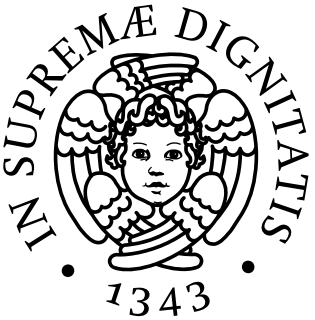
Università di Pisa



Facoltà di Scienze Matematiche Fisiche e Naturali
Corso di Laurea specialistica in Informatica

Tesi di laurea

Topological Calculus of Looping Sequences

Giovanni Pardini

Relatori
Prof. Roberto Barbuti
Dott. Paolo Milazzo

Controrelatore
Prof. Francesca Levi

Anno Accademico 2006/2007

Riassunto

Il Calculus of Looping Sequences (CLS) permette la descrizione dei sistemi biologici e della loro evoluzione. Nell'ambito del lavoro di tesi è stata sviluppata una estensione del CLS, chiamata Topological CLS (TCLS), dove ad ogni oggetto del sistema biologico sono associate una precisa posizione e dimensioni nello spazio. Gli oggetti possono muoversi autonomamente e le loro posizioni determinano se una specifica reazione può avvenire, cioè se la regola di riscrittura che modella tale reazione può essere applicata. Inoltre, alle regole di riscrittura è associato un parametro che ne specifica la velocità di reazione. Infine, dato che gli oggetti occupano un preciso spazio, possono venirsi a creare dei conflitti, che sono risolti assumendo un meccanismo di spinte tra di essi.

Il Topological CLS è stato quindi utilizzato per modellare il ciclo cellulare e simulare la proliferazione delle cellule in uno spazio limitato.

Summary

The Calculus of Looping Sequences (CLS) enables the description of biological systems and their evolution. The work for the thesis has developed an extension to CLS, called Topological CLS (TCLS), where each object of the biological system has associated a definite position and dimensions in space. Objects are able to move autonomously, and their positions determine whether or not a specific reaction could happen, i.e. whether or not the rewrite rule that models that reaction can be applied. Also, each rewrite rule has associated with it a parameter specifying its reaction rate. Conflicts that may arise between objects, as they occupy a non-null space, are resolved by assuming that they push each other when they become be too close.

The Topological CLS has been used for modelling cell cycle and simulating cell proliferation in a limited space.

Contents

Chapter 1

Introduction

Traditionally, the study of biological systems has involved the development of mathematical models, often based on differential equations, which permit the description and analysis of their behaviour. The use of abstract models has many advantages such as, for example, it allows the development of simulators which, in turn, may reduce the need for laboratory experiments.

However, as the complexity of the system increases, mathematical models become more difficult both in the specification and in the analysis [30]. Moreover, they may not be well-suited for modelling particular systems: for example, using differential equations for modelling the variation in concentrations of reactants in a solution is sufficiently accurate only when the number of reactants is high, whereas becomes less accurate when their number is low. This is because the number of reactants is abstracted as a continuous variable.

Recently, new approaches for the modelling of biological processes have been proposed, which involve the use of modelling formalisms coming from Computer Science. Many formalism developed for modelling and analysing interactive and communicating systems have been applied to Biology [24], as they allow for a finer specification of their behaviour than traditional models. Among these, there are: the π-*calculus* [13, 31, 27, 26], automata-based models [3, 18] and rewrite systems [14, 23] In particular, these formalisms have the ability to overcome some of the limitations of traditional models: for example, they allow to represent faithfully even systems comprising a small number of components, and among which complex interactions take place. At the same time, new formalisms specifically developed

for modelling biological systems have emerged [11, 26, 29]. Their uses comprise:

- the development of simulators, which allow biologists to test rapidly new hypoteses about mechanisms underlying biological processes (e.g.: SPiM [2], BioSPI [1]);
- the analysis of the systems and the verification of their properties by using formal means of Computer Science (for example, probabilistic model checking [17]).

Among the formalisms expressly developed for modelling microbiological processes is the *Calculus of Looping Sequences* (CLS) [9, 10, 19]. CLS is based on term rewriting and, as such, a model is composed of a term and a set of rewrite rules, which describe its evolution. CLS terms are either linear sequences of symbols, or *looping* sequences, i.e. close circular sequences containing other terms, or they are the parallel of two terms. Looping sequences can be used for modelling membranes. CLS allows representing interactions which involve elements on the membranes and, as such, is more expressive than other calculi which consider membranes as atomic objects. As for other biology-oriented calculi, CLS has been extended to Stochastic CLS [19, 7] in order to also allow the modelling of quantitative aspects of biological system; this involves, in particular, the description of speeds associated with different kinds of reaction.

1.1 Including space and time

As the thesis goal, there has been developed an extension of CLS, called *Topological CLS*, which introduces space and time into CLS. In particular, every object comprised within the system has a position and dimensions in space and can move autonomously during the passage of time. It is also possible to constrain rule application according to the exact positions and dimensions of the involved objects; for instance, it is possible to express that a rule is applicable only if the objects are within a certain distance. Objects are represented as hard-spheres with a containment hierarchy among them. Conflicts among objects may arise, when two (or more) objects overlap or when an object is positioned beyond the bounds of

the containing object; this problem is addressed by assuming that objects push each other if they occur to be too close, or if an object is beyond the bounds of the containing sphere. Moreover, in a similar way to Stochastic CLS, each TCLS reaction has associated a parameter specifying its rate which, intuitively, models its ease to occur when the reaction is applicable.

The ability to represent explicitly the position of elements and their movement is useful for modelling those biological processes where knowing the positions of the elements is decisive for accurately describing their evolution.

As an example application, the calculus has been used to model *cell cycle*, this being the sequence of events that lead up to the division of a cell into two daughter cells, and simulate the proliferation of cells in a limited space.

