Available online at www.sciencedirect.com
ScienceDirect
Electronic Notes in Theoretical Computer Science
ELSEVIER
Electronic Notes in Theoretical Computer Science 194 (2008) 165–180
www.elsevier.com/locate/entcs

Efficient Stochastic Simulation of Biological Systems with Multiple Variable Volumes

Cristian Versari [1] Nadia Busi[2]

Dipartimento di Scienze dell'Informazione
Università di Bologna
Bologna, Italy

Abstract

The application of concurrent calculi to the formalisation of biological systems constitutes a promising approach to the analysis in silico of biological phenomena. The Gillespie algorithm is one of the main models exploited for their stochastic simulation. While the original algorithm considers only one fixed-volume compartment, the simulation of biological systems often requires multi-compartment semantics. In this paper we present an enhanced formulation of an extended version of the algorithm which handles multiple compartments with varying volumes. The presented algorithm is used as basis for the implementation of an extension of the stochastic π-Calculus, called Sπ@, which allows an intuitive and concise formalisation of such systems. The algorithm is also efficient in presence of a high number of compartments and reactions, therefore Sπ@ represents the starting point for the development of an effective tool for the simulation of biological systems with dynamical structure even in presence of computationally expensive phenomena like diffusion.

Keywords: stochastic simulation, π-Calculus, process calculi

1 Introduction

The application of concurrency theory to Systems Biology represents a recent and promising approach to the modelling, simulation and analysis *in silico* of biological systems. After the first application of the π-Calculus [15,14,16,18] to the formalisation and simulation of biochemical systems [22,20] the interest of the research community has focused on the development of calculi (e.g. [21,3,19,5,11]) which aim at modelling more faithfully the biological reality of interest.

The typical structure of biological systems suggests that their effective modelling requires the introduction of multiple compartments with dynamical structure, as denoted by many of the cited approaches. The Gillespie stochastic simulation algorithm [9,10] (SSA for short) constitutes one of the more exploited chemical

[1] Email: versari@cs.unibo.it
[2] Email: busi@cs.unibo.it

doi:10.1016/j.entcs.2007.12.012

abstractions for the effective simulation of bio-systems expressed in terms of concurrent calculi, but its original formulation is limited to systems composed of a single, fixed-size volume. In order to overcome these limits we introduced in [24] an extended version of the SSA which handles multiple, variable volumes (multi-compartmental SSA, MSSA for short) together with an extended version of the stochastic π-Calculus, called Sπ@. The Sπ@ language allows to formalise straightforwardly biological systems with dynamical compartment structure which can be then simulated by means of the MSSA.

Unfortunately, the MSSA inherits the computational complexity of the original SSA with the additional parameter of the number of compartments. This causes any simulation to be unfeasible when the number of compartments grows significantly, as noted in [7]. Several improvements of the SSA have already been proposed [8,1,7] but none of them can be applied to the MSSA with appreciable gain because of its unique features of considering multiple, stochastically varying volumes.

In this paper we present an enhanced version of the MSSA (EMSSA) whose complexity scales logarithmically (instead of linearly) both with the number of reactions and with the number of compartments of the system. The EMSSA constitutes then a valid implementation for the exact stochastic simulation of biological systems in presence of dynamical structure and varying volumes which effectively handles phenomena like osmosis [24], diffusion [7] and cellular growth and division [13].

1.1 Structure of the Paper

The paper is organised as follows. In Sect. 2 the preliminary notions concerning the original SSA, the MSSA and the Sπ@ calculus are introduced. In Sect. 3 the enhanced MSSA is formulated together with some suggestions for further improvements. In Sect. 4 some conclusive remarks are reported.

2 Preliminaries

2.1 Gillespie Stochastic Simulation Algorithm

The original SSA considers a fixed volume V containing a mixture of N chemical species $S_1, \dots S_N$ interacting through M reaction channels $R_1, \dots, R_M$. $X_1, \dots, X_N$ represent respectively the number of molecules for each of the S_i chemical species. The state of the system at any time t is characterised by the state vector $X(t) = \big(X_1(t), \dots, X_N(t)\big) = x$ where $X_i(t)$ is the number of molecules of species S_i at the given time. $\nu_\mu = (\nu_{1\mu}, \dots, \nu_{N\mu})$ is the *state change vector* , which contains all the informations on the number and species of reactant molecules and reaction products for each reaction R_μ. For a reaction of the kind

$$R_\mu : \quad S_i + S_j \to S_k \qquad\qquad i \neq j$$

we have that $\nu_{i\mu} = \nu_{j\mu} = -1$, while $\nu_{k\mu} = +1$ and $\nu_{l\mu} = 0$ for every other index l. The probability that an R_μ reaction will occur inside V in the next infinitesimal

time interval $(t, t + \mathrm{d}t)$ is calculated as

$$a_\mu(x)\,\mathrm{d}t = X_i(t)X_j(t)c_\mu\,\mathrm{d}t$$

where $c_\mu\,\mathrm{d}t$ is the probability that a particular molecular pair of the involved chemical species will react according to R_μ inside V in the next infinitesimal interval $(t, t+\mathrm{d}t)$ and $a_\mu(x)$ is the *propensity function* of R_μ. In general, $a_\mu(x)$ is calculated as

$$a_\mu(x) = h_\mu(x)c_\mu \tag{1}$$

where $h_\mu(x)$ represents the number of distinct R_μ molecular reactant combinations available at some time t inside V.

