-1-4673-4781-5/12/$31.00 ©2012 IEEE

2 THE HFSS FORMALISM

The Heterogeneous Flow System Specification (HFSS) is a formalism aimed to represent hybrid modular systems (Barros 2003). HFSS defines two types of models: basic and network. Basic models provide state representation and state transition functions. Network models are a composition of basic models and/or other network models. Given its definition, a network provides an abstraction for representing hierarchical systems. HFSS networks have a dynamic topology that can be modified at runtime.The HFSS uses as the time base the set of Hyperreals **H** (Goldblatt 1998), a set that defines the infinitesimal number ε. As a particular case of topology adaptation, HFSS can also represent the migration of components between networks (Barros 2005).

2.1 HFSS Basic Model

In this section we describe HFSS basic models and their semantics. We consider $\widehat{B}$ as the set of names corresponding to basic HFSS models. A HFSS basic model associated with name $B \in \widehat{B}$ is defined by:

$$M_B = (X, Y, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda)$$

where

$X = \bar{X} \times \ddot{X}$ is the set of input flow values

$\bar{X}$ is the set of continuous input flow values

$\ddot{X}$ is the set of discrete input flow values

$Y = \bar{Y} \times \ddot{Y}$ is the set of output flow values

$\bar{Y}$ is the set of continuous output flow values

$\ddot{Y}$ is the set of discrete output flow values

P is the set of partial states (p-states)

$\rho : P \longrightarrow \mathbf{H}^0_{+\infty}$ is the time-to-input function

$\omega : P \longrightarrow \mathbf{H}^0_{+\infty}$ is the time-to-output function

$S = \{(p,e) | p \in P, 0 \leq e \leq \nu(p)\}$ is the state set

with $\nu(p) = \min\{\rho(p), \omega(p)\}$, representing the time to transition function

$\delta : S \times X^{\varnothing} \longrightarrow P$ is the transition function

where $X^{\varnothing} = \bar{X} \times (\ddot{X} \cup \{\varnothing\})$

and $\varnothing$ represents the null value (absence of value)

$\bar{\Lambda} : S \longrightarrow \bar{Y}$ is the continuous output function

$\lambda : P \longrightarrow \ddot{Y}$ is the partial discrete output function

HFSS models describe independent entities that can only communicate through input/output interfaces, defined by sets X and Y, respectively. These sets are structured into continuous and discrete parts since the corresponding HFSS components can accept and produce hybrid signals. The transition function, δ, describes how a component changes from the current p-state to the next one. Transitions can be triggered by different conditions. A component can change its p-state due the arrival of a discrete flow. A component can also change the p-state according to its autonomous behavior specified by the time-to-input/output functions ρ and τ. When the time elapsed in the current p-state reaches a value specified by one of these time functions, the component changes its p-state. A change can also be triggered by any combination of the previous conditions. The output function $\bar{\Lambda}$ describes the continuous flow output of a component, while the function λ describes the discrete flow. This latter function can only be non-null when the time elapsed in the current p-state reaches the time-to-output function τ.

We briefly describe the semantics of HFSS components, more details can be found in (Barros 2008). Figure 1 depicts the typical trajectories of a HFSS component. At time t_1 the component in p-state p_0 samples its input since its elapsed time reaches $\tau(p_0) = \rho(p_0) = e$. The component changes its p-state to $p_1 = \delta((p_0, \rho(p_0)), (x_1, \varnothing))$, where x_1 is the sampled value and no discrete flow is present.

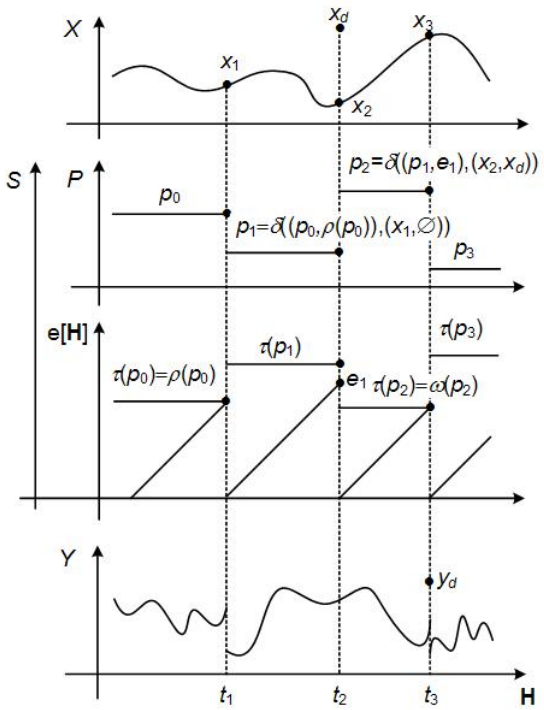

Figure 1: Basic HFSS component trajectories.

