Replace this file with `prentcsmacro.sty` for your meeting, or with `entcsmacro.sty` for your meeting. Both can be found at the ENTCS Macro Home Page.

Spatial Calculus of Looping Sequences[1]

Roberto Barbuti,[2] Andrea Maggiolo–Schettini,[3]
Paolo Milazzo[4] and Giovanni Pardini[5]

Dipartimento di Informatica
Università di Pisa
Largo Bruno Pontecorvo 3, 56127 Pisa, Italy

Abstract

The Calculus of Looping Sequences (CLS) enables the description of biological systems and of their evolution. This paper presents the Spatial CLS, an extension of CLS that allows the description of the position of biological elements, and of the space they take up in a 2D/3D space. The elements may move autonomously during the passage of time, and may interact when constraints on their positions are satisfied. The space occupied by each element is modeled as a hard sphere, hence space conflicts may arise during system evolution. These conflicts are resolved by an appropriate algorithm, which rearranges the position of the elements by assuming that they push each other when they are too close. Moreover, rewrite rules are endowed with a parameter describing their reaction rate.
The aim of Spatial CLS is to enable a more accurate description of those biological processes whose behaviour depends on the exact position of the elements. As example applications of the calculus, we present a model of cell proliferation, and a model of the quorum sensing process in *Pseudomonas aeruginosa*.

Keywords: Calculus of Looping Sequences, Spatial modeling, Systems biology

1 Introduction

The study of biological systems has traditionally involved the development of mathematical models (e.g. differential equations) for the description and analysis of their behaviour. New approaches for the modeling of biological processes have been recently proposed, which involve the use of modeling formalisms of Computer Science such as process calculi [10,25,23,22,9,24,28] automata-based models [4,18] and rewrite systems [11,20,8,19,16]. Their use comprises the development of simulators, and the analysis of the systems and the verification of their properties by using means of Computer Science, such as probabilistic model checking [17].

[1] This research has been partially supported by MiUR PRIN 2006 Project "Biologically Inspired Systems and Calculi and their Applications (BISCA)".

[2] Email: barbuti@di.unipi.it
[3] Email: maggiolo@di.unipi.it
[4] Email: milazzo@di.unipi.it
[5] Email: pardinig@di.unipi.it

©2008 Published by Elsevier Science B. V.

Among the simulators we quote SPiM [3] and BioSPI [2], both based on (stochastic) π-calculus, and SCLSm [1]. MGS [14] is a dynamically typed functional language, that allows describing transformations over collections of elements. Its main purpose is to allow the simulation of biological processes, and it is proposed as a unifying framework of many different models of computation inspired by biology and chemistry.

Recently, in order to model the protein chemistry of biological cells for phenomena where spatial effects are important, particle-based simulators have been developed [27,21] and a spatial extension of the π-calculus has been proposed [15].

The Calculus of Looping Sequences (CLS) [8,19] allows the modelling of biological systems and of their evolution. It is based on term rewriting, hence a CLS model is composed of a term, which describes the biological system, and a set of rewrite rules, modeling its evolution. Two kinds of structures are provided by the calculus: *sequences*, used to represent simple entities of biological systems such as proteins and DNA strands, and *looping sequences* that can be used to model more complex structures such as membranes.

In this paper we present the Spatial Calculus of Looping Sequeces (Spatial CLS), which extends CLS by allowing spatial information to be associated with CLS structures when this information is relevant for determining the system behaviour. Such structures are represented as *hard spheres* in a continuous space, which can move autonomously and can interact when conditions on their positions are satisfied. Conflicts on the space occupied by different structures may arise during evolution. Elements should not overlap and should not be positioned beyond the bounds of the membrane containing them. These conflicts are resolved by a suitable algorithm that rearranges the elements by assuming that they push each other if they overlap and that they are kept inside the membrane containing them when they exceed its bounds. Finally, rewrite rules are endowed with a parameter describing their reaction rate, that is their propensity to occur when applicable.

The aim of Spatial CLS is to enable a more accurate description of those biological processes whose behaviour depends on the exact position of the elements. This high level of accuracy is especially useful for cell biology, where there can be a high degree of spatial organization and molecular species may be distributed in the space not uniformly [5]. Such descriptions can then be used to simulate the system, so as to obtain a faithful representation of their evolution. As example applications of the calculus, we present a model of cell proliferation, and a model of the *quorum sensing* process in *Pseudomonas aeruginosa*. We show the result of the simulation of the former model, obtained with an ad hoc simulator.

A formalism dealing with spatial aspects of biological systems is SpacePI [15], an extension of the π-calculus with space and time. In SpacePI positions in a continuous space (such as $\mathbb{R}^2$) are associated with processes, and processes can move autonomously according to a movement function. However, the space occupied by a process cannot be expressed, and membranes cannot be modeled easily.

2 The Calculus of Looping Sequences

We recall the variant of CLS called CLS+ [19]. In the definition of the syntax of CLS+ terms we assume an infinite alphabet $\mathcal{E}$ ranged over by $a, b, c, \ldots$.

Definition 2.1 (Terms) Terms T, branes B *and* sequences S *of CLS+ are given by the following grammar:*

$$\begin{array}{lcl} T & ::= & S \mid (B)^L \rfloor T \mid T \mid T \\ B & ::= & S \mid B \mid B \\ S & ::= & \epsilon \mid a \mid S \cdot S \end{array}$$

The sets of all terms, branes and sequences are denoted by $\mathcal{T}, \mathcal{B}$ *and* $\mathcal{S}$*, respectively.*

The sequencing operator $\_ \cdot \_$ can be used to build sequences of symbols in $\mathcal{E}$ and ϵ denotes the empty sequence, that is a concatenation of zero symbols. For constructing terms, we have a looping operator $(\_)^L$, a containment operator $\_ \rfloor \_$ and a parallel composition operator $\_ \mid \_$. A term may contain simple sequences S and looping sequences $(B)^L \rfloor T$. The containment operator $\rfloor$ allows the representation of compartments; in fact, a looping sequence $(B)^L \rfloor T$ usually models a membrane with a surface modeled by B (a parallel composition of sequences) and a content modeled by T. Since, in CLS+, looping $(\_)^L$ and containment $\_ \rfloor \_$ are always applied together, we can consider them as a single binary operator which applies to a brane and to a term. Brackets can be used to indicate the order of application of the operators, and $(\_)^L \rfloor \_$ is assumed to have precedence over $\_ \mid \_$.

The structural congruence relation on terms identifies syntactically different terms that conceptually represent the same structure.

Definition 2.2 (Structural congruence) *The* structural congruence *relations on sequences* $\equiv_S$ *and terms* $\equiv_T$ *are the least congruences satisfying the following rules:*

$$S_1 \cdot (S_2 \cdot S_3) \equiv_S (S_1 \cdot S_2) \cdot S_3 \qquad S \cdot \epsilon \equiv_S \epsilon \cdot S \equiv_S S \qquad S_1 \equiv_S S_2 \Rightarrow S_1 \equiv_T S_2$$
$$T \mid \epsilon \equiv_T T \qquad T_1 \mid (T_2 \mid T_3) \equiv_T (T_1 \mid T_2) \mid T_3 \qquad T_1 \mid T_2 \equiv_T T_2 \mid T_1$$

The structural congruence states the associativity of both the sequencing and the parallel operator, the commutativity of the latter, and the neutral role of ϵ.

The evolution of a system is described by a set of rewrite rules, modeling reactions among system elements. A rule is composed of a pair of *patterns* (terms with variables) with the intuitive meaning that, if the first pattern occurs in a portion of the system, then that portion can be modified according to the second pattern.

We assume the following infinite and pairwise disjoint sets of variables: SV for sequence variables $\widetilde{x}, \widetilde{y}, \ldots$; $\mathcal{X}$ for element variables $x, y, \ldots$; TV for term variables $X, Y, \ldots$ and BV for brane variables $\overline{X}, \overline{Y}, \ldots$. We denote the set of all variables by $\mathcal{V}$. We distinguish among different kinds of pattern, as in the following definitions.

Definition 2.3 [Brane and sequence patterns] *Brane patterns* BP and *sequence patterns* SP are given by the following grammar:

$$BP ::= SP \mid BP \mid BP \qquad SP ::= \epsilon \mid a \mid SP \cdot SP \mid \widetilde{x} \mid x$$

We denote the sets of all brane and sequence patterns with $\mathcal{BP}$ and $\mathcal{SP}$, respectively.