The dynamics of the system obeys the *chemical master equation* (CME) [9,10,1]

$$\begin{aligned}&\delta P(x,t|x_0,t_0)/\delta t = \\ &\quad\textstyle\sum_{\mu=1}^{M}[a_\mu(x-\nu_\mu)P(x-\nu_\mu,t|x_0,t_0) - a_\mu(x)P(x,t|x_0,t_0)]\end{aligned} \tag{2}$$

where $P(x,t|x_0,t_0)$ is the probability that $X(t)$ will be x, given that $X(t_0) = x_0$. The CME is hard to solve except for very simple systems. The SSA provides a stochastic simulation method rigorously equivalent to the CME. Starting from an initial state, the SSA allows the system to evolve stochastically by providing the next state reached after a single firing of one of the M molecular reactions, chosen according to the CME. The aim of the algorithm is to find *when the next reaction fires* (i.e. the value of the time variable τ) and *which of the M reactions is* (the index μ of the next reaction R_μ).

The function $P(\tau,\mu|x)\,\mathrm{d}\tau$ represents the probability that, given the state x at some time t, the next reaction in V will occur in the infinitesimal time interval $(t+\tau, t+\tau+\mathrm{d}\tau)$ and will be an R_μ reaction. It can be calculated as the product of the probability $P(\tau|x)$ that, given the state x at some time t, no reaction will occur in the time interval $(t, t+\tau)$, times the probability $a_\mu(x)\,\mathrm{d}\tau$ that an R_μ reaction will occur in the time interval $(t+\tau, t+\tau+\mathrm{d}\tau)$:

$$P(\tau',\mu'|x')\,\mathrm{d}\tau' = P(\tau'|x')a_\mu(x')\,\mathrm{d}\tau'$$

Since $[1 - \sum_j a_j(x')\,\mathrm{d}\tau']$ is the probability that no reaction will occur in time $\mathrm{d}\tau'$ from the state x',

$$P(\tau' + \mathrm{d}\tau'|x') = P(\tau'|x') \cdot [1 - \sum_j a_j(x')\,\mathrm{d}\tau']$$

from which

$$P(\tau|x) = \exp(-\sum_{j=1}^{M} a_j(x)\tau)$$

Hence, the function $P(\tau, \mu|x)$ is given by

$$P(\tau,\mu|x) = \begin{cases} a_\mu(x)\exp(-a_0(x)\tau) & 0 \le \tau < +\infty, \\ & \mu = 1,\dots,M \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $a_0(x) = \sum_{j=1}^{M} a_j(x)$.

In order to generate a random pair (τ, μ) according to Expr. (3) by a pair (z_1, z_2) obtained by a unit-interval uniform random number generator, the following resampling is evaluated in the SSA for τ

$$\tau = \frac{1}{a_0(x)} \log \frac{1}{z_1} \tag{4}$$

while μ is calculated as the smallest integer satisfying

$$\sum_{j=1}^{\mu} a_j(x) > z_2 a_0(x) \tag{5}$$

The SSA can be summarised as follows:

Algorithm 1 *(Gillespie Stochastic Simulation Algorithm)*

(i) *calculate* $a_i, i = 1,\dots,M$ *and* a_0 *from Expr.* (1)*;*

(ii) *generate uniformly two random numbers* z_1, z_2 *in the interval* $(0, 1)$*;*

(iii) *calculate* τ *from Expr.* (4) *and* μ *from Expr.* (5)*;*

(iv) *update the states of the species to reflect the execution of reaction* μ *and set* $t = t + \tau$*;*

(v) *go to step* (i)*.*

For details we refer to [10].

2.2 Stochastic Simulation with Multiple Compartments

In [24] we introduced a simple extension of the original SSA which allows to handle multiple volumes with variable sizes and still reflects the original CME. This can be achieved by expressing the propensity function a_j of each reaction in function of the volume V of the compartment. In the case of reactions fired by two reactant molecules, we have:

$$a_\mu(x) = h_\mu(x)c_\mu = V^{-1}(x)h_\mu(x)r_\mu \tag{6}$$

The $r_\mu\,\mathrm{d}t$ value represent the probability that an R_μ reaction fires in the infinitesimal time $\mathrm{d}t$ inside a unit-size volume containing a single molecule pair undergoing reaction R_μ.

The volume $V(x)$ of the compartment at some time t in the state $x = X(t) = \big(X_1(t), \ldots, X_N(t)\big)$ is calculated as the sum of the volumes occupied by each of the molecules located inside V. Given the function $v : \{S_1, \ldots, S_M\} \to \mathbb{R}$ which returns the volume occupied by one molecule of each chemical species, V is calculated as

$$V(x) = \sum_{j=1}^{M} v(S_j) X_j(t) \tag{7}$$

In the case of aeriform systems or systems composed of one (or few) chemical species, the function $v(S_j)$ can be easily estimated by knowing the molecular weight of the species and their density. In the case of real systems composed of thousands of different species, $v(S_j)$ may only be estimated by formulating a specific kinetic model.

In presence of more than one compartment, each of the R_j reactions must be considered w.r.t. the compartment the reaction occurs in. This means that each R_μ reaction is characterised by C different propensity functions, one for each of the C compartments. In the case of reactions fired by two reactant molecules

$$a_\mu^k(x) = h_\mu^k(x) c_\mu^k = V_k^{-1}(x) h_\mu^k(x) r_\mu \qquad k = 1, \ldots, C \tag{8}$$

where

$$x = X(t) = \big(\tilde{X}_1(t), \ldots, \tilde{X}_C(t)\big), \quad \tilde{X}_k(t) = \big(X_1^k(t), \ldots, X_N^k(t)\big)$$

Each $X_j^k(t)$ represents the number of molecules of the S_j chemical species inside compartment k at time t. The value $a_\mu^k(x)\,\mathrm{d}t$ represents the probability that the reaction R_μ will happen inside compartment k in the next infinitesimal interval $\mathrm{d}t$. Each volume V_k is calculated as

$$V_k(x) = \sum_{j=1}^{M} v(S_j) X_j^k(t) \tag{9}$$

The value a_0 of Expr. (3) becomes

$$a_0(x) = \sum_{k=1}^{C} \sum_{j=1}^{M} a_j^k(x) = \sum_{k=1}^{C} a^k(x) \tag{10}$$

where $\sum_{j=1}^{M} a_j^k(x) = a^k(x)$. Expr. (5) is then unchanged:

$$\tau = \frac{1}{a_0(x)} \log \frac{1}{z_1} \tag{11}$$

where a_0 is calculated according to Expr. (10).