At time t_2 the discrete flow x_d is received by the component that changes to p-state $p_2 = \delta((p_1, e_1), (x_2, x_d))$, where x_2 is the continuous flow at t_2. At time t_3 the component reaches the time-to-output time limit and it changes to p-state $p_3 = \delta((p_2, \omega(p_2)), (x_3, \varnothing))$. At this time the discrete flow $y_d = \lambda(p_2)$ is produced. Additionally, component continuous output flow is always present and given by $\bar{\Lambda}(p, e)$. We provide next the HFSS definition of two basic models: Molecule Holder and Reactor that are used in this paper as the backbone for chemical reaction representation. These two models enable a hierarchical and modular definition of arbitrary networks of stochastic chemical equations.

Example I: Molecule Holder

The numerical solution of SCEs requires the computation of propensities, a value based of the number of molecules involved in each reaction. Modeling chemical reactions require thus the representation of molecule holders whose main task is to make available the number of molecules of each species. Since propensity changes with the number of molecules, holders need also to signal changes to all reactions depending on the corresponding species. A molecule holder is described by:

$$M_H = (X, Y, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda)$$

where

$X = \{\} \times \mathbf{N}_0^+$, where $\mathbf{N}_0^+$ is the set of positive integer numbers
$Y = \mathbf{N} \times \{\texttt{change}\}$

$P = \mathbf{R} \times \mathbf{N}_0^+$, where $\mathbf{R}$ is the set of real numbers
$\rho(\beta, n) = \infty$
$\omega(\beta, n) = \beta$
$\delta(((\beta, n), e), (\varnothing, \varnothing)) = (\infty, n)$
$\delta(((\beta, n), e), (\varnothing, n')) = (0, n + n')$
$\bar{\Lambda}((\beta, n), e) = n$
$\lambda(\beta, n) = \texttt{change}$

Holders receive as input the variation in the number of molecules. After receiving this value, the holder transmits the current number of molecules as a discrete flow signal. The number of molecules is kept as a continuous output flow so it can be sampled at arbitrary times by the involved reactions. Typical holder input and output trajectories are depicted in Figure 2, where molecules are added at times t_1 and t_2 and removed at time t_3.

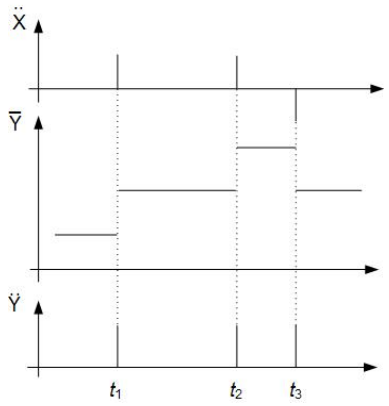

Figure 2: Holder discrete input flow and continuous/discrete output flows.

Example II: Reactor

A reactor is the other component required to model bio-molecular SCEs. Reaction time is computed based on the next reaction time (NRT) algorithm (Gibson and Bruck 2000). A reactor component samples the propensity and computes the reaction time. The reaction occurs then at the computed time if the propensity remains unchanged. Otherwise, a new reaction time is computed based on the current propensity and on the time elapsed since the last reaction instant. When the reaction occurs it issues a discrete flow `reaction` so the involved molecule holders can be updated. The reaction then samples the propensity to compute its next reaction time. A reactor is described by:

$$M_R = (X, Y, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda)$$

where

$X = \mathbf{R} \times \{\texttt{change}\}$
$Y = \{\} \times \{\texttt{reaction}\}$
$P = \mathbf{R} \times \mathbf{R} \times \mathbf{R}$
$\rho(\alpha, \beta, \mathit{propensity}) = \alpha$
$\omega(\alpha, \beta, \mathit{propensity}) = \beta$

$\delta(((\alpha,\beta,propensity),e),(p,\varnothing)) = (\infty, \text{EXPONENTIAL}(p), p)$
where EXPONENTIAL(p) is an exponential random variate with mean p
$\delta(((\alpha,\beta,propensity),e),(p,\texttt{change})) = (\alpha,(\beta-e)\cdot propensity/p,p)$
$\bar{\Lambda}((\alpha,\beta,propensity),e) = \varnothing$
$\lambda(\alpha,\beta,propensity) = \texttt{reaction}$

The direct method (DM) (Gillespie 1977), an alternative to NRT, is not amenable to a modular and hierarchical representation since it requires the overall knowledge of all propensities. On the contrary NRT is based exclusively on local information, enabling a distributed computation. We note that neither HFSS reactors nor HFSS molecule holders depend on the kind of chemical reaction they are actually involved in. This feature is based on the generalized sampling supported by HFSS that enables, in this case, the definition of independent models that can be used in arbitrary SCE networks.