Definition 2.4 [Term patterns] *Left patterns* P_L and *right patterns* P_R are given by the following grammar:

$$\begin{array}{rcl} P_L & ::= & SP \mid \left(BP_{LX}\right)^L \rfloor P_{LX} \mid P_L \mid P_L \\ BP_{LX} & ::= & BP \mid BP \mid \overline{X} \\ P_{LX} & ::= & P_L \mid P_L \mid X \\ P_R & ::= & SP \mid \left(BP_{RX}\right)^L \rfloor P_R \mid P_R \mid P_R \mid X \mid \overline{X} \\ BP_{RX} & ::= & BP \mid BP_{RX} \mid \overline{X} \end{array}$$

We denote the sets of all left and right patterns with $\mathcal{P}_L$ and $\mathcal{P}_R$, respectively. We assume brane patterns to be a subset of left and right patterns, i.e. $\mathcal{BP} \subset \mathcal{P}_L (\subset \mathcal{P}_R)$.

The set of all variables appearing in a pattern P is denoted by $\mathrm{Var}(P)$. We also assume the structural congruence relation to be extended to patterns.

A CLS+ term evolves by applying rewrite rules to it. A *rewrite rule* is a pair of patterns (P_L, P_R), usually written as $P_L \mapsto P_R$, such that $P_L \not\equiv \epsilon$ and $\mathrm{Var}(P_R) \subseteq \mathrm{Var}(P_L)$. If P_L and P_R are indeed brane patterns, then the rule is called *brane (rewrite) rule*.

Given a pattern, we may obtain a term by applying an *instantiation function* $\sigma : \mathcal{V} \to \mathcal{T}$, describing the bindings between variables and their values. This application, denoted by $P\sigma$, replaces each occurrence of $v \in \mathrm{Var}(P)$ in P with $\sigma(v)$. For example, we can instantiate the pattern $P = \left(a \cdot \widetilde{x} \mid \overline{X}\right)^L \rfloor (c \mid Y)$ with instantiation $\sigma = \{(\widetilde{x}, b \cdot b), (\overline{X}, a \cdot b \cdot b \mid d \cdot b \cdot b), (Y, c \mid d)\}$ obtaining the term $P\sigma = \left(a \cdot b \cdot b \mid a \cdot b \cdot b \mid d \cdot b \cdot b\right)^L \rfloor (c \mid c \mid d)$. An instantiation function σ must respect the type of variables, namely for all $X \in TV, \overline{X} \in BV, \widetilde{x} \in SV$ and $x \in \mathcal{X}$ we have $\sigma(X) \in \mathcal{T}$, $\sigma(\overline{X}) \in \mathcal{B}$, $\sigma(\widetilde{x}) \in \mathcal{S}$ and $\sigma(x) \in \mathcal{E}$, respectively. The set of all instantiations is denoted by Σ.

A rewrite rule $P_L \mapsto P_R$ states that a term $P_L\sigma$, obtained by instantiating variables in P_L by some instantiation function σ, can be transformed into the term $P_R\sigma$. Thus, a term T may evolve to another term T' by applying a rewrite rule to a subterm of T.

The use of different kinds of patterns allows us to constrain the occurrences of variables inside them. These constraints are useful to simplify the semantics and, as explained in [7], they do not restrict the expressive power of the calculus for modeling usual biological systems. First of all, brane and term variables may occur only on branes and inside looping sequences, respectively. Then, with respect to left patterns, term variables are not allowed at top–level, and at most one brane or term variable is allowed in each compartment. We do not allow term variables on branes: this ensures that the application of a rewrite rule never yields an invalid term, i.e. a term with looping sequences on branes. For the same reason, we identify brane rewrite rules $BP_1 \mapsto BP_2$ as the only kind of rules that can be applied to branes.

The semantics of the calculus is given as a transition system, in which states correspond to terms, and each transition $\to$ represent the applications of a rewrite rule. The definition uses the auxiliary transition relation $\to_{\mathcal{B}}$, that describes the

evolution of branes (ensuring that only brane rewrite rules can be applied to their elements).

Definition 2.5 [Semantics] Given a set of rewrite rules $\mathcal{R}$, let $\mathcal{R}_\mathcal{B}$ denote its subset of all and only brane rules ($\mathcal{R}_\mathcal{B} \subseteq \mathcal{R}$). The *semantics* of CLS+ is the least transition relation $\rightarrow$ on terms closed under $\equiv_T$ and satisfying the following inference rules:

$$\frac{P_1 \mapsto P_2 \in \mathcal{R} \qquad P_1\sigma \not\equiv \epsilon \qquad \sigma \in \Sigma}{P_1\sigma \rightarrow P_2\sigma} \qquad \frac{T_1 \rightarrow T_1'}{T_1 \mid T_2 \rightarrow T_1' \mid T_2}$$

$$\frac{BP_1 \mapsto BP_2 \in \mathcal{R}_\mathcal{B} \qquad BP_1\sigma \not\equiv \epsilon \qquad \sigma \in \Sigma}{BP_1\sigma \rightarrow_\mathcal{B} BP_2\sigma} \qquad \frac{B_1 \rightarrow_\mathcal{B} B_1'}{B_1 \mid B_2 \rightarrow_\mathcal{B} B_1' \mid B_2}$$

$$\frac{T_1 \rightarrow T_2}{(B)^L \rfloor T_1 \rightarrow (B)^L \rfloor T_2} \qquad \frac{B \rightarrow_\mathcal{B} B'}{(B)^L \rfloor T \rightarrow (B')^L \rfloor T}$$

As an example, let $T = (a \cdot b \cdot b \mid d \cdot b)^L \rfloor (c \mid d)$ and let $(a \cdot \widetilde{x} \mid \overline{X})^L \rfloor (c \mid Y) \mapsto (d \cdot \widetilde{x} \mid \overline{X})^L \rfloor Y$ be a rewrite rule modeling the formation of a complex on a membrane by the interaction of an element $a \cdot \widetilde{x}$ on the membrane with an element c inside the membrane. By applying the rule to T, we obtain $(d \cdot b \cdot b \mid d \cdot b)^L \rfloor d$ where $d \cdot b \cdot b$ is the resulting complex.

3 The Spatial CLS

The definition of Spatial CLS is based on CLS+, recalled in Section 2. The syntax of terms is an extension of that of CLS+, where sequences and membranes are enriched with spatial information. We assume an alphabet $\mathcal{E}$ (as in CLS+), and a set $\mathcal{M}$ of names denoting movement functions.

Definition 3.1 *Terms* T, *branes* B and *sequences* S of Spatial CLS are defined as:

$$\begin{array}{lcl} T & ::= & \lambda \mid (S)_d \mid (B)^L_d \rfloor T \mid T \mid T \\ B & ::= & (S)_d \mid B \mid B \\ S & ::= & \epsilon \mid a \mid S \cdot S \end{array}$$

where $d \in \mathcal{D} = ((\mathbb{R}^n \times \mathcal{M}) \cup \{\cdot\}) \times \mathbb{R}^+$. The set of all sequences, branes and terms are denoted by $\mathcal{S}$, $\mathcal{B}$ and $\mathcal{T}$, respectively. We assume that $\mathcal{B} \subset \mathcal{T}$.

The term λ denotes the *empty term*, while ϵ denotes the empty sequence. The parameter d associated with the elements describes their spatial information. The calculus allows representing two kinds of elements, *positional* and *non-positional*, depending on the form of the parameter d. For positional elements $d = \langle [p, m], r \rangle$, where p gives the position of the center of the sphere modeling the space occupied by the element, m is the movement function and r is the radius of the sphere. For non-positional elements $d = \langle \cdot, r \rangle$, as we do not keep track of their position and movement function (replaced by the special symbol $\cdot$), but only of their radius r. For looping sequences $(\_)^L_d \rfloor \_$, r describes the available space inside the membrane. The position of positional elements appearing on the brane of a looping sequence or inside it, are relative to the center of the looping sequence itself.