In order to identify both the compartment ψ and the reaction μ by a single generation of a unit-interval random number z_2, Expr. (5) is modified so that

(ψ, μ) is the smallest pair of indexes satisfying

$$\sum_{k=1}^{\psi} \sum_{j=1}^{\mu} a_j^k(x) > z_2 a_0(x) \tag{12}$$

where $(\psi, \mu) < (\psi', \mu') \iff f(\psi, \mu) < f(\psi', \mu')$, with $f(\psi, \mu) = (\psi * (M + 1) + \mu)$.

The multi-compartmental simulation algorithm (MSSA) can be finally expressed as a slight variation of the SSA:

Algorithm 2 *(Multi-compartmental Stochastic Simulation Algorithm)*

(i) *calculate* a_i^k *with* $k = 1, \ldots, C, \quad i = 1, \ldots, M$ *from Expr.* (8), (9) *and* a_0 *from Expr.* (10)*;*

(ii) *generate uniformly two random numbers* z_1, z_2 *in the interval* $(0, 1)$*;*

(iii) *calculate* τ *from Expr.* (11) *and* (ψ, μ) *from Expr.* (12)*;*

(iv) *update the states of the species to reflect the execution of reaction* (ψ, μ) *and set* $t = t + \tau$*;*

(v) *go to step* (i)*.*

As discussed in [24], the original Gillespie's propensity functions are recovered when the system is composed of a single compartment of unitary volume: this may be achieved by setting $v(S_j) = 0 \ \forall j$ and adding a fictitious element of volume 1 not participating in any reaction. Although immediate, this expedient seems quite artificial. A more faithful modelling would be obtained by specifying the total (not null) volume of the products of each reaction equal to the total volume of the respective reactants. Further elements not taking part in the reactions but influencing their rates (such as water, which may dilute reactants and slow down reactions even without direct chemical interaction) shall also be specified, because they actually determine the total volume of the compartment.

2.3 The Stochastic π@ calculus

The MSSA was introduced in [24] for the simulation of processes expressed in the Stochastic π@ calculus (Sπ@ for short), which allows to describe formally biological systems with dynamical compartment structure and variable volumes. The expressiveness of Sπ@ is comparable [23] to compartmentalised languages like Bioambients [21] and Brane Calculi [3].

The π@ language [23] is a conservative extension of the π-Calculus [15,16,14] with polyadic synchronisation for encoding compartments and different levels of priority for gaining atomicity [2,4,11]. The Sπ@ language can be considered in first approximation as the stochastic version of a core π@ limited to two levels of priority and two names for each channel. The capability of giving infinite rates to reactions replaces the two priority levels of this core π@, while the two names denoting each action assume different meaning, since the first represents the type of (chemical) reaction, while the second the compartment where the reaction takes place.

The syntax and semantics of Sπ@ follows.

Definition 2.1 Let $\mathcal{N}, \mathcal{C}$ be distinct sets of names on finite alphabet, with m, n ranging over $\mathcal{N}$, a, b over $\mathcal{C}$ and x, y over $\mathcal{X} = \mathcal{N} \cup \mathcal{C}$. Let also v range over $\mathbb{R}$ within the interval $[0, +\infty[$. The syntax of the Sπ@ language is defined as

$$\begin{aligned} P &::= \mathbf{0} \mid \sum_{i \in I} \pi_i.P_i \mid P \mid Q \mid !\pi.P \mid (\nu\ x)P \\ \pi &::= n@a : v(\tilde{x}) \mid \overline{n}@a : v\langle \tilde{x} \rangle \end{aligned}$$

where $\tilde{x}$ represents zero or more names $x_1, \ldots, x_i$ ranging over $\mathcal{X}$.

$\mathbf{0}$ is the null process, capable of doing nothing. $\sum_{i \in I} \pi_i.P_i$, written also $\pi_1.P_1 + \pi_2.P_2$ in the case $|I| = 2$, represents the guarded choice between different actions. $P \mid Q$ means that P and Q are two processes executing in parallel. $!\pi.P$ is the guarded replication. $(\nu\ x)P$ allows the scope restriction of the name x: the restriction of compartment names allows the creation of new compartments, while the restriction of reaction names is used in several ways, like for representing bindings between different elements. The expressions $n@a : v(\tilde{x})$ and $\overline{n}@a : v\langle \tilde{x} \rangle$ represent respectively the polyadic input and output capabilities of a process, where

- n is the kind of reaction the process is ready to perform: in Expr. (8) it corresponds to the index μ denoting the reaction R_μ;
- a is the compartment where the reaction may take place, corresponding to k in Expr. (8);
- v corresponds to $v(S_\mu)$ in Expr. (9) and represents the volume occupied inside compartment a by the process ready to perform the input or output action.

Sπ@ syntax allows to easily specify processes which are located (and hence may occupy volume) in more than one compartment. For example, the process P

$$P \equiv n@a : v_1.Q_1 + m@b : v_2.Q_2 + p@a : v_3.Q_3$$

occupies some space both in compartment a and in compartment b. Anyway, since Sπ@ syntax does not allow to associate a unique volume value with each action name, P may be written as well as

$$P \equiv n@a : v_{13}.Q_1 + m@b : v_2.Q_2 + p@a : 0.Q_3$$

with $v_{13} = v_1 + v_3$. In fact the volumes occupied in compartments a, b are the same in both cases. This kind of overloading may be avoided by changing the syntax of the choice operator, for example by specifying the volume occupied in each compartment in a list (associative array):

$$P \equiv [a : v_{13}, b : v_2]n@a.Q_1 + m@b.Q_2 + p@a.Q_3$$

Even if in this way the syntax is more rigorous, it loses readability, so Def. 2.1 is still preferable.