2.2 HFSS Network Model

As mentioned before, HFSS network models are compositions of HFSS models (basic or other HFSS network models). Let $\widehat{N}$ be the set of names corresponding to HFSS network models, with $\widehat{N} \cap \widehat{B} = \{\}$. Formally, a HFSS network model associated with name $N \in \widehat{N}$ is defined by:

$$M_N = (X, Y, \eta)$$

where

N is the network name
$X = \bar{X} \times \ddot{X}$ is the set of network input flows
$\bar{X}$ is the set of network continuous input flows
$\ddot{X}$ is the set of network discrete input flows
$Y = \bar{Y} \times \ddot{Y}$ is the set of network output flows
$\bar{Y}$ is the set of network continuous output flows
$\ddot{Y}$ is the set of network discrete output flows
$\eta \in \widehat{\eta}$ is the name of the dynamic topology network executive
with
$\eta \in \widehat{\eta}$ representing the set of all names associated with HFSS executive models, constrained to $\widehat{\eta} \cap \widehat{B} = \widehat{\eta} \cap \widehat{N} = \{\}$

Executives are uniquely assigned to network models, i.e.,

$\forall_{i,j\in\widehat{N},i\neq j}\eta_i \neq \eta_j$ with $M_k = (X,Y,\eta)_k, \forall_{k\in\widehat{N}}$

The model of an executive $\eta \in \widehat{\eta}$, is a modified HFSS basic model, defined by:

$$M_\eta = (X, Y, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda, \widehat{\Sigma}, \gamma)_\eta$$

where

$\widehat{\Sigma}_\eta$ is the set of network topologies
$\gamma_\eta : P_\eta \longrightarrow \widehat{\Sigma}_\eta$ is the topology function

The network topology $\Sigma_\alpha \in \widehat{\Sigma}_\eta$, corresponding to the p-state $p_\alpha \in P_\eta$, is given by the 3-tuple

$$\Sigma_\alpha = \gamma(p_\alpha) \quad = (C_\alpha, \{I_{i,\alpha}\} \cup \{I_{\eta,\alpha}, I_{N,\alpha}\}, F_{i,\alpha} \cup \{F_{\eta,\alpha}, F_{N,\alpha}\})$$

where

C_α is the set of names associated with the executive state p_α
for all $i \in C_\alpha \cup \{\eta\}$
    $I_{i,\alpha}$ is the sequence of asynchronous influencers of i
    $F_{i,\alpha}$ is the input function of i
$I_{N,\alpha}$ is the sequence of network influencers
$F_{N,\alpha}$ is the network output function

For all $i \in C_\alpha$

$M_i = (X, Y, P, \rho, \omega, \delta, \bar{\Lambda}, \lambda)_i$ if $i \in \widehat{B}$
$M_i = (X, Y, \eta)_i$ if $i \in \widehat{N}$

Variables are subjected to the following constraints for every $p_\alpha \in P_\alpha$

$N \notin C_\alpha, N \notin I_{N,\alpha}, \eta \notin C_\alpha$
$N \notin E_{i,\alpha}$ for all $i \in C_\alpha \cup \{\eta, N\}$
$F_{N,\alpha} : \times_{k \in I_{N,\alpha}} Y_k \longrightarrow Y^\varnothing$
$F_{i,\alpha} : \times_{k \in I_{i,\alpha}} V_k \longrightarrow X_i^\varnothing$
where

$$V_k = \begin{cases} Y_k^\varnothing & \text{if } k \neq N \\ X^\varnothing & \text{if } k = N \end{cases}$$

$F_{N,\alpha}((\bar{v}_{k_1}, \varnothing), (\bar{v}_{k_2}, \varnothing), ...) = (\bar{y}_N, \varnothing)$
$F_{i,\alpha}((\bar{v}_{k_1}, \varnothing), (\bar{v}_{k_2}, \varnothing), ...) = (\bar{x}_i, \varnothing)$

These two last constraints are characteristic of discrete systems and impose that non-null discrete flow values cannot be created from a sequence composed exclusively by null values.