1.2 Related calculi

As far as we know, there is currently only one other formalism that deals with space and time and that has been specifically developed for modelling biological systems. SpacePI [16] is a recently proposed formalism, which extends π-calculus [20] with space and time. A position in a continous space (such as $\mathbb{R}^2$) is associated with every process, and each process can move autonomously according to its movement function. Processes do not have dimensions, as they do not occupy any space. Time is advanced by discrete steps of fixed duration. Each process moves uniformly during each interval, and can act during it; if a process executes an action, then its movement may change, changing direction and/or speed.

As for the π-calculus, processes may communicate by a synchronization on a channel name, where a process does a send action and the other does its corresponding receive action. However, unlike the original calculus, each send or receive action has associated a parameter r which specifies its *interaction radius*. Besides those actions, there is also an action which permits to delay a process until a certain time.

The semantics of the calculus states that a synchronization between two processes may occur only if their distance is less than the sum of the radii of their complementary actions. Moreover, every communication happens as soon as it becomes applicable, i.e. as soon as they are close enough. In every time interval

there could occur an unlimited number of communications.

1.3 Structure of the thesis

The thesis is organized as follows:

- Chapter 2 introduces the Calculus of Looping Sequences, which is the calculus from which the Topological CLS is derived, and its stochastic extension, namely Stochastic CLS;
- In Chapter 3 the Topological CLS is defined, and its semantics is explained informally;
- Chapter 4 explains the formal semantics of the calculus;
- Chapter 5 shows an application of the calculus to the modelling of cell cycle, which makes it possible to represent and simulate the proliferation of cells in a limited space;
- Chapter 6 contains the conclusions on the work carried out.

Chapter 2

The Calculus of Looping Sequences

This chapter introduces the *Calculus of Looping Sequences* (CLS), which is the calculus from which the Topological CLS is derived. It is also presented the Stochastic CLS, which is another extension of CLS that uses stochastic rates for representing different reaction speeds, in order to allow the modelling of quantitative aspects of biological systems.

2.1 The Calculus of Looping Sequences

The Calculus of Looping Sequence [9, 10, 19] can be used to model biological systems and their evolution. CLS 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, modelling its evolution. In the definition of the syntax, which follows, is assumed a possibly infinite alphabet $\mathcal{E}$ of symbols ranged over by $a, b, c, \ldots$.

Definition 2.1.1 (Terms)**.** Terms T *and* Sequences S *of CLS are given by the following grammar:*

$$\begin{array}{rcl} T & ::= & S \;\mid\; \left(S\right)^{L} \rfloor T \;\mid\; T \,|\, T \\ S & ::= & \epsilon \;\mid\; a \;\mid\; S \cdot S \end{array}$$

where a is a generic element of $\mathcal{E}$. $\mathcal{T}$ denotes the infinite set of terms, and $\mathcal{S}$ denotes the infinite set of sequences.

The elements of $\mathcal{E}$ can be used to build sequences, by using sequencing operator $\cdot$. An empty sequence is denoted by ϵ. A term can either be a simple sequence S, a looping sequence $\left(S\right)^L$ containing a term T, or the parallel of two terms. The containment operator $\rfloor$ allows representing compartments; in fact, the looping sequence usually models membranes from the biological system. Also, since membranes are circular, the elements on a looping sequence are assumed to constitute a close circular sequence and, thus, they can "rotate".

Next follows the definition of the structural congruence between terms, that is used to identify the terms that conceptually represent the same biological system, even though they are syntactically different.

Definition 2.1.2 (Structural congruence)**.** *The* structural congruence *is the least congruence relation on terms satisfying the following axioms:*

$$S_1 \cdot (S_2 \cdot S_3) \equiv (S_1 \cdot S_2) \cdot S_3 \tag{2.1}$$

$$S \cdot \epsilon \equiv \epsilon \cdot S \equiv S \tag{2.2}$$

$$T_1 \mid (T_2 \mid T_3) \equiv (T_1 \mid T_2) \mid T_3 \tag{2.3}$$

$$T_1 \mid T_2 \equiv T_2 \mid T_1 \tag{2.4}$$

$$T \mid \epsilon \equiv T \tag{2.5}$$

$$(S_1 \cdot S_2)^L \rfloor T \equiv (S_2 \cdot S_1)^L \rfloor T \tag{2.6}$$

$$(\epsilon)^L \rfloor \epsilon \equiv \epsilon \tag{2.7}$$

Axioms 2.1 and 2.3 express, respectively, associativity of sequencing operator and associativity of parallel operator between terms. Axioms 2.2, 2.5 and 2.7 indicate the neutral role of ϵ: the first is with respect to the sequencing operator, while the second is with respect to parallel operator. Axiom 2.4 states the commutativity of the parallel operator. Finally, axiom 2.6 indicates that the symbols of a looping sequence can rotate.

Example 2.1.1. The following are some example CLS terms (all three of them are graphically represented in figure 2.1):

$$T_1 = \left(a \cdot b \cdot c\right)^L$$
$$T_2 = \left(a \cdot b \cdot c\right)^L \rfloor \left(d \cdot e\right)^L \rfloor \epsilon$$
$$T_3 = \left(a \cdot b \cdot c\right)^L \rfloor (f \cdot g \mid \left(d \cdot e\right)^L \rfloor \epsilon)$$

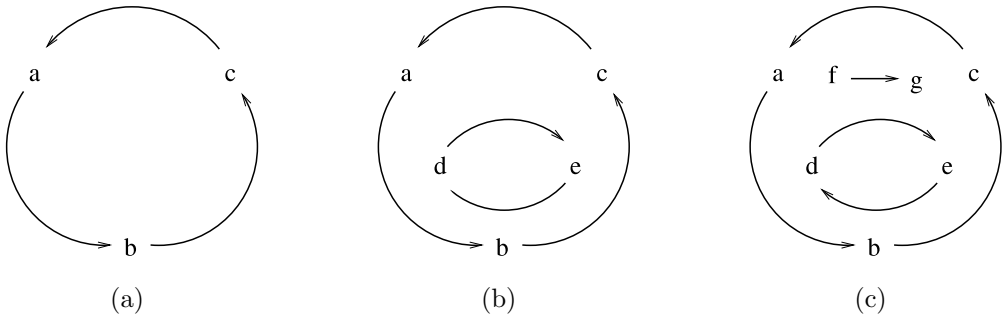


Figure 2.1: *A graphical representation of some example terms.*

Term T_1 (figure 2.1(a)) models an empty membrane, composed of the elements a, b, c. Term T_2 (figure 2.1(b)) represents the same membrane as of T_1, but which contains another looping sequence composed of d, e. In the last term T_3 (figure 2.1(c)), there is also a simple sequence $f \cdot g$ inside the outer looping sequence.