Definition 2.2 The congruence relation $\equiv$ is defined as the least congruence satisfying alpha conversion, the commutative monoidal laws with respect to both ($\mid$,$\mathbf{0}$) and ($+$,$\mathbf{0}$) and the following axioms:

$$\begin{aligned} (\nu\, x)P \mid Q &\equiv (\nu\, x)(P \mid Q) && \text{if } x \notin fn(Q) \\ (\nu\, x)P &\equiv P && \text{if } x \notin fn(P) \\ !\pi.P &\equiv \pi.(\,!\pi.P \mid P) \end{aligned}$$

where the function fn is defined as

$$\begin{aligned} \mathrm{fn}(n@a\!:\!v(\tilde{x})) &\stackrel{def}{=} \{n,a\} & \mathrm{fn}(\overline{n}@a\!:\!v\langle\tilde{x}\rangle) &\stackrel{def}{=} \{n,a,\tilde{x}\} \\ \mathrm{fn}(\mathbf{0}) &\stackrel{def}{=} \emptyset & \mathrm{fn}((\nu\, x)P) &\stackrel{def}{=} \mathrm{fn}(P)\setminus\{x\} \\ \mathrm{fn}(\pi.P) &\stackrel{def}{=} \mathrm{fn}(\pi)\cup\mathrm{fn}(P) & \mathrm{fn}(\textstyle\sum_{i\in I}\pi_i.P_i) &\stackrel{def}{=} \bigcup_i \mathrm{fn}(\pi_i.P_i) \\ \mathrm{fn}(P \mid Q) &\stackrel{def}{=} \mathrm{fn}(P)\cup\mathrm{fn}(Q) & \mathrm{fn}(\,!\pi.P) &\stackrel{def}{=} \mathrm{fn}(\pi.P) \end{aligned}$$

Definition 2.3 Sπ@ semantics is given in terms of the following reduction system:

$$(C)\ \frac{}{(n@a\!:\!v_1(\tilde{x}).P+M) \mid (\overline{n}@a\!:\!v_2\langle\tilde{y}\rangle.Q+N) \xrightarrow{\mathrm{rate}(n)} P\{\tilde{y}/\tilde{x}\} \mid Q}$$

$$(R)\ \frac{P \xrightarrow{r} P'}{(\nu\, x)P \xrightarrow{r} (\nu\, x)P'} \qquad (P)\ \frac{P \xrightarrow{r} P'}{P \mid Q \xrightarrow{r} P' \mid Q}$$

$$(E)\ \frac{P \equiv Q \quad P \xrightarrow{r} P' \quad P' \equiv Q'}{Q \xrightarrow{r} Q'}$$

The rule (C) allows the communication of the names $\tilde{x}$ from process P to Q, where they are properly substituted to names $\tilde{y}$. The function rate : $\mathcal{N} \to (\mathbb{R} \cup +\infty)$ is an external function which permits to associate the correct rate with each reaction, where the rate corresponds to the value r_μ of Expr. (8). Rules $(R), (P), (E)$ allow the transition of processes in presence of restriction, parallel operator or by exploiting structural equivalence.

Definition 2.4 A Sπ@ system S is said to be in *standard form* if

$$S = (\nu\, \tilde{x})\big(P_1 \mid \cdots \mid P_j \mid\ !P_{j+1} \mid \cdots \mid\ !P_k\big)$$

and each P_i is a non-empty sum.

Proposition 2.5 *For every Sπ@ system S, there exists a system S' such that $S \equiv S'$ and S' is in standard form.*

In order to calculate the value $h_{\mu c}$ of Expr. (8), we introduce, according to [17], the function Act which permits to know the number of possible combinations of inputs and outputs on a reaction channel inside a given compartment.

Definition 2.6 The activity Act of channel n inside compartment a in the system S is defined as

$$\mathrm{Act}_{n@a}(S) = (\mathrm{In}_{n@a}(S) \cdot \mathrm{Out}_{n@a}(S)) - \mathrm{Mix}_{n@a}(S)$$

where S is in standard form, $\mathrm{In}_{n@a}(S)$ and $\mathrm{Out}_{n@a}(S)$ are the number of unguarded inputs and outputs on channel n inside compartment a, and $\mathrm{Mix}_{n@a}(S)$ is the sum of $\mathrm{In}_{n@a}(\sum_i) \cdot \mathrm{Out}_{n@a}(\sum_i)$ for each summation $\sum_i$ in S.

The function chan allows to know all the active channels inside each compartment in a given system S.

Definition 2.7 Given a Sπ@ system S in standard form

$$S = (\nu\ \tilde{x})\big(P_1 \mid \cdots \mid P_j \mid\ !P_{j+1} \mid \cdots \mid\ !P_k\big)$$

the function chan is defined recursively as follows:

$$\mathrm{chan}(S) = \bigcup_{i=1}^{k} \mathrm{chan}(P_i) \qquad \mathrm{chan}(\sum_{i\in I} \pi_i.P_i) = \bigcup_{i\in I} \mathrm{chan}(\pi_i)$$
$$\mathrm{chan}(n@a : v(\tilde{x})) = \mathrm{chan}(\overline{n}@a : v\langle\tilde{x}\rangle) = \{n@a\}$$

Definition 2.8 Given a Sπ@ system S in standard form

$$S = (\nu\ \tilde{x})\big(P_1 \mid \cdots \mid P_j \mid\ !P_{j+1} \mid \cdots \mid\ !P_k\big)$$

the volume Vol_a of the compartment a in the system S is calculated as follows:

$$\mathrm{Vol}_a(S) = \sum_{i=1}^{k} \mathrm{Vol}_a(P_i)$$
$$\mathrm{Vol}_a(\sum_{i\in I} \pi_i.P_i) = \sum_{i\in I} \mathrm{Vol}_a(\pi_i)$$
$$\mathrm{Vol}_a(\overline{n}@a : v\langle\tilde{x}\rangle) = \mathrm{Vol}_a(n@a : v(\tilde{x}))$$
$$\mathrm{Vol}_a(n@b : v(\tilde{x})) = \begin{cases} v & a = b \\ 0 & \text{otherwise} \end{cases}$$

If $\mathrm{Vol}_a(S) = 0$, then a is given the default volume value 1.