The executive is a special component that controls the network topology. Topology depends on the current executive p-state and it is established by the *topology function* γ. Changes in network topology include the ability to modify composition and coupling through add and delete operations. Although HFSS relies on a central component to manage the topology of each network, the decision to change this topology can be made by any arbitrary component or in cooperative manner by several components. However, this decision needs to be communicated to the executive so it can become effective. HFSS topology management guarantees well defined and deterministic behavior when changes in topology occur (Barros 2008).

Example III: Reaction Network

Molecules and reactors described in the last section are the basic elements to define stochastic chemical systems. We proceed now to the description of a simple system of bio-molecular SCEs (Paulsson et al. 2000). This system involves three kinds of molecules I, S and P whose interactions are described in Table 1 where i, s and p represent the number of molecules of product I, S and P, respectively.

The network executive has a single p-state p_0 associated with the topology

$$\Sigma_0 = \gamma(p_0) = (C_0, \{I_{i,0}\}, \{F_{i,0}\})$$

Table 1: SCE network.

Id	Equation	Propensity
$R1$	$\emptyset \xrightarrow{r_1} I$	r_1
$R2$	$I,S \xrightarrow{r_2} S$	$r_2 \cdot i \cdot s$
$R3$	$I \xrightarrow{r_3} P$	$r_3 \cdot i$
$R4$	$P \xrightarrow{r_4} \emptyset$	$r_4 \cdot p$
$R5$	$\emptyset \xrightarrow{r_5} S$	r_5
$R6$	$S \xrightarrow{r_6} \emptyset$	$r_6 \cdot s$

where

$$
\begin{aligned}
&C_0 = \{R1, R2, R3, R4, R5, R6, I, S, P\} \\
&I_{I,0} = \{R1, R3\}, I_{S,0} = \{R5, R6\}, I_{P,0} = \{R2\} \\
&I_{R1,0} = I_{R5,0} = \{\}, I_{R2,0} = \{I\}, I_{R3,0} = \{P\} \\
&I_{R4,0} = \{I, S\}, I_{R6,0} = \{S\} \\
&F_{R1,0}() = (r_1, \varnothing) \\
&F_{R2,0}((i_c, i_d), (s_c, s_d)) = (r_2 \cdot i_c \cdot s_c, m_d(i_d, s_d)) \\
&F_{R3,0}(i_c, i_d) = (r_3 \cdot i_c, i_d) \\
&F_{R4,0}(p_c, p_d) = (r_4 \cdot p_c, i_d) \\
&F_{R5,0}() = (r_5, \varnothing) \\
&F_{R6,0}(s_c, s_d) = (r_6 \cdot s_c, s_d) \\
&F_{I,0}((R1_c, R1_d), (R2_c, R2_d), (R3_c, R3_d)) = (\varnothing, n_I) \\
&F_{S,0}((R5_c, R5_d), (R6_c, R6_d)) = (\varnothing, n_S) \\
&F_{P,0}((R3_c, R3_d), (R4_c, R4_d)) = (\varnothing, n_P)
\end{aligned}
$$

with

$$
m_d(v_1, \ldots, v_n) = \begin{cases} \varnothing & \text{if } v_1 = \ldots = v_n = \varnothing \\ \texttt{change} & \text{otherwise} \end{cases}
$$

and

$$
n_I = \begin{cases} 1 & \text{if } R1_d = \texttt{change} \\ -1 & \text{if } R2_d = \texttt{change} \vee R3_d = \texttt{change} \\ \varnothing & \text{otherwise} \end{cases}
$$

$$
n_P = \begin{cases} 1 & \text{if } R3_d = \texttt{change} \\ -1 & \text{if } R4_d = \texttt{change} \\ \varnothing & \text{otherwise} \end{cases}
$$

$$
n_S = \begin{cases} 1 & \text{if } R5_d = \texttt{change} \\ -1 & \text{if } R6_d = \texttt{change} \\ \varnothing & \text{otherwise} \end{cases}
$$

In the network topology we have omitted executive and network influencers and input/output functions. The topology defines reaction propensity and reaction rates $r_i, i = 1..6$. This definition, enabled by HFSS generalized sampling, supports model reuse since both reactors and molecule holders defined in the last

section are completely independent from the context they are inserted. These two models can thus be reused with no modification to define any arbitrary SCE network.

A graphical representation of the SCE network is depicted in Figure 3, where components are represented by a rectangle, functions by circles, and the set of influencers of component A is represented by the components having links to the input function of component A. We note that there is a direct mapping between the reactions R1, ..., R6, and the network topology. This direct representation make HFSS models of SCEs very easy to develop and to understand.