Non-positional elements correspond to CLS terms in Spatial CLS, and therefore their behaviour is in accordance with the Law of Mass Action: they are assumed homogeneously distributed in the space available inside the compartment, and the reaction rate of the rules involving those elements is proportional to the product of the concentrations of the reactants. Therefore Spatial CLS allows descriptions at different levels, as precise spatial information can be used only for those elements for which it is relevant.

Each $m \in \mathcal{M}$ denotes a function m_{fun} describing the autonomous movement of an element over time. In particular, m_{fun} computes the new position $p' = m_{\text{fun}}(p, r, x, l, t, \delta t)$ of an element after a time interval δt from the current time t, by taking into account its current position p, its radius r, and the parameters x and l specifying where the element appears: either on the surface (on) or inside (in) a membrane with radius l, or at top–level (in such case $x = in$, $l = \infty$). Thus $m_{\text{fun}} : \mathbb{R}^n \times \mathbb{R}^+ \times \{in, on\} \times (\mathbb{R}^+ \cup \{\infty\}) \times \mathbb{R}^+ \times \mathbb{R}^+ \to \mathbb{R}^n$. For example, a linear motion can be modeled by a movement function defined as $m_{\text{fun}}(p, r, x, l, t, \delta t) = \boldsymbol{q} + \boldsymbol{v}t$, where $\boldsymbol{q}$ is the initial position and $\boldsymbol{v}$ is the velocity. A useful movement function, which is often needed in models, is the one associated with the elements that do not move autonomously. We denote it as m_0, and define it as $m_{0\text{fun}}(p, r, x, l, t, \delta t) = p$.

This formalization of Spatial CLS does not allow direct description of stochastic motions, such as *Brownian motion*. However, it can be extended to allow pseudo-random motion (that is still deterministic) by introducing another parameter for movement functions. This parameter represents a *seed* used to initialize their pseudo-random behaviour, similarly to the initialization of a pseudo-random number generator. The (multiset of) possible values for the seed could be specified in the spatial information of positional elements. In this way, we could define a movement function which realizes, depending on the parameter, a different trajectory for the stochastic motion. Because of lack of space, we cannot actually show this extension of the calculus, but in the examples we will assume it to be available.

Definition 3.2 The *structural congruence* relations on sequences $\equiv_S$ and on terms $\equiv_T$ are the least congruences satisfying the following rules:

$$S_1 \cdot (S_2 \cdot S_3) \equiv_S (S_1 \cdot S_2) \cdot S_3 \qquad S \cdot \epsilon \equiv_S \epsilon \cdot S \equiv_S S \qquad S_1 \equiv_S S_2 \Rightarrow \left(S_1\right)_d \equiv_T \left(S_2\right)_d$$
$$T \mid \lambda \equiv_T T \qquad T_1 \mid (T_2 \mid T_3) \equiv_T (T_1 \mid T_2) \mid T_3 \qquad T_1 \mid T_2 \equiv_T T_2 \mid T_1$$

From a biological point of view, we cannot consider all possible terms to be valid. Intuitively, we have to avoid that different elements occupy the same space. In particular, we introduce the following constraints for positional elements:

- elements inside a membrane (or at top–level) must not occupy the same space;
- elements of a brane must not occupy the same space;
- elements of a brane must not occupy the space of any other element either inside or outside the membrane;
- elements inside a membrane must not exceed the limits of the sphere representing the membrane;
- the center of the elements of a brane must be located on the surface of the sphere representing the membrane.

Moreover, we want to ensure that the space occupied by all the elements in a membrane does not exceed the volume of the membrane. To take into account the space occupied by non–positional elements, we assume a function SpaceCheck that determines whether there is enough space in a membrane for all the elements inside it, and for all those on its surface. The constraints described are captured by the following definition of *well-formedness*, where we assume the function dist : $\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}^+$ that gives the distance between two points.

Definition 3.3 [Well-formed terms] The set of *well-formed terms* is defined as [6]

$$\mathcal{T}_{wf} = \{T \in \mathcal{T} \mid \exists I \in \mathcal{I}.\ I \models T\}$$

where $\mathcal{I} = \mathcal{P}(\mathcal{J} \times \mathcal{P}(\mathcal{J}))$, $\mathcal{J} = (\mathbb{R}^n \cup \{\cdot\}) \times \mathbb{R}^+$, and relation $\_ \models \_ \subseteq \mathcal{I} \times \mathcal{T}$ is defined by the following inference rules (where $\mathbf{0}$ denotes the null vector):

$$\emptyset \models \lambda \qquad \{(\cdot, r, \emptyset)\} \models (S)_{\cdot,r} \qquad \{(p, r, \emptyset)\} \models (S)_{[p,m],r}$$

$$\frac{\begin{array}{c} I_1 \models B \qquad I_2 \models T \qquad \mathrm{SpaceCheck}(r, I_1, I_2) = tt \\ \forall\, (p_1, r_1) \in \mathrm{All}(I_1).\ \mathrm{dist}(\mathbf{0}, p_1) = r \qquad \forall\, (p_2, r_2) \in \mathrm{All}(I_2).\ \mathrm{dist}(\mathbf{0}, p_2) + r_2 \leq r \\ \forall\, (p_1, r_1) \in \mathrm{All}(I_1),\ (p_2, r_2) \in \mathrm{All}(I_2).\ \mathrm{dist}(p_1, p_2) \geq r_1 + r_2 \end{array}}{\{(\cdot, r, \mathrm{All}(I_1))\} \models (B)^L_{\cdot,r} \rfloor T}$$

$$\frac{\{(\cdot, r, J)\} \models (B)^L_{\cdot,r} \rfloor T}{\{(p, r, J)\} \models (B)^L_{[p,m],r} \rfloor T} \qquad \frac{\begin{array}{c} I_1 \models T_1 \quad I_2 \models T_2 \\ \forall\, (p_1, r_1) \in \mathrm{All}(I_1),\ (p_2, r_2) \in \mathrm{All}(I_2).\ \mathrm{dist}(p_1, p_2) \geq r_1 + r_2 \end{array}}{I_1 \cup I_2 \models T_1 \mid T_2}$$

where SpaceCheck : $\mathbb{R}^+ \times \mathcal{I} \times \mathcal{I} \to \{tt, ff\}$ is assumed, and All : $\mathcal{I} \to (\mathbb{R}^n \times \mathbb{R}^+)$ is defined as $\mathrm{All}(I) = \bigcup_{(p,r,J) \in I \mid \cdot \neq p} \{(p, r)\} \cup \{(p + p', r') \mid (p', r') \in J \wedge p' \neq \cdot\}$.

The above inference rules allow deriving pairs of the form $I \models T$, where I describes all the elements appearing at top-level in T and, for each top-level looping sequence, all the elements appearing on its brane. In particular, the set I contains pairs $\langle (p, r), \{(p_1, r_1), \ldots, (p_n, r_n)\}\rangle$ where (p, r) describes the spatial information of an element, and the set $\{(p_1, r_1), \ldots, (p_n, r_n)\}$ the spatial information of the elements on its brane (if (p, r) corresponds to a looping sequence, otherwise it is empty). A term T is well-formed iff there exists an I such that $I \models T$.

The function SpaceCheck takes as parameters the radius r of the looping sequence, and the spatial descriptions of the elements inside it (I_2) and of those on the brane (I_1). It can use that information to determine, approximately, if there is enough space for all the elements. For example, the simplest, but the least accurate, definition of the function is the constant function that always returns tt, namely $\mathrm{SpaceCheck}(r, I_1, I_2) = tt$. However, more accurate definitions can be used, if needed.

As for CLS+, we have different kinds of patterns. Now, we also distinguish between brane patterns appearing on the left and on the right part of a rewrite rule. The sets of variables SV, $\mathcal{X}$, BV and TV are assumed as in CLS+, with

[6] Symbol $\mathcal{P}$ denotes the powerset operator.

$\mathcal{V} = SV \cup \mathcal{X} \cup BV \cup TV$. Moreover, we assume a set of position variables PV ranged over by $u, v, \ldots$. We distinguish between the instantiation of variables $\mathcal{V}$ and that of position variables PV. An *instantiation function* for variables in $\mathcal{V}$ is a partial function $\sigma : \mathcal{V} \to \mathcal{T}_{wf} \cup \mathcal{B} \cup \mathcal{S} \cup \mathcal{E}$ that respects the type of variables, while the one for position variables is a partial function $\tau : PV \to \mathcal{D}$. We denote by Σ and $\mathbf{T}$ the sets of all instantiation functions of the two kinds, respectively.