The following algorithm corresponds to each repetition of the loop of Alg. 2.

Algorithm 3 *Given a Sπ@ system S in standard form, the selection of the next reaction* $\mathrm{Next}(S)$ *and of the delay* $\mathrm{Delay}(S)$ *relative to the MSSA are described by the following algorithm:*

(i) *For each channel* c_i *in* $\mathrm{chan}(S)$*, with* $\mathrm{chan}(S) = \{c_1, \ldots, c_j\}$*, calculate*

$$a_i = \mathrm{Act}_{n@b}(S) * \mathrm{rate}(n) / \mathrm{Vol}_b(S)$$

where $c_i = n@b$ *for some* $n \in \mathcal{N}, b \in \mathcal{C}$.

(ii) *Calculate* $a_0 = \sum_{i=1}^{j} a_i$

(iii) *Generate two random numbers* $z_1, z_2 \in [0,1]$ *and calculate* τ, λ *such that*

$$\tau = (1/a_0)\ln(1/z_1) \qquad \sum_{i=1}^{\lambda-1} a_i < z_2 a_0 \leq \sum_{i=1}^{\lambda} a_i$$

(iv) $\mathrm{Next}(S) = c_\lambda$ *and* $\mathrm{Delay}(S) = \tau$.

The value $c_\lambda = n@b$ for some n, b denotes the reaction channel n (corresponding to μ in Alg. 2) and the compartment c (corresponding to ψ in Alg. 2) of the next reaction happening after τ time. The two processes performing the synchronisation step on c_λ are then randomly chosen as for SPiM.

For further description and applicability of the Sπ@ calculus we refer to [24].

3 Efficient Formulation of the Simulation Algorithm

In the original formulation of the SSA [10] each transition of the system from one state x to a subsequent state x' requires at most M operations needed to know which of the M reactions will happen next, in function of the random number generated in the second step of the algorithm. In fact, in order to find the index μ in Expr. (5), M summations are required in the worst case. Several improvements or alternatives to the SSA have been proposed (e.g. [8,1,7]) which reduce the computational complexity of each transition to $O(\log M)$ or optimise it in function of reaction rates.

The MSSA inherits the complexity order of the original SSA, linear in the number of distinct reactions. However in this case the number of independent reaction is $M \cdot C$, where C is the number of compartments. As noted in [7], the simulation becomes computationally unfeasible if C grows significantly. Unfortunately, none of the proposed improvements can be directly applied to the MSSA with appreciable gain. The main reason is that the propensity functions a_j^k depend of the volumes V_k. Each reaction firing may change the volume of one or more compartments, so that all the propensity functions of reactions located into the involved compartments shall be recalculated. This would cause the complexity of the algorithm to be still linear in the number of the M reactions even after the optimisations proposed in [8,7]. Also the optimised direct method (ODM) formulated in [1] would provide no substantial gain in the (not unusual) case that the chemical composition of the compartments is almost the same: in this situation the complexity would be almost linear in the number of compartments in the best case.

Nevertheless, the computational complexity order of the MSSA can be reduced to $O(\log M + \log C)$ by exploiting the same data structures proposed in [8] for enhancing directly the SSA. These structures are justified by two observations. The first is that only few propensity functions change at each transition and these can be easily identified by building a dependency graph. The second is that the linear

search in step (iii) can be improved by exploiting (twice) a binary search tree. The definitions of the needed data structures follow.

Definition 3.1 Let $\nu^{k\mu}$ be the state change vector of reaction μ firing inside compartment ψ, $\nu^{\psi\mu} = (\tilde{\nu}_1^{\psi\mu}, \ldots, \tilde{\nu}_C^{\psi\mu})$, with $\tilde{\nu}_j^{\psi\mu} = (\nu_{j1}^{\psi\mu}, \ldots, \nu_{jN}^{\psi\mu})$.

Let $\mathrm{Re}(R_\mu) \subseteq \{S_1, \ldots, S_N\}$ be the chemical species needed to fire reaction R_μ.

Let $G(V_G, E_G)$ be a directed graph with vertex set $\{0, \ldots, (C \cdot M) - 1\}$. For each vertex pair $(n, n'), n = f(\psi, \mu), n' = f(\psi', \mu')$, where $f(\psi, \mu) = (\psi * M + \mu)$, the edge $e = (n, n')$ is in E_G iff $\exists j \in \{1, \ldots, N\} : S_j \in \mathrm{Re}(R_{\mu'}) \wedge \nu_{\psi' j}^{\psi\mu} \neq 0$.

G represents the dependency graph of the system. Each vertex n of G represents a reaction R_μ inside a compartment ψ. Every edge from n to n' indicates that reaction R_μ inside ψ influences reaction $R_{\mu'}$ inside ψ' by changing the concentration in ψ' of at least one of the reactants of $R_{\mu'}$. The dependency graph allows to know the only propensity functions which need to be updated after each transition.

We now define the structure of *non-cumulative complete binary search tree*.

Definition 3.2 A binary tree T is recursively defined as nil (the empty tree) or (n, T_l, T_r) where $n = (v, D)$ is the node, $v \in \mathbb{R}^+$ is its value and D the associated data, T_l and T_r are binary trees.

Let $\mathrm{hgt}(T)$ be the height of the tree, with $\mathrm{hgt}(\mathrm{nil}) = 0$ and $\mathrm{hgt}((n, T_l, T_r)) = 1 + \max(\mathrm{hgt}(T_l), \mathrm{hgt}(T_r))$.