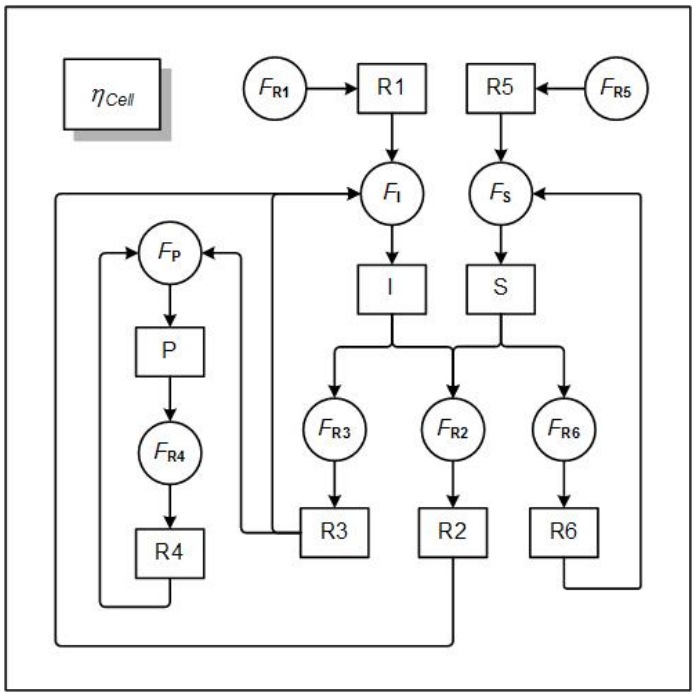

Figure 3: SCE network corresponding to chemical reactions defined in Table 1.

3 PRELIMINARY SIMULATION RESULTS

HFSS is currently implemented in JUSE, a language supporting Pluggable software units (Barros 2011). We have developed a library of basic components for describing bio-molecular systems in JUSE. This library can support the modeling and simulation of arbitrary SCE networks through the PUs `Holder` and `Reactor` whose HFSS models were defined in previous Sections. Figure 4 represents the partial network topology defined in Listing 1.

Molecule holders and reactors are modeled by JUSE pluggable software units (PUs) that have gates as the only mechanism to communicate. The `Holder` PU issues a signal through output gate `change` when its content is modified. Input gates `add` and `remove` add/remove molecules to/from the `Holder`. The `Reactor` PU sends a signal through output gate `reaction` when a reaction occurs. Propensity is obtained through output gate `propensity`. In Figure 4, `R6` gets the propensity from `S`. When the reaction occurs `R6` removes a molecule from `S`. `R5` is a creation reaction and it does not depend on any molecule. When the corresponding reaction occurs it adds a molecule to `S`. This behavior closely follows the equations defined in Table 1, being very easy to define and to verify.

JUSE definition of the overall network is given in Listing 1. To help defining the chemical reactions we have created method `defineReaction(left, name, propensity, remove, add)`. The first parameter corresponds to the required molecules to enable the chemical reaction. The second parameter is the reaction name. The reaction propensity is given next and it is given by a closure whose parameters

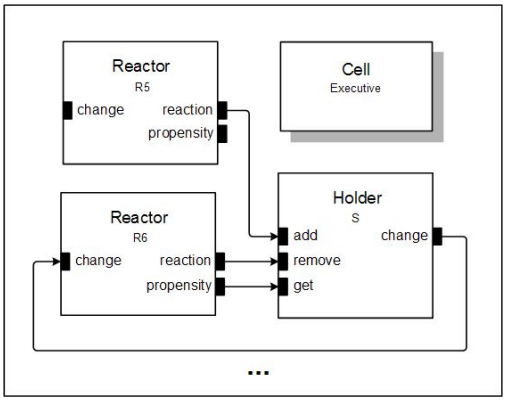

Figure 4: JUSE partial representation of the SCE network of Table 1.