Definition 3.4 [Sequence and brane patterns] *Left brane patterns* BP_L, *right brane patterns* BP_R and *sequence patterns* SP are defined by the following grammar:

$$BP_L ::= \left(SP\right)_u \mid BP_L \mid BP_L \qquad BP_R ::= \left(SP\right)_g \mid BP_R \mid BP_R$$
$$SP ::= \epsilon \mid a \mid SP \cdot SP \mid \widetilde{x} \mid x$$

where $u \in PV$, $g : \mathbf{T} \to \mathcal{D}$. We denote the sets of all left and right brane patterns, and sequence patterns, by $\mathcal{BP}_L$, $\mathcal{BP}_R$ and $\mathcal{S}$, respectively.

Definition 3.5 [Term patterns] *Left patterns* P_L and *right patterns* P_R are given by the following grammar:

$$\begin{array}{lll} P_L & ::= & \left(SP\right)_u \mid \left(BP_{LX}\right)^L_u \rfloor P_{LX} \mid P_L \mid P_L \\ BP_{LX} & ::= & BP_L \mid BP_L \mid \overline{X} \mid \overline{X} \\ P_{LX} & ::= & P_L \mid P_L \mid X \\ P_R & ::= & \lambda \mid \left(SP\right)_g \mid \left(BP_{RX}\right)^L_g \rfloor P_R \mid P_R \mid P_R \mid X \mid \overline{X} \\ BP_{RX} & ::= & BP_R \mid BP_{RX} \mid \overline{X} \mid \overline{X} \end{array}$$

where $u \in PV$, $g : \mathbf{T} \to \mathcal{D}$. We denote the sets of all left patterns by $\mathcal{P}_L$, the set of all right patterns by $\mathcal{P}_R$, and we assume them to be supersets of $\mathcal{BP}_L$ and $\mathcal{BP}_R$, respectively. We denote by $\mathrm{Var}(P)$ the set of all variables appearing in a pattern P, including position variables from PV.

Definition 3.6 A *rewrite rule* is a 4-tuple (f_c, P_L, P_R, k), usually written as

$$[\, f_c \,] \quad P_L \overset{k}{\mapsto} P_R$$

where $f_c : \mathbf{T} \to \{tt, ff\}$, $k \in \mathbb{R}^+$, $\mathrm{Var}(P_R) \subseteq \mathrm{Var}(P_L)$, and each function g appearing in P_R refers only to position variables in $\mathrm{Var}(P_L)$. A rewrite rule where P_L and P_R are brane patterns BP_L and BP_R, respectively, is called *brane (rewrite) rule*.

Rewrite rules are used for modeling the reactions that can occur in the system. Conceptually, a reaction occurs among the elements of the system that match the sequences (simple and looping) appearing in the left pattern. Term and brane variables appearing in the left pattern are used as placeholders for the other elements of a compartment which are not involved in the reaction.

A term can be obtained from a pattern by applying a pair of instantiation functions τ and σ to it. This entails the instantiation of the variables of the pattern and, for right patterns P_R, the replacement of each function $g : \mathbf{T} \to \mathcal{D}$ with the value obtained by applying each of them to τ. A rewrite rule states that, if there exists a pair of instantiation functions τ and σ such that $P_L\tau\sigma$ matches a subterm

of the current system, then that subterm may be rewritten to $P_R\tau\sigma$. The function τ conceptually carries the bindings between position variables if the left pattern of the rule and the actual spatial information (radius and, possibly, position and movement function) of the matched elements. In this way, the g functions are used to compute the spatial information for the elements on the right patterns, using the spatial information of the elements that match with the left pattern.

The rewrite rule, besides left and right patterns, is formed by a function f_c that specifies its *application constraints*, that is whether or not the rule can be applied to specific matching elements. The applicability of the rule is determined by evaluating function f_c over the τ used for the matching. For instance, this function may be used to check the positions of the involved elements, and to allow the reaction only if they are close enough. Moreover, similarly to the variant of CLS called Stochastic CLS [7], rewrite rules are endowed with a *rate* constant $k \in \mathbb{R}$, modeling its propensity of application. In particular, the value $1/k$ represents the expected duration of a reaction involving a precise combination of reactants. For example, a value of $k = 4$ means that each occurrence of the reaction modeled by the rule, on the average, lasts 0.25 time units.

3.1 Semantics

A biological system, described by a term and a set of rewrite rules, evolves by performing a sequence of steps. A step represents the evolution of the system in a finite timespan, and is conceptually composed of two phases: the first, where at most one reaction can occur; the second, where the positions of all the (positional) elements of the system are updated according to their movement functions. The processes we want to describe can be considered to be Poisson processes. This justifies the assumption that at most one reaction occurs at each step, provided that the time length of the step is chosen short enough.

The result of the application of a rewrite rule could be a non well-formed term (according to Definition 3.3), because collisions arise. The same situation can happen after moving the elements. This problem is solved by performing, as last operation of each of the two phases, a "rearrangement" of the elements in the system. The rearrangement is accomplished by the Arrange algorithm (described in Section 3.2), which may fail to determine a well-formed term: if such is the case when trying to apply a rule to a group of elements, then it is assumed that the considered group of elements cannot react (so the rule cannot be applied). If the rearrangement fails after the phase of movement, then for the current step the elements are not moved.

Before defining the semantics, we need to introduce some auxiliary definitions. Let $Move : \mathcal{T}_{wf} \times \mathbb{R}^+ \times \mathbb{R}^+ \to \mathcal{T}_{wf}$ be the following function:

$$Move(T, t, \delta t) = \begin{cases} T'' & \text{if } \exists T', T''.\ \langle T, t, \delta t\rangle \xrightarrow{in\ \infty}_{\text{mov}} T' \wedge T'' = \text{Arrange}(T') \neq \perp; \\ T & \text{otherwise;} \end{cases}$$

where relation $\xrightarrow{x\ l}_{\text{mov}}$, with $x \in \{in, on\}$ and $l \in \mathbb{R}^+$, is the least labeled transition relation given by the inference rules shown in Figure 1. $Move$ is used to perform the movement phase, and gives a new term, obtained from the current state T by

$$\frac{x \in \{in, on\} \qquad l \in \mathbb{R}^+}{\langle \lambda, t, \delta t\rangle \xrightarrow{x\ l}_{\mathrm{mov}} \lambda} \qquad \frac{x \in \{in, on\} \qquad l \in \mathbb{R}^+}{\langle (S)_{\cdot,r}, t, \delta t\rangle \xrightarrow{x\ l}_{\mathrm{mov}} (S)_{\cdot,r}}$$

$$\frac{p' = m_{\mathrm{fun}}(p, r, x\ l, t, \delta t) \qquad x \in \{in, on\} \qquad l \in \mathbb{R}^+}{\langle (S)_{[p,m],r}, t, \delta t\rangle \xrightarrow{x\ l}_{\mathrm{mov}} (S)_{[p',m],r}}$$

$$\frac{\begin{array}{c}\langle B_1, t, \delta t\rangle \xrightarrow{on\ r}_{\mathrm{mov}} B_2 \qquad \langle T_1, t, \delta t\rangle \xrightarrow{in\ r}_{\mathrm{mov}} T_2 \\ p' = m_{\mathrm{fun}}(p, r, x\ l, t, \delta t) \qquad x \in \{in, on\} \qquad l \in \mathbb{R}^+\end{array}}{\langle (B_1)^L_{[p,m],r} \rfloor T_1, t, \delta t\rangle \xrightarrow{x\ l}_{\mathrm{mov}} (B_2)^L_{[p',m],r} \rfloor T_2}$$

$$\frac{\langle B_1, t, \delta t\rangle \xrightarrow{on\ r}_{\mathrm{mov}} B_2 \qquad \langle T_1, t, \delta t\rangle \xrightarrow{in\ r}_{\mathrm{mov}} T_2 \qquad x \in \{in, on\} \qquad l \in \mathbb{R}^+}{\langle (B_1)^L_{\cdot,r} \rfloor T_1, t, \delta t\rangle \xrightarrow{x\ l}_{\mathrm{mov}} (B_2)^L_{\cdot,r} \rfloor T_2}$$

$$\frac{\langle T_1, t, \delta t\rangle \xrightarrow{x\ l}_{\mathrm{mov}} T_1' \qquad \langle T_2, t, \delta t\rangle \xrightarrow{x\ l}_{\mathrm{mov}} T_2'}{\langle T_1 \mid T_2, t, \delta t\rangle \xrightarrow{x\ l}_{\mathrm{mov}} T_1' \mid T_2'}$$

Fig. 1. Rules of the transition relation $\rightarrow_{\mathrm{mov}}$.

updating the positions of all the (positional) elements to the positions reached after δt time units from time t.