Definition 2.1.3 (Context)**.** Contexts C *are derived from the following grammar:*

$$C \quad ::= \quad \Box \quad | \quad S \quad | \quad \big(S\big)^L \rfloor C \quad | \quad C \mid T \quad | \quad T \mid C$$

where $T \in \mathcal{T}$ *and* $S \in \mathcal{S}$. $\Box$ *denotes the* empty context*. The set of all contexts is denoted by* $\mathcal{C}$*.*

A context C can be composed with a term T, producing a term $C[T]$ obtained by replacing the only $\Box$ appearing in C with the term T.

Example 2.1.2. Consider the following term T and context C:

$$T = c \mid \big(c\big)^L \rfloor d$$
$$C = \big(a \cdot a\big)^L \rfloor (b \mid \Box)$$

then, context C can be composed with T, yielding the term:

$$C[T] = \big(a \cdot a\big)^L \rfloor \Big(b \,|\, c \,|\, \big(c\big)^L \rfloor d\Big).$$

2.1.1 Specifying interactions

The evolution of a system is described by a set of rewrite rules, representing reactions among elements of the system. Each rule is composed of a pair of terms, with the intuitive meaning that, if the first term occurs in the system, then it can be replaced by the second term.

Precisely, each rewrite rule is composed of a pair of *patterns*, that are terms with variables. The following sets of variables are assumed, each representing a different kind of variable:

- TV, the set of *term variables*, ranged over by $X, Y, Z, \ldots$;
- SV, the set of *sequence variables*, ranged over by $\widetilde{x}, \widetilde{y}, \widetilde{z}, \ldots$;
- $\mathcal{X}$, the set of *element variables*, ranged over by $x, y, z, \ldots$.

All these sets are assumed to be pairwise disjoint and possibly infinite. The set of all variables is denoted by $\mathcal{V}$; i.e. $\mathcal{V} = TV \cup SV \cup \mathcal{X}$.

Definition 2.1.4 (Patterns)**.** Patterns P *and* sequence patterns SP *of CLS are given by the following grammar:*

$$\begin{array}{rcl} P & ::= & SP \;\mid\; \left(SP\right)^{L} \rfloor P \;\mid\; P \,|\, P \;\mid\; X \\ SP & ::= & \epsilon \;\mid\; a \;\mid\; SP \cdot SP \;\mid\; \widetilde{x} \;\mid\; x \end{array}$$

where a is a generic element of $\mathcal{E}$, and X, $\widetilde{x}$ and x are generic elements of TV, SV and $\mathcal{X}$, respectively. The infinite set of patterns is denoted with $\mathcal{P}$.

Given a pattern P, the set of all variables appearing in it is denoted by $\mathrm{Var}(P)$. It is also assumed that structural congruence relation is extended to patterns.

A term can be obtained from a pattern by instantiating the variables appearing in it, with terms, sequences or symbols. The binds between variables and their values are described by an *instantiation function*, whose definition follows.

Definition 2.1.5 (Instantiation function)**.** *An* instantiation function *for variables in $\mathcal{V}$ is a partial function $\sigma : \mathcal{V} \to \mathcal{T} \cup \mathcal{S} \cup \mathcal{E}$ that respects the type of variables; thus it is such that:*

$$\forall X \in TV, \widetilde{x} \in SV, x \in \mathcal{X}. \quad \sigma(X) \in \mathcal{T},\ \sigma(\widetilde{x}) \in \mathcal{S},\ \sigma(x) \in \mathcal{E}\,.$$

The set of all instantiation function for $\mathcal{V}$ is denoted by Σ.

The application of an instantiation function σ to a pattern P, written as $P\sigma$, is obtained by replacing each occurence of each variable $v \in \mathrm{Var}(P)$ with $\sigma(v)$.

Example 2.1.3. Let σ be an instantiation function such that:

$$\sigma(\widetilde{x}) = a \cdot a$$
$$\sigma(X) = \left(c\right)^L \rfloor d$$

then, its application to pattern $P = b \mid \left(a \cdot \widetilde{x}\right)^L \rfloor X$ yields the term:

$$P\sigma = b \mid \left(a \cdot a \cdot a\right)^L \rfloor \left(c\right)^L \rfloor d$$

2.1.2 Rewrite rules

Definition 2.1.6 (Rewrite Rule)**.** *A* rewrite rule *is a pair of patterns (P_1, P_2), usually written as*

$$P_1 \mapsto P_2$$

where $P_1, P_2 \in \mathcal{P}$, $P_1 \not\equiv \epsilon$ and such that $Var(P_2) \subseteq Var(P_1)$.

A rewrite rule which does not contain any variable, i.e. such that $Var(P_1) = Var(P_2) = \emptyset$, is said to be *ground*.

A rewrite rule (P_1, P_2) states that a term $P_1\sigma$, obtained by instantiating variables in P_1 by some instantiation function σ, can be transformed into the ground term $P_2\sigma$. A rewrite rule is applicable to a term T only if the left part of a ground rewrite rule, obtained by instantiating it, can be "matched" by a subterm of T. Specifically, there needs to be a context C such that $C[P_1\sigma] \equiv T$.