correspond to the molecules defined by the `left` parameter. The next parameter is the list of the molecules removed by the reaction. Finally, the last parameter defines the molecules created by the reaction.

```
public class Cell extends Executive {                                                                    1
    //...                                                                                                2
   addS(Holder, 'I', {Holder m -> m.set(mi)});     //initial number of type I molecules                  3
   addS(Holder, 'S', {Holder m -> m.set(ms)});     //initial number of type S molecules                  4
   addS(Holder, 'P', {Holder m -> m.set(mp)});     //initial number of type P molecules                  5
   addS(Reactor, 'R1');                                                                                  6
   addS(Reactor, 'R2');                                                                                  7
   addS(Reactor, 'R3');                                                                                  8
   addS(Reactor, 'R4');                                                                                  9
   addS(Reactor, 'R5');                                                                                  10
   addS(Reactor, 'R6');                                                                                  11
   defineReaction([], 'R1', '{r1}', [], [1, 'I']);                                                       12
   defineReaction([1, 'I', 1, 'S'], 'R2', '{Integer i, Integer s -> r2 * i * s}', [1, 'I'], []);        13
   defineReaction([1, 'I'], 'R3', '{Integer i -> r3 * i}', [1, 'I'], [1, 'P']);                          14
   defineReaction([1, 'P'], 'R4', '{Integer p -> r4 * p}', [1, 'P'], []);                                15
   defineReaction([], 'R5', '{r5}', [], [1, 'S']);                                                       16
   defineReaction([1, 'S'], 'R6',  '{Integer s -> r6 * s}', [1, 'S'], []);                               17
    //...                                                                                                18
}                                                                                                        19
```

Listing 1: Cell definition in JUSE.

Listing 1 defines the set of SCEs in a *declarative* form enabling a simple and intuitive creation of chemical networks.

PUs `Holder` and `Reactor` can be (re)used to define any SCE network. Domain dependent information is declarative and it is expressed through the propensity function and the links between components.

Preliminary simulation results are presented in Figure 5 that displays the plots of variables I, S and P for a single simulation run.

JUSE supports HFSS capabilities for supporting changes in topology. Given the HFSS ability to structure systems into compartments it becomes possible to model complex hierarchical bio-molecular systems. Future work will also address the automatic translation of Systems Biology Markup Language (SBML) into JUSE. We plan also to exploit HFSS and JUSE as a basis for a graphical formalism to describe biological systems (Faeder 2011).

4 RELATED WORK

Hierarchical and modular principles have been used as a powerful heuristic for handling complex problems in many fields. One of the first formal descriptions of modular decomposition have been made in the area of

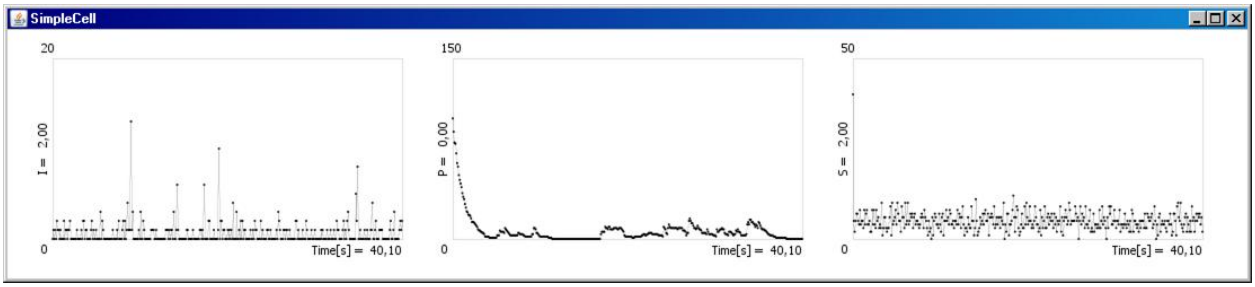

Figure 5: Simulation results.

General Systems Theory (Mesarovic and Takahara 1975), (Wymore 1967). An earlier use of modularly in software was made in (Kahn 1974), where a synchronous programming language was defined. The modular description of discrete event systems was first developed in (Zeigler 1976). The concept of generalized sampling was introduced in (Barros 2002). The combination of generalized-sampling and discrete events was achieved in HFSS (Barros 2003). The concept of dynamic topology general system network was developed in (Barros 1997).

Modeling and simulation of biological systems is a recent area that has been subject of intense research (Curti, Degano, and Baldari 2003), (Kleijn and Koutny 2009), (Mao and Resat 2004), (Phillips 2009), (Priami and Quaglia 2005), (Regev et al. 2004). Most of these approaches are not based upon a general purpose modeling and simulation formalisms but they develop *ad-hoc* solutions to solve SCEs, instead. The Direct Method (DM) (Gillespie 1977) is often used to simulate SCEs. This choice seems to be imposed by the lack of an explicit topology support that makes it difficult to use other algorithms like, for example, the Next Reaction Time (NRT) (Gibson and Bruck 2000). A major drawback of the DM is that it is not amenable to provide a hierarchical and modular representations of SCEs, similar to that enabled by HFSS. The DM requires a global knowledge of all chemical equations imposing a centralized engine responsible to choose the next equation and the corresponding reaction time. This kind of algorithm is not easily represented in a distributed and hierarchical & modular formalism like HFSS. Although the DM could be represented in HFSS, it would impose a flat (non-hierarchical) model with all the equations.