The rate of a rule application is calculated with the help of the functions *comb*, *comb*$'$ and *binom*. In the following definition, we denote the multiset of top-level elements appearing in a pattern (or term) P by $\overline{P}$, and assume the function $\mathbf{n} : \mathcal{T} \times \mathcal{T} \to \mathbb{N}$ that, applied to a term T_1, representing a (simple or looping) sequence, and another term T_2, gives the number of times the T_1 appears at top-level in T_2.

Let $comb, comb' : \mathcal{P}_L \times \mathbf{T} \times \Sigma \to \mathbb{N}$ and $binom : \mathcal{T} \times \mathcal{T} \times \mathcal{T} \to \mathbb{Q}$ be recursively defined as follows:

$$\begin{aligned}
comb(P_{L1} \mid P_{L2}, \tau, \sigma) &= comb(P_{L1}, \tau, \sigma) \cdot comb(P_{L2}, \tau, \sigma) \\
comb((BP_{LX})^L_u \rfloor P_{LX}, \tau, \sigma) &= comb'(BP_{LX}, \tau, \sigma) \cdot comb'(P_{LX}, \tau, \sigma) \\
comb((SP)_u, \tau, \sigma) &= 1 \\
comb'(P_L \mid U, \tau, \sigma) &= \prod_{T \in \overline{P_L \tau \sigma}} \binom{\mathbf{n}((P_L|U)\tau\sigma, T)}{\mathbf{n}(P_L\tau\sigma, T)} \cdot comb(P_L, \tau, \sigma) \qquad U \in BV \cup TV \\
comb'(P_L, \tau, \sigma) &= comb(P_L, \tau, \sigma) \\
binom(T_1, T_2, T_3) &= \prod_{T \in \overline{T_1}} \prod_{i=1}^{\mathbf{n}(T_3, T)} \frac{\mathbf{n}(T_2, T) + i}{\mathbf{n}(T_2, T) - \mathbf{n}(T_1, T) + i}
\end{aligned}$$

Given a finite set of rewrite rules $\mathcal{R}$, let $\mathcal{R}_{\mathcal{B}} \subseteq \mathcal{R}$ be the set of all brane rules in $\mathcal{R}$ and let $\xrightarrow{R,T,c}$, with $R \in \mathcal{R}, T \in \mathcal{T}$ and $c \in \mathbb{N}$, be the least labeled transition relation on terms closed with respect to $\equiv_T$ and satisfying the inference rules shown in Figure 2. These definitions follows closely that of Stochastic CLS given in [7], to which we refer for more details.

The defined transition relation is used to determine each group of elements to which a rewrite rule can be applied, without considering the subsequent rearrangement. Each transition $T_1 \xrightarrow{R,T,c} T_2$ describes the application of rewrite rule R inside

$$\frac{\left(R : [\, f_c \,] \quad P_L \overset{k}{\mapsto} P_R\right) \in \mathcal{R} \qquad f_c(\tau) = tt \qquad \tau \in \mathbf{T} \qquad \sigma \in \Sigma}{P_L\tau\sigma \xrightarrow{R,P_L\tau\sigma,comb(P_L,\tau,\sigma)} P_R\tau\sigma}$$

$$\frac{B \xrightarrow{R,T,c} B' \qquad R \in \mathcal{R}_{\mathcal{B}}}{\left(B\right)^L_d \rfloor T_1 \xrightarrow{R,(B)^L_d \rfloor T_1,c} \left(B'\right)^L_d \rfloor T_1} \qquad \frac{T_1 \xrightarrow{R,T,c} T_1'}{\left(B\right)^L_d \rfloor T_1 \xrightarrow{R,(B)^L_d \rfloor T_1,c} \left(B\right)^L_d \rfloor T_1'}$$

$$\frac{T_1 \xrightarrow{R,T,c} T_1'}{T_1 \mid T_2 \xrightarrow{R,T,c\cdot binom(T,T_1,T_2)} T_1' \mid T_2}$$

Fig. 2. The inference rules used for computing the rate of a rule application.

term T_1 yielding to T_2. The value $c \in \mathbb{N}$ corresponds to the number of different reactant combinations among which the reaction described by R may, conceptually, occur. This is needed because, for example, if we consider a rewrite rule involving non-positional elements, such as $\left(a\right)_{\cdot,0} \mid \left(b\right)_{\cdot,0} \overset{k}{\mapsto} \left(a \cdot b\right)_{\cdot,0}$, then the reaction can conceptually occur between each pair of elements $\left(a\right)_{\cdot,0}$ and $\left(b\right)_{\cdot,0}$ contained in a compartment. Nevertheless, all of them yield to a same term, obtained by replacing one $\left(a\right)_{\cdot,0}$ and one $\left(b\right)_{\cdot,0}$ with $\left(a \cdot b\right)_{\cdot,0}$. Thus, in this case, the value c is the number of pairs of elements that can react, which is equal to the number of a's times the number of b's in the compartment.

The following definition gives all the reactions enabled in a state, by also taking into account the subsequent rearrangement:

$$Appl(R,T) = \{(T_r, c, T', T'') \mid T \xrightarrow{R,T_r,c} T' \wedge T'' \equiv \text{Arrange}(T') \neq \perp\}$$

Given a rewrite rule R and a term T_1, function *Appl* computes a set of tuples of the form (T_r, c, T', T''), where T_r identifies the subterm of T_1 to which R is applied (i.e. the reactants), T' is the term transformed by the application excluding the rearrangement, and T'' is the rearrangement of T'. We consider two reactions to be different if they involve different reactants T_r or different resulting terms T''. Finally, the number $m_T^{(R)}$ of different reactant combinations enabled in state T, for a reaction R, and the total number m_T of reactions considering a set of rules $\mathcal{R}$, are defined as:

$$m_T^{(R)} = \sum_{(T_r,c,T',T'') \in Appl(R,T)} c \qquad\qquad m_T = \sum_{R \in \mathcal{R}} m_T^{(R)}.$$

Let T describe the state of the system at a certain step, and k_R denote the rate associated with a rewrite rule R. At each step of the evolution of the system, in order to assume that at most one reaction can occur, we have to choose a time interval δt such that $\left(\sum_{R \in \mathcal{R}} k_R \, m_T^{(R)}\right) \, \delta t \leq 1$. Given a set of rewrite rules $\mathcal{R}$, we choose an arbitrary value N such that for each rule $R \in \mathcal{R}$ it holds $0 < k_R/N \leq 1$. Then we compute the time interval for a step as $\delta t = \frac{1/N}{m_T}$, thus satisfying the above condition. The value of N also determines the maximum permitted length of each step as $1/N$ time units. Its choice may indirectly affect the precision in representing movement.

Finally, the probability that no reaction happens in the time interval δt is:

$$\overline{p}_T = 1 - \sum_{R \in \mathcal{R}} \left(\sum_{(T_r, c, T', T'') \in Appl(R,T)} \frac{k_R/N}{m_T} \cdot c \right),$$

and the probability $P(T_1 \to T_2, t)$ of reaching state T_2 from T_1 after a time interval $\delta t = \frac{1/N}{m_{T_1}}$ by t is such that:

$$P(T_1 \to T_2', t) = \sum_{R \in \mathcal{R}} \left(\sum_{\substack{(T_r, c, T', T_2) \in Appl(R,T_1) \text{ s.t.} \\ T_2' \equiv Move(T_2, t, \delta t)}} \frac{k_R/N}{m_{T_1}} \cdot c \right) + \begin{cases} \overline{p}_{T_1} & \text{if } T_1 \equiv T_2'; \\ 0 & \text{otherwise.} \end{cases}$$

The semantics, given as a *Probabilistic Transition System*, is defined as follows.