The application of rewrite rules to terms allows the system to evolve. The semantics of the calculus is given as a transition system, in which states correspond to terms, and in which each transition represents the application of a rewrite rule to a term.

Definition 2.1.7 (Semantics)**.** *Given a set of rewrite rules $\mathcal{R}$, the* semantics *of CLS is the least transition relation $\longrightarrow$ on terms closed under $\equiv$, and satisfying the following inference rule:*

$$\frac{P_1 \mapsto P_2 \in \mathcal{R} \qquad P_1\sigma \not\equiv \epsilon \qquad \sigma \in \Sigma \qquad C \in \mathcal{C}}{C[P_1\sigma] \longrightarrow C[P_2\sigma]}$$

Example 2.1.4. Let $\mathcal{R}$ be a set containing only the following rewrite rules:

$$a \mid \left(b \cdot \widetilde{x}\right)^L \rfloor X \mapsto \left(b \cdot \widetilde{x}\right)^L \rfloor (X \mid a) \tag{2.8}$$

$$\left(b \cdot \widetilde{x}\right)^L \rfloor (X \mid a) \mapsto a \mid \left(b \cdot \widetilde{x}\right)^L \rfloor X \tag{2.9}$$

$$c \mid a \mapsto e \tag{2.10}$$

Let the term $T = a \mid \left(b \cdot b\right)^L \rfloor c$ denote the initial system. It may evolve, by applying rule 2.8, to term $T_2 = \left(b \cdot b\right)^L \rfloor (a \mid c)$. Then, both rules 2.9 and 2.10 can be applied to T_2: the first yields back term T_1 while the second yields term $T_3 = \left(b \cdot b\right)^L \rfloor e$, to which no rule can be applied anymore.

2.2 Stochastic CLS

Stochastic CLS [19, 7] incorporates in CLS the stochastic framework developed by Gillespie [15], in order to allow the modelling of quantitative aspects of biological systems.

The syntax of SCLS is the same as that of CLS, while rewrite rules are enriched with stochastic rates: these stochastic rates model the speeds of the reactions described by each rule. Precisely, as in Gillepie's algorithm, given a biological system and a specific reaction (with associated a certain constant, derived from its kinetic constant), the *application rate* of the reaction occurring in the system is computed by multiplying its constant by the number of possible combinations of reactants occurring in the system. Then, the time which passes between two occurrences of the considered reaction is modelled by an exponential distribution, whose parameter is the computed application rate.

Given a rule that can be applied to a term T, the application rate of the reaction modelled by the rule depends on the number of different ways in which the rule can be applied to T, that conceptually represents the number of different reactants among which the reaction could occur. In order to give the semantics of the calculus, the following definitions are needed.

Definition 2.2.1 (Stochastic Rewrite Rule Schema)**.** *A* rewrite rule schema *is a triple* (P_1, P_2, f)*, denoted with* $P_1 \overset{f}{\mapsto} P_2$*, where* $P_1, P_2 \in \mathcal{P}$*,* $P_1 \not\equiv \epsilon$ *and such that* $Var(P_2) \subseteq Var(P_1)$*, and* $f : \Sigma \to \mathbb{R}^{\geq 0}$ *is the* rewriting rate function*.*

As it can be noted, each stochastic rewrite rule schema is enriched with a *rewriting rate function* f, instead of a simple constant $k \in \mathbb{R}$. This is because application rate of a rewrite rule may depend on the particular instantiation of the variables appearing in it. For instance, the binding of a molecule b with a molecule a on a membrane can be represented by the following rule:

$$b \mid \left(a \cdot \widetilde{x}\right)^L \rfloor X \overset{k}{\mapsto} \left(c \cdot \widetilde{x}\right)^L \rfloor X$$

Intuitively, b can react with any a-element occurring on the membrane, hence the application rate depends on the number of occurrences of a-elements on the membrane. Therefore, the rewriting rate function is introduced for computing the application rate depending on the particular instantiation of the rule (that is, depending on σ).

Given an instantiation function σ, a *stochastic rewrite rule schema* can be turned into a *stochastic ground rewrite rule* by instantiating its patterns and replacing the rewrite rate function f with its actual value $c = f(\sigma)$ (called *rewriting rate constant*).

Definition 2.2.2 (Stochastic Ground Rewrite Rule)**.** *Given a stochastic rewrite rule schema $R = (P_1, P_2, f)$ and an instantiation function $\sigma \in \Sigma$, the* ground rewrite rule *derived from R and σ is a triple (T_1, T_2, c), denoted as $T_1 \overset{c}{\mapsto} T_2$, where $T_1 = P_1\sigma$, $T_2 = P_2\sigma$, and $c = f(\sigma)$ is the* rewriting rate constant*.*

Example 2.2.1. Consider the following rewrite rule schema:

$$R \ : \ a \mid \left(c \cdot \widetilde{x}\right)^L \rfloor \epsilon \overset{f}{\mapsto} b \mid \left(\widetilde{x}\right)^L \rfloor \epsilon$$

where the rewriting rate function f is defined as:

$$f(\sigma) = 1 + occ(c, \sigma(\widetilde{x}))$$

and where, in turn, the function $occ : \mathcal{E} \times \mathcal{S} \to \mathbb{N}$ counts the number of occurrences of a symbol in a sequence.

Given an instantiation function σ such that $\sigma(\widetilde{x}) = b \cdot c$, the ground rewrite rule obtained from R is:

$$a \mid \left(c \cdot b \cdot c\right)^L \rfloor \epsilon \overset{2}{\mapsto} b \mid \left(b \cdot c\right)^L \rfloor \epsilon \,.$$

where its rewriting rate constant is $f(\sigma) = 2$.

The following definition computes the set of all ground rules, deriving from a given set of rule schemata, that can be applied to a given term.