Let $\mathrm{lev}_i(T), i \geq 0$ be the list of elements of the i-th level of the tree, with $\mathrm{lev}_i(\mathrm{nil}) = [\emptyset]$, $\mathrm{lev}_0((n, T_l, T_r)) = [n]$ and $\mathrm{lev}_i((n, T_l, T_r)) = \mathrm{lev}_{i-1}(T_l) \& \mathrm{lev}_{i-1}(T_r)$ for $i > 0$, with & representing the appending of lists.

Let $\mathrm{el}(T)$ be the list of element values of T, with $\mathrm{el}(\mathrm{nil}) = []$, $\mathrm{el}(((v, D), T_l, T_r)) = [v] \& \mathrm{el}(T_l) \& \mathrm{el}(T_r)$.

A binary tree T is *complete* if is empty, or if $\emptyset \notin \mathrm{lev}_{\mathrm{hgt}(T)-2}(T)$ and there exist no lists l_1, l_2 and node n, such that $\mathrm{lev}_{\mathrm{hgt}(T)-1}(T) = l_1 \& [\emptyset, n] \& l_2$.

A complete binary tree is a non-cumulative binary search tree (NCBST) if it is empty, or if $T = ((v, D), \mathrm{nil}, \mathrm{nil})$, or if $T = ((v, D), T_l, T_r)$ and $v = \sum \mathrm{el}(T_l) + \sum \mathrm{el}(T_r)$, and both T_l and T_r are non-empty non-cumulative search trees.

A binary tree is complete if all the levels are full, except for the last which presents all the remaining leaves on the left. The tree is also a non-cumulative search tree if each node value is equal to the sum of the values of its offspring. The definitions are illustrated in Fig. 1. A non-cumulative binary search tree can be transformed into a binary search tree by summing up the value of each node to all the nodes of its right sub-tree.

We define now the function which implements the search in a NCBST.

Algorithm 4 *Let $T = ((v, D), T_l, T_r)$ be a non-empty NCBST and $p \in \mathbb{R}, p \in [0, v[$, with T, p formal parameters of the function:*

(i) *if $T_l = T_r = \mathrm{nil}$ (i.e. T is a leaf) then return $(p/v, D)$, else*

(ii) *let $T_l = ((v^l, D), T_l^l, T_r^l)$; if $p < v^l$ then set $T \leftarrow T_l$ and go to (i), else*

(iii) *set $p \leftarrow (p - v^l), T \leftarrow T_r$ and go to (i).*

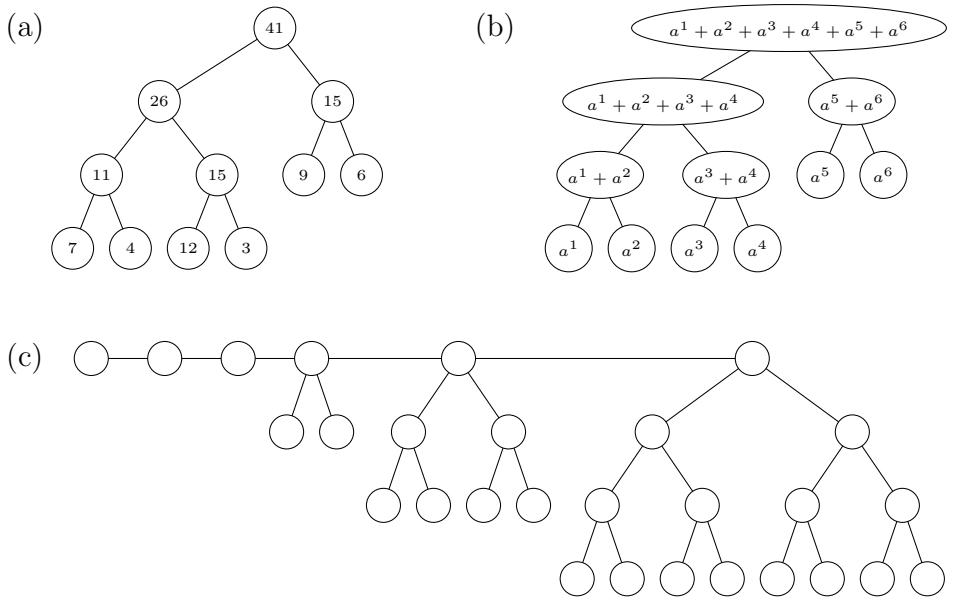

Fig. 1. Non-cumulative complete binary search trees.

Given a NCBST $T = ((v, D), T_l, T_r)$ and a random number $p = r \cdot v$, with $r \in [0, 1[$, the function returns the data associated with the leaf i, where

$$\sum_{j<i}(\text{leaf j value}) < p < \sum_{j\leq i}(\text{leaf j value})$$

and leaves are numbered from left to right, as in Fig. 1 (b). The function returns also the remainder of p scaled to the interval $[0, 1[$, which allows to avoid the generation of a further random number in the next algorithm. A short example of execution follows.

Example 3.3 Consider the tree in Fig. 1 (a). Let $p = 20$. The root node is not a leaf, so it must be checked if $p < v_l = 26$, where v_l is the value associated with the left sub-tree. The condition is true, so the loop cycle must be repeated starting from T_l. The value of the left sub-sub-tree is $v_{ll} = 11 < p$, hence $p \leftarrow 20 - 11$ and the search continues in T_{lr}. Now $p = 9 < v_{lrl} = 12$ and T_{lrl} is a leaf, so the algorithm ends and returns $(0.75, D)$.

The computational complexity of the algorithm is $O(\log K)$, where K is the number of nodes of the tree. Given a list of l non-negative real numbers, it is also possible to build a corresponding NCBST in $O(l)$ with all the elements of the list appearing as leaves of the tree. This is the technique exploited for executing step (iii) of the MSSA in logarithmic time.

We can now define the improved MSSA. We first note that the values a^k in Expr.