Definition 3.7 [Semantics] Given a finite set of rewrite rules $\mathcal{R}$, the semantics of Spatial CLS is the least relation satisfying the following inference rules:

$$\frac{(T_r, c, T', T_2) \in Appl(R, T_1) \qquad R \in \mathcal{R} \qquad T_1 \not\equiv T_2' \qquad p = P(T_1 \to T_2', t) \qquad T_2' \equiv Move(T_2, t, \delta t) \qquad \delta t = \frac{1/N}{m_{T_1}}}{\langle T_1, t \rangle \xrightarrow{p} \langle T_2', t + \delta t \rangle}$$

$$\frac{p = P(T \to T', t) \qquad T' \equiv Move(T, t, \delta t) \qquad \delta t = \frac{1/N}{\max\{1, m_T\}}}{\langle T, t \rangle \xrightarrow{p} \langle T', t + \delta t \rangle}$$

The states of the transition system are of the form $\langle T, t \rangle$, representing the system at time t. Each transition $\langle T_1, t \rangle \xrightarrow{p} \langle T_2, t + \delta t \rangle$ describes all the reactions, in the current step, that may lead to T_2 from T_1, also by considering the rearrangements of elements done after each phase. The first inference rule is used to derive transitions which yield a term T_2' different from T_1, while the second is used to derive the transition representing the case in which no reaction occurs. In the second case, the function P computes a probability which takes into account those reactions that do not change the state of the system.

3.2 Resolving space conflicts

Space conflicts that may arise during the evolution of the system are resolved by the algorithm Arrange, which simulates the movement of the elements as if they push each other. The problem is modeled by assuming an instant velocity associated with each element, whose direction and speed depend on the instant position of every element.

Given an element, the velocity it is subjected to is calculated as the sum of other velocities:

- for each other element it overlaps with, we assume a velocity directed opposite to the other element (the elements are trying to increase their distance) and whose magnitude is proportional to the length of the overlap;

- if the element is not completely within the bounds of the containing membrane, then we assume a velocity directed towards the center of the membrane, whose magnitude is inversely proportional to the distance from the center;
- if the element is on the surface of a membrane, but its center is not located on the surface of the sphere, then we assume a velocity directed towards the nearest point of the sphere with a magnitude proportional to the distance from the sphere.

If any of those velocities are not well-defined (for instance, if the centers of two elements coincide), then we assume an arbitrary fixed direction along which the elements move.

This behaviour is modeled by a system of differential equations. Given a term T, let $\boldsymbol{x_1}, \ldots, \boldsymbol{x_n}$ be the variables for the centers of the positional elements appearing in T, and I_L and I_S the set of indices denoting looping and simple sequences, respectively ($I_L \cup I_S = \{1, \ldots, n\}$). Moreover, let $\mathrm{In}(i)$, with $i \in I_L$, be the set of indices denoting elements inside the looping sequence i, or, if $i = 0$, the indices of the top–level elements; let $\mathrm{On}(i)$, with $i \in I_L$, denote the elements appearing on the surface of i; and $\mathrm{Inner}(i) = \mathrm{In}(i) \cup \bigcup_{j \in \mathrm{In}(i)} \mathrm{On}(j)$.

The system of differential equations used by Arrange is the following, where h_i is such that $i \in \mathrm{Inner}(h_i)$:

$$\begin{cases} \dfrac{\mathrm{d}\boldsymbol{x_i}}{\mathrm{d}t} = \left(\displaystyle\sum_{j \in \mathrm{Inner}(h_i) \setminus \{i,k\} \cup \mathrm{In}(k)} \boldsymbol{v_{ij}} \right) - \boldsymbol{u_i} + \boldsymbol{w_i} & \forall i.\ \exists k.\ i \in \mathrm{On}(k) \\ \dfrac{\mathrm{d}\boldsymbol{x_i}}{\mathrm{d}t} = \left(\displaystyle\sum_{j \in \mathrm{Inner}(h_i) \setminus (\{i\} \cup \mathrm{On}(i))} \boldsymbol{v_{ij}} \right) - \boldsymbol{u_i} & \forall i \in \mathrm{In}(h_i) \end{cases}$$

and where the following definitions are used ($\boldsymbol{a}$, $\boldsymbol{b}$, $\boldsymbol{c}$ are non-null vectors specifying directions used when velocities cannot be calculated) [7]:

$$\boldsymbol{y_i} = \begin{cases} \boldsymbol{x_i} + \boldsymbol{y_j} & \text{if } \exists j \neq 0.\ i \in \mathrm{On}(j) \vee i \in \mathrm{In}(j) \\ \boldsymbol{x_i} & \text{otherwise} \end{cases}$$

$$\boldsymbol{v_{ij}} = \begin{cases} v_{ij}\, \widehat{(\boldsymbol{y_i} - \boldsymbol{y_j})} & \text{if } \boldsymbol{y_i} \neq \boldsymbol{y_j} \\ +v_{ij}\, \boldsymbol{a} & \text{if } \boldsymbol{y_i} = \boldsymbol{y_j} \text{ and } i < j \\ -v_{ij}\, \boldsymbol{a} & \text{if } \boldsymbol{y_i} = \boldsymbol{y_j} \text{ and } i \geq j \end{cases} \qquad v_{ij} = \max\{0, r_i + r_j - \|\boldsymbol{y_i} - \boldsymbol{y_j}\|\}$$

$$\boldsymbol{u_i} = \begin{cases} u_i\, \widehat{\boldsymbol{y_i}} & \text{if } \boldsymbol{y_i} \neq \boldsymbol{0} \\ u_i\, \boldsymbol{b} & \text{if } \boldsymbol{y_i} = \boldsymbol{0} \end{cases} \qquad u_i = \begin{cases} \max\{0, r_i + \|\boldsymbol{y_i}\| - r_h\} & \text{if } \exists h \neq 0.\ i \in \mathrm{Inner}(h) \\ 0 & \text{otherwise} \end{cases}$$

$$\boldsymbol{w_i} = \begin{cases} w_i\, \widehat{\boldsymbol{x_i}} & \text{if } \boldsymbol{x_i} \neq \boldsymbol{0} \\ w_i\, \boldsymbol{c} & \text{if } \boldsymbol{x_i} = \boldsymbol{0} \end{cases} \qquad w_i = r_k - \|\boldsymbol{x_i}\|$$

The Arrange algorithm stops when a stable setting is reached. If the term representing this setting is not well-formed, the algorithm returns the special value $\perp$, otherwise it returns a term where the positions of all the elements are updated. Note

[7] $\widehat{\boldsymbol{x}}$ denotes the normalized vector $\boldsymbol{x}$, i.e. $\widehat{\boldsymbol{x}} = \boldsymbol{x}/\|\boldsymbol{x}\|$ if $\boldsymbol{x} \neq \boldsymbol{0}$.

that, even if all space conflicts are resolved, still the term may not be well-formed because there is not enough space for the non-positional elements, as determined by the function SpaceCheck.

The use of this mechanism is not the only solution for dealing with space conflicts among the elements of a system. We could represent the elements as points, which means that they do not occupy any space. However, this assumption would be reasonable only if the elements are small with respect to the environment, and this is not always the case in biological systems. Another option would be allowing the application of a rule only if it would not create a conflict. Unfortunately, this option would lessen the modeling capacity of the calculus; for example, it would not be possible to model a membrane that, increasing its size, pushes contiguous elements.

A more physically sound modeling of the behaviour of the system in case of space conflicts, could be obtained by extending the calculus with the association of a "weight" and consequently a "pushing strength" to elements. The Arrange algorithm could take such weigths into account, so that an element could push other elements according to its strength.

4 Examples of modeling

In this section we show two examples of Spatial CLS models.

A model of cell proliferation

We show the development of a biological tissue. We represent the way in which a cell performs the mitosis cycle, and the development of the tissue as a consequence of it. The structure of a cell is made up of a permeable cell membrane, which separates it from external environment but still allows messages to pass through, and contains several organelles scattered in the cytoplasm. We model a simple eukaryotic cells as a membrane containing the nucleus, which, in turn, contains two DNA molecules (the chromosomes). Each cell performs the *cell cycle* [6], that is the sequence of phases that lead to its division into two daughter cells, structurally alike to the mother cell. Customarily, cell cycle repeats for every generated cell, but, in particular cases, the cell may decide to stop the process in a permanent or temporary way (for instance, in case of unfavourable ambient conditions).