Definition 2.2.3 (Applicable Ground Rewrite Rules)**.** *Given a rewrite rule schema* $R = P_1 \overset{f}{\mapsto} P_2$ *and a term* $T \in \mathcal{T}$*, the set of* ground rewrite rules *derived from* R *and applicable to* T *is defined as*

$$\begin{aligned} \mathrm{AR}(R,T) = \{\ T_1 \overset{c}{\mapsto} T_2 \ \mid \exists \sigma \in \Sigma, C \in \mathcal{C}. \\ T_1 = P_1\sigma,\ T_2 = P_2\sigma,\ T \equiv C[T_1],\ c = f(\sigma)\ \} \end{aligned} \tag{2.11}$$

Given a finite set of rewrite rule schemata $\mathcal{R}$ *and a term* $T \in \mathcal{T}$*, the set of* ground rewrite rules *derived from* $\mathcal{R}$ *and applicable to* T *is:*

$$AR(\mathcal{R}, T) = \bigcup_{R \in \mathcal{R}} AR(R, T)$$

Function $ext(T)$, defined in the following, computes the number of different ways in which a ground rule can be applied to a term T. Precisely, it computes the *multiset of extracted reactants* of T, which is the multiset of all the pairs (T', C) where $T' \not\equiv \epsilon$ is a reactant in T and C is the context from which T' is extracted, i.e. such that $C[T'] \equiv T$. The definition uses the auxiliary function $\circ : \mathcal{C} \times (\mathbb{N} \times \mathcal{T} \times \mathcal{C}) \mapsto (\mathbb{N} \times \mathcal{T} \times \mathcal{C})$ defined as:

$$C \circ (i, T, C') = (i, T, C[C'])$$

and extended to multisets of triples over $\mathbb{N} \times \mathcal{T} \times \mathcal{C}$.

Definition 2.2.4 (Multiset of Extracted Reactants)**.** *Given a term* $T \in \mathcal{T}$*, the multiset of reactants extracted from* T *is defined as*

$$ext(T) = \{(T', C) | (n, T', C) \in ext_\ell(0, T)\}$$

where ext_ℓ is given by the following recursive definition:

$$
\begin{aligned}
ext_\ell(i, S) &= \{(i, S, \Box)\} \\
ext_\ell(i, \big(S\big)^L) &= \{(i, \big(S\big)^L, \Box)\} \\
ext_\ell(i, \big(S\big)^L \rfloor T') &= \{(i, \big(S\big)^L \rfloor T', \Box)\} \ \cup \ \big(S\big)^L \rfloor \Box \circ ext_\ell(i+1, T') \\
ext_\ell(i, T_1 \mid T_2) &= T_2 \mid \Box \circ ext_\ell(i, T_1) \ \cup \ T_1 \mid \Box \circ ext_\ell(i, T_2) \\
&\quad \cup \ \{(i, T_1^e \mid T_2^e, C_1^e[C_2^e]) \mid (i, T_j^e, C_j^e) \in ext_\ell(i, T_j), j \in \{1, 2\}\}
\end{aligned}
$$

Function $ext(T)$ determines all the possible reactants (subterms) of T to which a rewrite rule could be applied, along with the contexts from which they are extracted.

Example 2.2.2. Let T be the following term:

$$T = a \mid \big(b\big)^L \rfloor c$$

then, in order to compute the multiset of extracted reactants of T it is needed to compute $ext_\ell(0, T)$, which gives the following set of triples:

$$
\begin{aligned}
ext_\ell(0, T) = \{ \ & (0, a, \Box \mid \big(b\big)^L \rfloor c), \\
& (0, \big(b\big)^L \rfloor c, a \mid \Box), \\
& (1, c, a \mid \big(b\big)^L \rfloor \Box), \\
& (0, T, \Box) \ \} .
\end{aligned}
$$

The obtained set $ext(T)$ is:

$$
\begin{aligned}
ext(T) = \{ \ & (a, \Box \mid \big(b\big)^L \rfloor c), \\
& (\big(b\big)^L \rfloor c, a \mid \Box), \\
& (c, a \mid \big(b\big)^L \rfloor \Box), \\
& (T, \Box) \ \} .
\end{aligned}
$$

Next is defined the *application cardinality* of a rule which counts, given a ground rule R and two terms T, Tr, the number of possible reactants of T to which rule R could be applied, and which lead to the same term Tr.

Definition 2.2.5 (Application Cardinality)**.** *Given a ground rewrite rule* $R = T_1 \overset{c}{\mapsto} T_2$ *and two terms* $T, Tr \in \mathcal{T}$*, the* application cardinality *of rule* R *leading from* T *to* Tr*,* $AC(R, T, Tr)$*, is defined as follows:*

$$AC(R, T, Tr) = \#\left\{(T', C) \in ext(T) \text{ such that } T' \equiv T_1 \wedge C[T_2] \equiv Tr\right\} .$$

Example 2.2.3. Consider the ground rewrite rule $R : a \overset{c}{\mapsto} b$ and the term $T = a \,|\, a \,|\, \left(m\right)^L \rfloor\, a$. Rule R can be applied to T in three different ways, which correspond to the following pairs contained in $ext(T)$:

$$(a, C_1) = (a,\ \Box \,|\, a \,|\, \left(m\right)^L \rfloor\, a) \quad \text{with multiplicity 2;}$$
$$(a, C_2) = (a,\ a \,|\, a \,|\, \left(m\right)^L \rfloor\, \Box) \quad \text{with multiplicity 1.}$$

The first tuple is relative to the application of the rule to one of the two external a-element, while the second is relative to the application to the a-element inside the looping sequence.