(10) can be written as

$$\begin{aligned} a^k(x) &= \textstyle\sum_{j=1}^{M} a_j^k(x) = \sum_{j=1}^{M} V_k^{-1}(x) h_j^k(x) r_\mu \\ &= V_k^{-1}(x) \textstyle\sum_{j=1}^{M} h_j^k(x) r_\mu = V_k^{-1}(x) \alpha^k(x) \end{aligned} \tag{13}$$

where

$$\alpha^k(x) = \sum_{j=1}^{M} h_j^k(x) r_\mu = \sum_{j=1}^{M} \alpha_j^k(x) \tag{14}$$

with

$$\alpha_j^k(x) = h_j^k(x) r_\mu \tag{15}$$

Furthermore, the summation of Expr. (12) can be expressed as

$$\begin{aligned} \textstyle\sum_{k=1}^{\psi} \sum_{j=1}^{\mu} a_j^k(x) = \textstyle\sum_{k=1}^{\psi-1} a^k(x) + \sum_{j=1}^{\mu} a_j^\psi(x) = \\ \textstyle\sum_{k=1}^{\psi-1} a^k(x) + V_\psi^{-1} \sum_{j=1}^{\mu} \alpha_j^\psi(x) \end{aligned} \tag{16}$$

The enhanced MSSA (EMSSA) is defined as follows.

Algorithm 5 *(Enhanced Multi-compartmental Stochastic Simulation Algorithm)*

(i) *calculate* V_k *from* (9), α_j^k *from Expr.* (15), α^k *from Expr.* (14), a^k *from Expr.* (13), *with* $k = 1, \ldots, C, \quad j = 1, \ldots, M$, *and* a_0 *from Expr.* (10);

(ii) *build the dependency graph of the system according to Def. 3.1;*

(iii) *build the NCBST* T_0 *such that its leaves are* $((a^k, k), \text{nil}, \text{nil})$, *with* $k = 1, \ldots, C$;

(iv) *build* C *NCBSTs such that* $((\alpha_j^k, j), \text{nil}, \text{nil})$ *are the leaves of the* k*-th NCBST* T^k, *with* $j = 1, \ldots, M$ *and* $k = 1, \ldots, C$;

(v) *generate uniformly two random numbers* z_1, z_2 *in the interval* $(0, 1)$;

(vi) *calculate* τ *from Expr.* (11) *and set* $t = t + \tau$;

(vii) *let* (v, D) *the value returned by Alg. 4 called with parameters* $(T_0, a_0 \cdot z_2)$; *set* $\psi \leftarrow D$, *according to Expr.* (16);

(viii) *let* (v', D) *the value returned by Alg. 4 called with parameters* $(T^\psi, \alpha^\psi \cdot v)$; *set* $\mu \leftarrow D$, *according to Expr.* (16);

(ix) *update the states of the species and* $T_0, T^1, \ldots, T^C$ *to reflect the execution of reaction* (ψ, μ) *by updating only the propensity functions and volumes and subtrees indicated by the dependency graph built at step* (ii);

(x) *go to step* (v).

The initialisation of the algorithm includes the building of the dependency graph and of *two kinds* of NCBSTs. The first, constituted by T_0, is the only which contains informations on the V_k volumes of the compartments. The second kind,

represented by $\{T^1, \ldots, T^C\}$, contains the informations on the propensity functions of the reactions inside each compartment. This double-tree organisation allows to express the propensity functions independently of their respective volumes, so that they do not need to be recalculated as the volumes change.

The height of T_0 is $\lceil \log(2 \cdot C) \rceil$, while the height of each T^k is $\lceil \log(2 \cdot M) \rceil$. The search of steps (vii) and (viii) are consequently executed in $O(\log C)$ and $O(\log M)$ respectively. The most expensive operation is step (ix): if Max_V is the maximum number of compartments influenced by some reaction R_μ and Max_R is the maximum number of propensity functions modified by some reaction R'_μ, the number of operations is bounded by $(\mathrm{Max}_V \log C + \mathrm{Max}_R \log M)$, because each update of the leaf of a NCBST T requires $\mathrm{hgt}(T)$ updates of the ancestor nodes. Since $\mathrm{Max}_V, \mathrm{Max}_R$ are constants, the computational complexity of the algorithm is $O(L(\log C + \log M))$, where L is the number of transitions of the systems, each corresponding to one reaction firing and one execution of the loop in Alg. 5.

3.1 Further enhancements

The number of operations needed to perform step (ix) can be considerably reduced by grouping together the indexes of the compartments and reactions whose volumes and propensity functions change simultaneously. This can be achieved by a proper analysis of the dependency graph of the system and may lead in the best case to $(2\,\mathrm{Max}_V + 2\,\mathrm{Max}_R + \log C + \log M)$ operations. Steps (vii) and (viii) can be improved by adapting the enhancements to the SSA discussed in [1] to NCBSTs. The NCBST structure may be changed as shown in Fig. 1 (c). Here the leaves of NCBSTs of increasing height linked in a list contain the propensity function values in increasing order of firing frequency of the corresponding reactions. The frequencies can be calculated by previous benchmarking, as in [1].

3.2 Related work

Some extensions of the Gillespie algorithm which handle variable volumes were already defined in [13,12]. The volume here is expressed as known function of time and introduced in the evaluation of the propensity functions. Although this approach may likely provide less computationally expensive algorithms, it can be applied only in the case that the variation of the volume in function of time can be considered deterministic and it is known. Conversely, the MSSA allows to associate the volume with the propensity functions so that its variation is introduced transparently in the stochastic evolution of the system and is handled coherently with the CME without requiring previous knowledge of its behaviour.

The next subvolume method (NSM) [7] faces the problem of efficient simulation of chemical systems in presence of molecular diffusion. The NSM is thought for a high number of subvolumes with statical structure, of fixed and equal size. Any implementation of the EMSSA would likely be slower in the special case considered for the NSM, but constitutes an efficient generalisation in the case of dynamical structure and different, varying volumes. In fact, the computational complexity

order of the EMSSA is the same of the NSM, since they are both logarithmic in function of the number of compartments and reactions.