The initial state of the biological system is described by the following term [8]:

$$T = \left(b\right)^{L}_{\cdot,50} \rfloor \left(m\right)^{L}_{[(0,0),m_1],10} \rfloor \left(n\right)^{L} \rfloor \left(cr \cdot g_1 \cdot g_2 \cdot g_3 \mid cr \cdot g_4 \cdot g_5\right)$$

The term contains the looping sequence $\left(b\right)^{L}_{\cdot,50}$, representing the space available for the proliferation as a circle with a $50\mu m$ radius. It contains a single cell $\left(m\right)^{L}$, positioned in $(0,0)$ and with a radius of $10\mu m$. The cell is subjected to a small Brownian motion, modeled by m_1. The nucleus, represented as $\left(n\right)^{L}$, and the contained chromosomes, are represented as non-positional elements. The nucleus may be in two states, depending on the symbol appearing on its looping sequence.

[8] For the sake of clarity, we omit the spatial information for non-positional elements, i.e. those elements $\left(\_\right)_{\langle q,r\rangle}$ such that $\langle q,r\rangle = \langle \cdot, 0\rangle$.

Initially, the nucleus is identified by the symbol n appearing on the surface of its membrane. During the evolution of the system, the symbol n is replaced by n_{dup}, indicating a state in which the nucleus has started the duplication process and is about to divide. Chromosomes are modeled as sequences starting with cr symbol, followed by the genes, represented by the symbols g_i's. Instead, a duplicated chromosome is identified by having $2cr$ as its first symbol in the sequence.

The Brownian motion of cells represents the small movements the cells are subjected to in a biological tissue. The use of Spatial CLS to represent the cell cycle allows showing the spatial arrangement of cells during the tissue development. This is an important point in many biological tissue, see for example [13] for a spatial mathematical modeling of a neural tissue.

The evolution of the system is modeled by the following rewrite rules [9]:

$$
\begin{aligned}
&R_1 : [\, r = 7 \,] \quad \left(m\right)^L_{[p,f],r} \rfloor X \overset{0.33}{\mapsto} \left(m\right)^L_{[p,f],10} \rfloor X \\
&R_2 : [\, r = 10 \,] \quad \left(m\right)^L_{[p,f],r} \rfloor X \overset{0.25}{\mapsto} \left(m\right)^L_{[p,f],14} \rfloor X \\
&R_3 : [\, r = 14 \,] \quad \left(m\right)^L_{u} \rfloor \left(\left(n\right)^L \rfloor X\right) \overset{0.5}{\mapsto} \left(m\right)^L_{u} \rfloor \left(\left(n_{\mathrm{dup}}\right)^L \rfloor X\right) \\
&R_4 : \left(n_{\mathrm{dup}}\right)^L \rfloor \left(cr \cdot \widetilde{x} \mid X\right) \overset{0.125}{\mapsto} \left(n_{\mathrm{dup}}\right)^L \rfloor \left(2cr \cdot \widetilde{x} \mid X\right) \\
&R_5 : \left(n_{\mathrm{dup}}\right)^L \rfloor \left(2cr \cdot \widetilde{x} \mid 2cr \cdot \widetilde{y}\right) \overset{0.17}{\mapsto} \left(n\right)^L \rfloor \left(cr \cdot \widetilde{x} \mid cr \cdot \widetilde{y}\right) \mid \left(n\right)^L \rfloor \left(cr \cdot \widetilde{x} \mid cr \cdot \widetilde{y}\right) \\
&R_6 : \left(m\right)^L_{[(x,y),f],r} \rfloor \left(\left(n\right)^L \rfloor X \mid \left(n\right)^L \rfloor Y\right) \overset{1}{\mapsto} \\
&\qquad \left(m\right)^L_{[(x-5,y),f],7} \rfloor \left(n\right)^L \rfloor X \mid \left(m\right)^L_{[(x+5,y),f],7} \rfloor \left(n\right)^L \rfloor Y
\end{aligned}
$$

The first three rules describe the growth of the cell. Rule R_1 increases the radius from 7 to 10, leading a just-splitted cell to its normal size. Rule R_2 represents the starting of the division process, where the cell grows to 14 and will eventually divide. A cell may block its cell cycle if there is not enough space: this happens when neither R_1 nor R_2 are applicable to it. The application of rule R_3 signals the start of the division process for the nucleus. Rule R_4 models the duplication of a single chromosome. Finally, rule R_5 and R_6 describe the division of the nucleus and the subsequent cellular division. In rule R_6, the splitted cells are arbitrarily positioned one next to the other, near the position of the parent cell. The rates have been estimated according to the common relative lengths of the phases forming the cell cycle, and so as to obtain, on the average, a duration of 24 hours for the complete cycle ([6]).

Figure 3 shows the state of the system at certain times during the simulation, obtained by an ad hoc simulator. At time $t = 0$ the system contains only one cell, positioned inside the limiting membrane. The proliferation stops at time $t = 141h$, when the space left is not enough for any cell to grow. We can also see that, from time $102h$ to $108h$, a cell near the center has splitted into two small cells, and another cell on the bottom right has grown, thus initiating the division process. By growing, the cell pushed the surrounding cells and caused the rearrangement.

[9] For the sake of readability, we use a simpler syntax for writing rewrite rules: in the left part of the rules, we use placeholders for positions, movement functions and radii, and reuse them in the right part in a shorthand notation for defining instantiation functions for position variables.

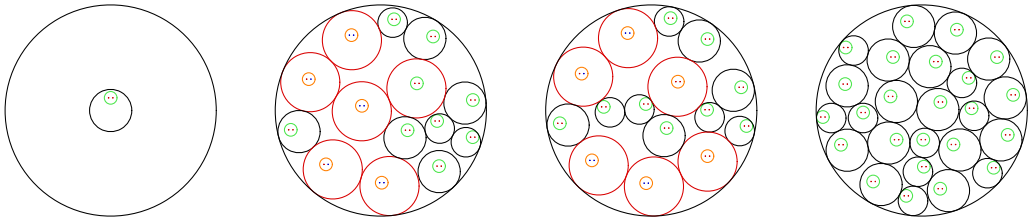

Fig. 3. The graphical representation of the system at times 0, $102h$, $108h$ and $141h$ during simulation.

A model of the quorum sensing process

Many bacteria have the ability of monitoring their population density and modulating their gene expressions according to this density. This process is called *quorum sensing*, and the main entities involved in it are the *autoinducers*, small molecules that can cross the cellular membrane and can diffuse freely either out or in bacteria, and the *R-proteins*, transcriptional activator proteins located within the cell.

At low cell density, the autoinducer is synthesized at basal levels and diffuse in the environment where it is diluted. With high cell density both the extracellular and intracellular concentrations of the autoinducer increase until they reach thresholds beyond which the autoinducer is produced autocatalytically. This autocatalytic production results in a dramatic increase of its concentration.