The following two terms can be reached from the two rule applications described above (where element b represents the right part of the rule):

$$Tr_1 = C_1[b] = a \,|\, \left(m\right)^L \rfloor\, a \,|\, b$$
$$Tr_2 = C_2[b] = a \,|\, a \,|\, \left(m\right)^L \rfloor\, b$$

which correspond, respectively, to the first and the second application of the rule, as described previously. Finally, the application cardinalities are:

$$AC(R, T, Tr_1) = 2$$
$$AC(R, T, Tr_2) = 1$$

Definition 2.2.6 (Semantics)**.** *Given a finite set of rewrite rule schemata* $\mathcal{R}$*, the semantics of Stochastic CLS is the least labelled transition relation satisfying the following inference rule:*

$$\frac{R = T_1 \overset{c}{\mapsto} T_2 \in AR(\mathcal{R}, T) \qquad T \equiv C[T_1]}{T \xrightarrow{R, c \cdot AC(R,T,C[T_2])} C[T_2]}$$

The semantics of Stochastic CLS constructs a transition system in which states are represented by terms, and where each transition represents the application of a

(ground) rewrite rule to some reactants in T. Each transition has associated a pair, composed of (i) the ground rule applied and (ii) a rate, obtained multiplying the rewriting rate constant of the ground rule by its application cardinality. That rate corresponds to the parameter of an exponential distribution, which characterizes the speed of the reaction described by the application of the rewrite rule. Finally, the transition graph obtained by applying the semantics to a given term T can be transformed into a *Continuous–time Markov Chain* (CTMC) [33].

Example 2.2.4. Let $\mathcal{R}$ be the following set of ground rewrite rules:

$$R_1 \ : \ a \overset{c_1}{\mapsto} b$$
$$R_2 \ : \ a \,|\, a \overset{c_2}{\mapsto} d$$

Considering term $T = a \,|\, a \,|\, \left(m\right)^L \rfloor\, a$ as initial state, the CTMC obtained is shown in figure 2.2, where terms $T_1, \ldots, T_7$ are:

$$T_1 = \left(m\right)^L \rfloor\, a \,|\, d \qquad T_2 = \left(m\right)^L \rfloor\, b \,|\, d$$
$$T_3 = a \,|\, a \,|\, \left(m\right)^L \rfloor\, b \qquad T_4 = a \,|\, b \,|\, \left(m\right)^L \rfloor\, b$$
$$T_5 = b \,|\, b \,|\, \left(m\right)^L \rfloor\, b \qquad T_6 = a \,|\, b \,|\, \left(m\right)^L \rfloor\, a$$
$$T_7 = b \,|\, b \,|\, \left(m\right)^L \rfloor\, a$$

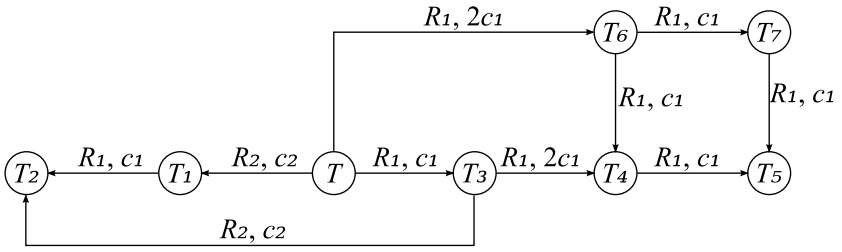


Figure 2.2: *Continuous-time Markov Chain from example 2.2.4.*

Chapter 3

Including spatiality

This chapter introduces the Topological Calculus of Looping Sequences, that has been developed as the thesis goal. The TCLS is based on CLS, and extends it by representing objects as spheres in space. The objects can move autonomously, within the container membrane, and are able to react when they get close to each other. At first, the calculus syntax and its supporting definitions are presented, and finally the semantics is explained informally.

3.1 Definition of the calculus

In TCLS, every object (simple sequence or looping sequence) of the system is represented as a *sphere* in the continuous space $\mathbb{R}^n$, with a precisely defined position, that can move autonomously during time passage and can react when particular conditions are met (for example, when two reactants are within a certain distance)[1].

Containment operator $\cdot \rfloor \cdot$ naturally defines a containment hierarchy among spheres, with the constraint that an inner sphere cannot be positioned beyond the bounds of the containing sphere. Also, object are modelled as hard-spheres, hence there cannot be any overlap among the spheres.

TCLS syntax extends the one of CLS by associating a triple $d = \langle p, r, m \rangle$ with each term representing a simple sequence (S) or a looping sequence $(S)^L$. The

[1] In the following, it is assumed a *norm* $\|\cdot\|$ on $\mathbb{R}^n$.

triple is composed of:

- the centre p of the sphere, relative to the centre of the containing sphere (if it exists) or to the global coordinate system;
- the radius r of the sphere, which needs to be greater than 0 (this will ensure that there cannot be two sphere located in the same position);
- the movement function m, which describes the autonomous motion of the object during time passage.

Definition 3.1.1 (Syntax)**.** Terms T *and* sequences S *of the calculus are defined by the following grammar:*

$$T \quad ::= \quad \lambda \quad | \quad (S)_d \quad | \quad (S)_d^L \rfloor T \quad | \quad T \mid T \tag{3.1}$$

$$S \quad ::= \quad \epsilon \quad | \quad a \quad | \quad S \cdot S \tag{3.2}$$

where $d \equiv \langle p, r, m \rangle$, $p \in \mathbb{R}^n$, $r \in \mathbb{R}^+$, $m : \mathbb{R}^n \times \mathbb{R} \cup \{\infty\} \times \mathbb{R} \times \mathbb{R} \to \mathbb{R}^n$ *and* $a \in \mathcal{E}$.

As for CLS, it is assumed an alphabet $\mathcal{E}$ of symbols, ranged over by $a, b, c, \ldots$, and used for forming sequences. The term λ denotes the *empty term*, while ϵ denotes the empty sequence (it is worth noting that in the original CLS ϵ is used for denoting both the empty sequence and the empty term).

The set of all terms is denoted by $\mathcal{T}$, and the set of all sequences is denoted by $\mathcal{S}$. Finally, $\mathcal{M}$ denotes the set of all movement functions.