The most common formalisms to represent SCEs are based on the π-calculus (Priami and Quaglia 2005), (Phillips 2009), (Versari 2007). Other process algebras formalisms have also been developed like, for example, Bio-PEPA (Ciocchetta and Hillston 2009), and sCCP (Bortolussi and Policriti 2008). However, these approaches provide mainly support for the DM that tends to blend models and simulation algorithms. We found no evidence that the representation of SCEs based on reactors and molecule holders models, as defined in Section 2.1, can be achieved using these process algebras. We also found that the analysed process algebras make no provisions for representing dynamic topologies in general and component mobility in particular, as supported by HFSS (Barros 2005).

We have focused on the representation of chemical reactions using SCSs. Other types of models based, for example, on ODEs (ordinary differential equations), have been developed (Galpin, Bortolussi, and Hillston 2009). However, these models provide only a good approximation when a large number of molecules is involved which is not often the case when describing cellular models. The HFSS formalism can also describe numerical solvers to represent systems of ODEs (Barros 2003).

The HFSS formalism offers a unique SCEs representation by enabling chemical equations to be described by only two kinds of models: molecules holders and reactors as shown in this paper. Since HFSS is a modeling and simulation formalism it offers full support for time advance mechanisms. HFSS also guarantees a deterministic execution of simulation models and the closure under the composition operation. The formalism is also able to represent a plethora of models, including, for example, digital controllers and ODEs (Barros 2003). Given the explicit and dynamic nature of HFSS network topologies, the formalism can seamlessly support the representation of chemical reactions in dynamic compartments. These features enable an intuitive and and expressive representation of large and complex SCEs networks, necessary to

model cells and their organization. Other approaches require the use of explicit components to define timed behavior, and they seem not to be amenable to support the definition of general purpose components to support SCEs (Kuttler and Niehren 2006). The support for declarative definitions of SCE networks seems also limited in these approaches (Kuttler and Niehren 2006).

HFSS explicit topology representation makes it trivial to use the Next Reaction Method (Gibson and Bruck 2000), a very efficient algorithm to solve SCEs (Mauch and Stalzer 2011). HFSS provides also an intuitive representation for SCEs enabling a easy development of SCEs networks.

The validation of HFSS models, in particular network models requiring changes in topology, will benefit from the case studies reported in the literature (Uhrmacher et al. 2011).

5 CONCLUSIONS AND FUTURE WORK

The simulation of biological systems is a challenge to modeling methodologies. We have shown that HFSS offers key features that are fundamental to represent such systems. HFSS enables a modular description of biological systems enabling a seamless representation of chemical reaction in compartments. This modular partitioning can be directly mapped into the organization of living entities into cells. The hierarchical nature of HFSS enables also the division of cells into other compartments, like the nucleus, enabling the direct mapping of cell organization. Living entities poses another challenge to modeling methodologies by exhibiting a dynamic creation and destruction of cells. HFSS provides support for arbitrary dynamic topologies, including model mobility, (Barros 2005), and we plan to exploit these capabilities to represent cell division, cell dead and virus attack. Rule-based topology transformation (Michel, Spicher, and Giavitto 2008), will be considered as an extension to the HFSS formalism. Future work will address the development of libraries to represent actual molecular pathways that can support accurate cell modeling.

ACKNOWLEDGEMENTS

This work was partially supported by the Portuguese Foundation for Science and Technology under project PTDC/EIA-EIA/100752/2008.

REFERENCES

Barros, F. 1997. “Modeling Formalisms for Dynamic Structure Systems”. *ACM Transactions on Modeling and Computer Simulation* 7 (12): 505–515.

Barros, F. 2002. “Towards a Theory of Continuous Flow Models”. *International Journal of General Systems* 31 (1): 29–39.

Barros, F. 2003. “Dynamic Structure Multiparadigm Modeling and Simulation”. *ACM Transactions on Modeling and Computer Simulation* 13 (3): 259–275.

Barros, F. 2005. “A Formal Representation of Mobile Hybrid Components”. *SIMULATION: Transactions on the SCS* 13 (3): 259–275.

Barros, F. 2008. “Semantics of Discrete Event Systems”. In *Distributed Event-Based Systems*, 252–258.