4 Conclusion

We have presented an enhanced stochastic simulation algorithm based on Gillespie SSA [10] which allows to model systems with multiple compartments and variable volumes. Its computational complexity scales logarithmically – instead of linearly – with both the number of reactions and the number of compartments, so that it turns out to be very efficient also in the case of high number of compartments. The algorithm can be used for an improved implementation of the Sπ@ calculus [24] which constitutes an elegant candidate for the representation of systems with dynamical compartment structure and varying volumes. The improved Sπ@ calculus allows the efficient and exact stochastic simulation of biological systems also in presence of phenomena like osmosis, cellular growth and division [13], diffusion [7].

Although the algorithm does not require to know the volume associated with each molecule species (as discussed in [24]), the formulation of a specific kinetic model for the estimation of reactant densities would allow to refine the simulation and is left for future work.

The same optimisations considered here may also be used for an efficient implementation of the SSA in the case of multiple compartments with fixed volumes but variable temperature.

References

[1] Cao, Y., H. Li and L. Petzold, *Efficient formulation of the stochastic simulation algorithm for chemically reacting systems.*, J Chem Phys **121** (2004), pp. 4059–4067.

[2] Carbone, M. and S. Maffeis, *On the expressive power of polyadic synchronisation in pi-calculus.*, Nord. J. Comput. **10** (2003), pp. 70–98.

[3] Cardelli, L., *Brane calculi.*, in: Danos and Schächter [6], pp. 257–278.

[4] Cleaveland, R., G. Lüttgen and V. Natarajan, *Priority in process algebra*, in: J. Bergstra, A. Ponse and S. Smolka, editors, *Handbook of Process Algebra*, Elsevier Science Publishers, 2001 pp. 711–765.

[5] Danos, V. and C. Laneve, *Formal molecular biology.*, Theor. Comput. Sci. **325** (2004), pp. 69–110.

[6] Danos, V. and V. Schächter, editors, "Computational Methods in Systems Biology, International Conference CMSB 2004, Paris, France, May 26-28, 2004, Revised Selected Papers," Lecture Notes in Computer Science **3082**, Springer, 2005.

[7] Elf, J. and M. Ehrenberg, *Spontaneous separation of bi-stable biochemical systems into spatial domains of opposite phases*, Systems Biology **1** (2004), pp. 230–236.
URL http://link.aip.org/link/?BDJ/1/230/1

[8] Gibson, M. and J. Bruck, *Efficient exact stochastic simulation of chemical systems with many species and many channels*, Journal of Physical Chemistry A **104** (2000), pp. 1876–1889.
URL http://pubs3.acs.org/acs/journals/doilookup?in_doi=10.1021/jp993732q

[9] Gillespie, D. T., *A general method for numerically simulating the stochastic time evolution of coupled chemical reactions*, Journal of Computational Physics **22** (1976), pp. 403–434.
URL http://dx.doi.org/10.1016/0021-9991(76)90041-3

[10] Gillespie, D. T., *Exact stochastic simulation of coupled chemical reactions*, J. Phys. Chem. **81** (1977), pp. 2340–2361.

[11] Kuttler, C., C. Lhoussaine and J. Niehren, *A stochastic pi calculus for concurrent objects*, in: *Proceedings of the Second International Conference on Algebraic Biology*, Lecture Notes in Computer Science (2007).

[12] Lecca, P., *A time-dependent extension of gillespie algorithm for biochemical stochastic π-calculus*, in: *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing* (2006), pp. 137–144.

[13] Lu, T., D. Volfson, L. Tsimring and J. Hasty, *Cellular growth and division in the gillespie algorithm*, in: *Systems Biology, IEE Proceedings*, 2004, pp. 121–128.

[14] Milner, R., "Communicating and mobile systems: the π-calculus," Cambridge University Press, New York, NY, USA, 1999.

[15] Milner, R., J. Parrow and D. Walker, *A calculus of mobile processes, i*, Inf. Comput. **100** (1992), pp. 1–40.

[16] Milner, R., J. Parrow and D. Walker, *A calculus of mobile processes, ii*, Inf. Comput. **100** (1992), pp. 41–77.

[17] Phillips, A. and L. Cardelli, *A correct abstract machine for the stochastic pi-calculus*, in: *Bioconcur'04* (2004).

[18] Priami, C., *Stochastic pi-calculus.*, Comput. J. **38** (1995), pp. 578–589.

[19] Priami, C. and P. Quaglia, *Beta binders for biological interactions.*, in: Danos and Schächter [6], pp. 20–33.

[20] Priami, C., A. Regev, E. Y. Shapiro and W. Silverman, *Application of a stochastic name-passing calculus to representation and simulation of molecular processes.*, Inf. Process. Lett. **80** (2001), pp. 25–31.

[21] Regev, A., E. M. Panina, W. Silverman, L. Cardelli and E. Y. Shapiro, *Bioambients: an abstraction for biological compartments.*, Theor. Comput. Sci. **325** (2004), pp. 141–167.

[22] Regev, A., W. Silverman and E. Y. Shapiro, *Representation and simulation of biochemical processes using the pi-calculus process algebra.*, in: *Pacific Symposium on Biocomputing*, 2001, pp. 459–470.

[23] Versari, C., *A core calculus for a comparative analysis of bio-inspired calculi*, European Symposium on Programming (2007), available at http://www.cs.unibo.it/∼versari/files/cversari-esop07.pdf.

[24] Versari, C. and N. Busi, *Stochastic simulation of biological systems with dynamical compartment structure*, in: *CMSB*, 2007, available at http://www.cs.unibo.it/∼versari/files/cversari-cmsb07.pdf. URL http://www.cs.unibo.it/~versari/files/cversari-cmsb07.pdf