Example 3.1.1. Let T_1 be the following term (defined over space $\mathbb{R}^2$)

$$T_1 = (a)_{\langle (1,2), 0.5, m_1 \rangle} \mid (b \cdot c \cdot d)^L_{\langle (4,3), 2, m_2 \rangle} \rfloor (a)_{\langle (-1,0), 0.5, m_3 \rangle}$$

As illustrated in figure 3.1, term T is composed of two top-level elements:

- a simple sequence (a) with center in $(1, 2)$, radius 0.5 and movement function m_1;
- a looping sequence $(b \cdot c \cdot d)^L$ with center in $(4, 3)$, radius 2 and movement function m_2;

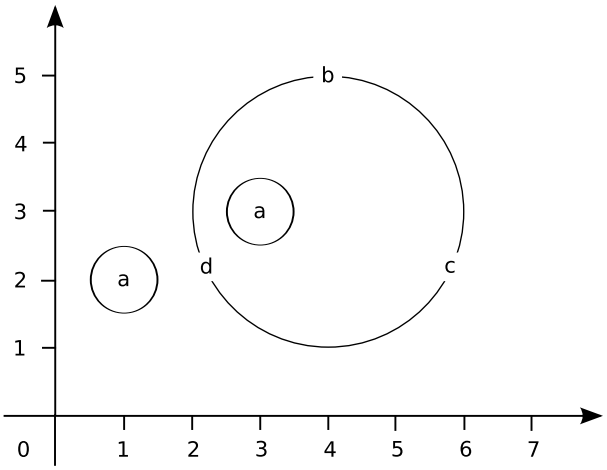


Figure 3.1: *Graphical representation of the term from example 3.1.1*

- another sequence (a), contained in the looping sequence, with the same radius and movement function as the outer sequence, and whose position (relative to the center of the container) is $(-1, 0)$.

Some syntactically different terms conceptually define the same biological system, so a *congruence relation* on terms needs to be defined.

Definition 3.1.2 (Structural congruence)**.** *The* structural congruence *is the least congruence relation on term satisfying the following axioms:*

$$S_1 \cdot (S_2 \cdot S_3) \equiv (S_1 \cdot S_2) \cdot S_3 \tag{3.3}$$

$$S \cdot \epsilon \equiv \epsilon \cdot S \equiv S \tag{3.4}$$

$$T_1 \mid (T_2 \mid T_3) \equiv (T_1 \mid T_2) \mid T_3 \tag{3.5}$$

$$T_1 \mid T_2 \equiv T_2 \mid T_1 \tag{3.6}$$

$$T \mid \lambda \equiv T \tag{3.7}$$

$$(S_1 \cdot S_2)^L_d \rfloor T \equiv (S_2 \cdot S_1)^L_d \rfloor T \tag{3.8}$$

Axiom 3.3 and 3.5 express, respectively, associativity of the sequencing operator and associativity of parallel operator between terms. Axiom 3.4 indicates the neutral role of ϵ with respect to sequencing, while 3.7 expresses the neutral role of λ with respect to parallel operator. Axiom 3.6 states the commutativity of

the parallel operator. Finally, axiom 3.8 indicates that the symbols of a looping sequence can rotate.

As previously indicated, not all of the terms derived from the grammar are valid, since an object cannot exceed the limits of the sphere in which it is contained, and cannot overlap with sibling objects. This intuition is captured by the notion of *well-formedness*, defined next.

Definition 3.1.3 (Well-formed terms)**.** *The set of* well-formed terms *is defined as:*

$$\mathcal{T}_{wf} = \{T \in \mathcal{T} \mid \exists\, l, I.\ \langle T, l\rangle \to I\}$$

where relation $\langle\cdot,\cdot\rangle \to \cdot \subseteq \mathcal{T} \times \mathbb{R}^+ \times (\mathbb{R}^n \times \mathbb{R}^+)$ *is defined by the following set of inference rules*[2]*:*

$$\frac{}{\langle \lambda, l\rangle \to \emptyset} \tag{3.9}$$

$$\frac{}{\langle (S)_{\langle p,r,m\rangle}, l\rangle \to \{(p,r)\}} \qquad \|p\| + r \leq l \tag{3.10}$$

$$\frac{\langle T, r\rangle \to I}{\langle (S)^L_{\langle p,r,m\rangle} \rfloor\, T, l\rangle \to \{(p,r)\}} \qquad \|p\| + r \leq l \tag{3.11}$$

$$\frac{\langle T_1, l\rangle \to I_1 \qquad \langle T_2, l\rangle \to I_2}{\langle T_1 \mid T_2, l\rangle \to I_1 \cup I_2} \qquad \begin{array}{c}\forall\, (p_1, r_1) \in I_1,\ (p_2, r_2) \in I_2.\\ \|p_1 - p_2\| \geq r_1 + r_2\end{array} \tag{3.12}$$

Example 3.1.2. The term T_1 from example 3.1.1 is well-formed. The following is an example of a not well-formed term:

$$T_2 = (e)_{\langle (4.5,3.5),1,m_4\rangle} \mid (b \cdot c \cdot d)^L_{\langle (2.5,2.5),2,m_2\rangle} \rfloor\, (a)_{\langle (-1,-1.5),0.5,m_3\rangle}$$

The term, as shown in figure 3.2, is not well-formed because (i) the two sibling objects (e) and $(b \cdot c \cdot d)^L$ overlap, and also because (ii) object (a) is beyond the limits of the containing looping sequence.

In order to give the semantics of the calculus, as for CLS, we need the definition of contexts, i.e. terms with a "hole".

[2]In the definition, $\|p_1 - p_2\|$ represents the distance between p_1 and p_2.