Barros, F. 2011. *Achieving Reuse with Pluggable Software Units*, 183–191. Number 6727 in LNCS. Springer.

Bortolussi, L., and A. Policriti. 2008. “Modeling Biological Systems in Stochastic Concurrent Constraint Programming”. *Constraints* 13 (1): 66–90.

Ciocchetta, F., and J. Hillston. 2009. “Bio-PEPA: A framework for the modelling and analysis of biological systems”. *Theoretical Computer Science* 410 (33–34): 3065–3084.

Curti, M., P. Degano, and C. Baldari. 2003. “Causal π-Calculus for Biochemical Modelling”. In *Computational Methods in Systems Biology*, Volume LNCS 2602, 21–34.

Faeder, J. 2011. “Toward a Comprehensive Language for Biological Systems”. *BMC Biology* 9 (68).

Galpin, V., L. Bortolussi, and J. Hillston. 2009. “HYPE: A Process Algebra for Compositional Flows and Emergent Behaviour”. In *Proceedings of the 20th International Conference on Concurrency Theory*, 305–320.

Gibson, A., and J. Bruck. 2000. "Efficient Exact Stochastic Simulation of Chemical Systems with many Species and many Channels". *Journal of Physical Chemistry* 104:1876–1889.

Gillespie, D. 1977. "Exact Stochastic Simulation of Coupled Chemical Reactions". *Journal of Physical Chemistry* 25:2340–2361.

Goldblatt, R. 1998. *Lectures on the Hyperreals: An Introduction to Nonstandard Analysis*. New York: Springer.

Kahn, G. 1974. "The Semantics of a Simple Language for Parallel Programming". In *IFIP Congress*, 471–475.

Kleijn, J., and M. Koutny. 2009. "A Petri Net Model for Membrane Systems with Dynamic Structure". *Natural Computation* 8:781–796.

Kuttler, C., and J. Niehren. 2006. "Gene Regulation in the Pi Calculus: Simulating Cooperativity in the Lambda Switch". *Transactions on Computational Systems Biology vol. 4230, VII*:24–55.

Mao, L., and H. Resat. 2004. "Probabilistic Representation of Gene Regulatory Networks". *Bioinformatics* 20 (14): 2258–2269.

Mauch, S., and M. Stalzer. 2011. "Efficient Formulations for Exact Stochastic Simulation of Chemical Systems". *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 8 (1): 27–35.

Mesarovic, M., and Y. Takahara. 1975. *General Systems Theory: A Mathematical Foundation*. Academic Press.

Michel, O., A. Spicher, and J.-L. Giavitto. 2008. "Rule-Based Programming for Integrative Biological Modeling". *Natural Computing* 8 (4): 865–889.

Paulsson, J., O. Berg, and M. Ehrenberg. 2000. "Stochastic Focusing: Fluctuation-Enhanced Sensitivity of Intracellular Regulation". *Proceedings of the National Academy of Sciences* 97 (13): 7148–7153.

Phillips, A. 2009. "An Abstract Machine for the Stochastic Bioambient Calculus". *Electronic Notes on Theoretical Computer Science* 227:143–159.

Priami, C., and P. Quaglia. 2005. "Beta Binders for Biological Interaction". In *Computational Methods in Systems Biology*, Volume LNBI 3082, 20–33.

Regev, A., E. Panina, W. Silverman, L. Cardelli, and E. Shapiro. 2004. "BioAmbients: An Abstraction for Biological Compartments". *Theoretical Computer Science* 325:141–167.

Uhrmacher, A., J. Himmelspach, and R. Ewald. 2011. *Discrete-Event Modeling and Simulation: Theory and Applications*, Chapter Effective and Efficient Modeling and Simulation with DEVS Variants, 139–176. CRC Press.

Versari, C. 2007. "Encoding Catalytic P Systems in π@". *Electronic Notes in Theoretical Computer Science* 171 (2): 171–186.

Wymore, A. 1967. *A Mathematical Theory of Systems Engineering: The Elements*. Krieger.

Zeigler, B. 1976. *Theory of Modeling and Simulation*. Kluwer.

AUTHOR BIOGRAPHY

Fernando J. Barros is an assistant professor at the University of Coimbra. His research interests include Theory of Modeling and Simulation, Self-Adaptive Systems and Software Reuse. Fernando Barros has organized several conference and workshops on Modeling and Simulation and he has served in the editorial board of SIMULATION and the International Journal of Simulation & Process Modelling. He has published more than 60 scientific papers. Fernando Barros has received the best paper award in the Summer Computer Simulation Conference 2011.