We show a simple model of the quorum sensing process in *Pseudomonas aeruginosa* (see [12] for a more detailed description of the phenomenon). Such a bacterium uses quorum sensing to keep low the expression of virulence factors until the colony has reached a certain density, when an autoinduced production of virulence factors is started. The initial state of each bacterium is:

$$Bact_i = \big(m\big)^L_{[p_i,m_0],r_B} \rfloor (lasO \cdot lasR \cdot lasI)$$

where the bacterium membrane, denoted $\big(m\big)^L$, contains only a DNA strand. Bacteria do not move, hence their movement function is m_0, and their radii are set to r_B. The DNA is modeled as a sequence of genes $lasO \cdot lasR \cdot lasI$, where $lasO$ represents the target to which a complex autoinducer/R-protein binds to promote transcription. The following rewrite rules model the system behaviour.

$$R_1 : lasO \cdot lasR \cdot lasI \overset{k_1}{\mapsto} lasO \cdot lasR \cdot lasI \mid 3oxo$$

$$R_2 : \big(m\big)^L_u \rfloor (3oxo \mid X) \overset{k_2}{\mapsto} \big(m\big)^L_u \rfloor (3R \mid X)$$

$$R_3 : \big(m\big)^L_u \rfloor (3R \mid X) \overset{k_3}{\mapsto} \big(m\big)^L_u \rfloor (3oxo \mid X)$$

$$R_4 : 3R \mid lasO \cdot lasR \cdot lasI \overset{k_4}{\mapsto} 3RO \cdot lasR \cdot lasI$$

$$R_5 : 3RO \cdot lasR \cdot lasI \overset{k_5}{\mapsto} 3R \mid lasO \cdot lasR \cdot lasI$$

$$R_6 : 3RO \cdot lasR \cdot lasI \overset{k_6}{\mapsto} 3RO \cdot lasR \cdot lasI \mid 3oxo$$

$$R_7 : \big(m\big)^L_{[(x,y),f],r} \rfloor (3oxo \mid X) \overset{k_7}{\mapsto} \big(3oxo\big)_{[(x+r,y+r),m_B],r_{3oxo}} \mid \big(m\big)^L_{[(x,y),f],r} \rfloor X$$

$$R_8 : [\,\mathrm{dist}(p_1,p_2) \leq r_2 + 10\,] \quad \big(3oxo\big)_{[p_1,m_B],r_{3oxo}} \mid \big(m\big)^L_{[p_2,f],r_2} \rfloor X \overset{k_8}{\mapsto} \big(m\big)^L_{[p_2,f],r_2} \rfloor (3oxo \mid X)$$

$$R_9 : \big(3oxo\big)_u \overset{k_9}{\mapsto} \lambda$$

Rule R_1 describes the basal production of the autoinducer, modeled as $3oxo$. The

value k_1 is chosen according to its basal rate of production. Rules R_2 and R_3 describe the formation of the complex autoinducer/R-protein, modeled as $3R$, and its decomplexation, respectively. To keep the model simple, we do not describe explicitly the R-protein. The next two rules, R_4 and R_5, model the binding and unbinding of the $3R$ complex to gene $lasO$ of the DNA. Rule R_6 describes the increased rate of production of the autoinducer, due to the binding of the complex $3R$ to the DNA. The promoted production of $3oxo$ can be modeled using a rate k_6 much higher than the basal rate k_1 used in rule R_1. Rules R_7 and R_8 describe the ability of the autoinducer to cross the membrane, in both directions. An autoinducer inside the bacterium is modeled as a non-positional element, while autoinducers outside have an associated position, radius r_{3oxo} and movement function m_B. The movement function m_B models Brownian motion, which describes the diffusion of the autoinducer in the environment. Finally, rule R_9 models the degradation of the autoinducer, which can happen both inside and outside the bacterium.

This Spatial CLS model of the quorum sensing process is more accurate than other stochastic models (such as the Stochastic CLS model given in [7]). In fact, other stochastic models are usually based on the assumption that biological entities are homogeneously distributed in the environment, and in a quorum sensing process this is not true for the autoinducer proteins outside the bacteria. Note that taking into account the spatial diffusion has a particular significance when reactions are comparatively faster than diffusion rates [26].

Differently with respect to the cell proliferation example, in this case we have not developed an ad hoc simulator, for which the values of reaction constants should be given. We leave the development of a general simulator for Spatial CLS models as future work.

5 Conclusions

We have presented the Spatial Calculus of Looping Sequences (Spatial CLS), which extends the Calculus of Looping Sequences (CLS) by allowing spatial information to be associated with structures. The Spatial CLS formalism enables the accurate description of those biological processes whose behaviour depends on the exact position of the elements. As example applications, we have presented a model of cell proliferation and a model of the quorum sensing process in *Pseudomonas aeruginosa*. An ad hoc simulator has been developed for the model of cell proliferation. We leave as future work the development of a general simulator for Spatial CLS models.

References

[1] *SCLSm: The Stochastic CLS machine (web page).*
URL http://www.di.unipi.it/~milazzo/biosims/

[2] *The BioSPI Project.*
URL http://www.wisdom.weizmann.ac.il/~biospi/

[3] *The Stochastic Pi Machine (SPiM).*
URL http://research.microsoft.com/~aphillip/spim/

[4] Alur, R., C. Belta, F. Ivančić, V. Kumar, M. Mintz, G. J. Pappas, H. Rubin and J. Schug, *Hybrid Modeling and Simulation of Biomolecular Networks*, LNCS **2034** (2001), pp. 19–32.

[5] Andrews, S. S. and D. Bray, *Stochastic simulation of chemical reactions with spatial resolution and single molecule detail*, Physical Biology **1** (2004), pp. 137–151.

[6] Avers, C. J., "Molecular Cell Biology," Addison-Wesley, 1986.

[7] Barbuti, R., G. Caravagna, A. Maggiolo-Schettini, P. Milazzo and G. Pardini, *The Calculus of Looping Sequences*, LNCS **3016** (2008), pp. 387–423.

[8] Barbuti, R., A. Maggiolo-Schettini, P. Milazzo and A. Troina, *A Calculus of Looping Sequences for Modelling Microbiological Systems*, Fundamenta Informaticae **72** (2006), pp. 21–35.

[9] Cardelli, L., *Brane Calculi. Interactions of Biological Membranes*, LNCS **3082** (2005), pp. 257–280.

[10] Curti, M., P. Degano, C. Priami and C. T. Baldari, *Modelling biochemical pathways through enhanced π-calculus*, Theor. Comput. Sci. **325** (2004), pp. 111–140.

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

[12] Dockery, J. and J. Keener, *A mathematical model for quorum sensing in pseudomonas aeruginosa*, Bulletin of Mathematical Biology **63** (2001), pp. 95–116.

[13] Galli-Resta, L., E. Novelli, Z. Kryger, G. H. Jacobs and B. E. Reese, *Modelling the mosaic organization of rod and cone photoreceptors with a minimal-spacing rule*, European Journal of Neuroscience **11** (1999), pp. 1461–1469.

[14] Giavitto, J.-L. and O. Michel, *MGS: a rule-based programming language for complex objects and collections*, Electronic Notes in Theoretical Computer Science **59** (2001).

[15] John, M., R. Ewald and A. M. Uhrmacher, *A spatial extension to the π-Calculus*, ENTCS **194** (2008), pp. 133–148.

[16] Krivine, J., R. Milner and A. Troina, *Stochastic bigraphs*, in: *Proc. of MFPS'08, 24th Conference on the Mathematical Foundations of Programming Semantics*, ENTCS **to appear** (2008).

[17] Kwiatkowska, M., G. Norman and D. Parker, *Probabilistic Symbolic Model Checking with PRISM: A Hybrid Approach*, International Journal on Software Tools for Technology Transfer (STTT) **6** (2004), pp. 128–142.

[18] Matsuno, H., A. Doi, M. Nagasaki and S. Miyano, *Hybrid Petri Net Representation of Gene Regulatory Network*, Pacific Symposium on Biocomputing (2000), pp. 341–352.

[19] Milazzo, P., "Qualitative and Quantitative Formal Modeling of Biological Systems," Ph.D. thesis, Università di Pisa (2007).

[20] Paun, G., "Membrane Computing: An Introduction," Springer, New York, 2002.

[21] Plimpton, S. J. and A. Slepoy, *Microbial cell modeling via reacting diffusive particles*, Journal of Physics: Conference Series **16** (2005), pp. 305–309.

[22] Priami, C. and P. Quaglia, *Beta Binders for Biological Interactions*, LNCS **3082** (2005), pp. 20–33.

[23] Priami, C., A. Regev, E. Shapiro and W. Silverman, *Application of a stochastic name-passing calculus to representation and simulation of molecular processes*, Inform. Proc. Letters **80** (2001), pp. 25–31.

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

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

[26] Takahashi, K., S. N. Arjunan and M. Tomita, *Space in systems biology of signaling pathways – towards intracellular molecular crowding in silico*, FEBS Letters (2005).

[27] Tolle, D. P. and N. L. Novère, *Particle-Based Stochastic Simulation in Systems Biology*, Current Bioinformatics (2006).

[28] Versari, C., *A core calculus for a comparative analysis of bio-inspired calculi*, in: *European Symposium on Programming*, LNCS **4421** (2007), pp. 411–425.