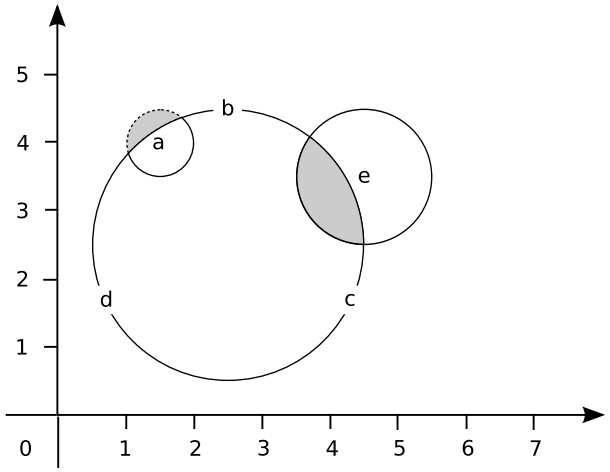


Figure 3.2: *The not well-formed term from example 3.1.2, with the collisions highlighted.*

Definition 3.1.4 (Context)**.** Contexts C *are derived from the following grammar:*

$$C \quad ::= \quad \square \quad | \quad (S)_d \quad | \quad (S)^L_d \rfloor C \quad | \quad C \mid T \quad | \quad T \mid C$$

where $T \in \mathcal{T}$ *and* $S \in \mathcal{S}$*. The set of all contexts is denoted by* $\mathcal{C}$*. A context* C *can be composed with a term* T*, producing a term* $C[T]$ *obtained by filling the hole of* C *with the term* T*.*

Definition 3.1.5 (Structural congruence over contexts)**.** *Two contexts* $C_1, C_2 \in \mathcal{C}$ *are structurally congruent if and only if, given* $\overline{e} \in \mathcal{E}$ *such that it does not appear in* C_1 *nor* C_2*:*

$$C_1[\overline{e}] \equiv C_2[\overline{e}]$$

3.2 Specifying interactions

Rewrite rules allow to specify how the system evolves, due to the interactions among elements of the biological system. Before formally defining rewrite rules, the definitions of *patterns* and *instantiation functions* are needed.

3.2.1 Patterns

Intuitively, patterns are simply terms with variables. Four types of variable are present in the calculus:

- *position variables*, ranged over by $u, v, \ldots$ and contained in set PV;
- *term variables*, ranged over by $X, Y, Z, \ldots$ and contained in set TV;
- *sequence variables*, ranged over by $\widetilde{x}, \widetilde{y}, \widetilde{z}, \ldots$ and contained in set SV;
- *element variables*, ranged over by $x, y, z, \ldots$ and contained in set $\mathcal{X}$.

All of the sets of variables are pairwise disjoint and possibly infinite. The sets TV, SV and $\mathcal{X}$ are the same sets defined in CLS, while the set PV is new. Moreover, $\mathcal{V}$ denotes the set of all term, sequence and element variables, that is: $\mathcal{V} = TV \cup SV \cup \mathcal{X}$.

Besides the instantiation function σ for variables in $\mathcal{V}$ (the same as for CLS), another instantiation function for position variables in PV is needed. Both definitions follow.

Definition 3.2.1 (Instantiation function for $\mathcal{V}$)**.** *An* instantiation function *for variables in $\mathcal{V}$ is a partial function $\sigma : \mathcal{V} \to \mathcal{T}_{wf} \cup \mathcal{S} \cup \mathcal{E}$ that respects the type of variables; thus it is such that:*

$$\forall X \in TV, \widetilde{x} \in SV, x \in \mathcal{X}. \quad \sigma(X) \in \mathcal{T}_{wf},\ \sigma(\widetilde{x}) \in \mathcal{S},\ \sigma(x) \in \mathcal{E}\,.$$

The set of all instantiation functions for $\mathcal{V}$ is denoted by Σ.

Definition 3.2.2 (Instantiation function for positions)**.** *An* instantiation function *for position variables is a partial function $\tau : PV \to Pos$, where $Pos = \{\langle p, r\rangle \mid p \in \mathbb{R}^n,\ r \in \mathbb{R}^+\}$. The set of all instantiation functions for positions is denoted by $\mathbf{T}$.*

Definition 3.2.3 (Patterns)**.** Left patterns P_L, right patterns P_R, located patterns P *and* sequence patterns PS *are defined by the following grammar:*

$$\begin{aligned}
P_L &::= \lambda \mid (PS)_u \mid (PS)^L_u \rfloor P_L \mid P_L \mid P_L \mid X \\
P_R &::= \lambda \mid (PS)_{\langle f,m\rangle} \mid (PS)^L_{\langle f,m\rangle} \rfloor P_R \mid P_R \mid P_R \mid X \\
P &::= \lambda \mid (PS)_d \mid (PS)^L_d \rfloor P \mid P \mid P \mid X \\
PS &::= \epsilon \mid a \mid PS \cdot PS \mid x \mid \widetilde{x}
\end{aligned}$$

where $u \in PV$, $d \in Pos \times \mathcal{M}$, $f : \mathbf{T} \to Pos$.

Given a pattern P, the set of all variables appearing in it is denoted with $\mathrm{Var}(P)$.

Instantiation functions for position variables may be applied to left and right patterns, producing located patterns, according to the following definitions.

Definition 3.2.4. *An instantiation function* $\tau \in \mathbf{T}$ *can be applied to a left pattern* P_L*, giving an infinite set of patterns* P*:*

$$\begin{aligned}
\lambda\tau &= \{\lambda\} \\
(PS)_u\tau &= \{(PS)_{\langle\tau(u),m\rangle} \ : \ m \in \mathcal{M}\} \\
((PS)^L_u \rfloor P_L)\tau &= \{(PS)^L_{\langle\tau(u),m\rangle} \rfloor P \ : \ P \in P_L\tau,\ m \in \mathcal{M}\} \\
(P_L \mid P'_L)\tau &= \{P \mid P' \ : \ P \in P_L\tau,\ P' \in P'_L\tau\} \\
X\tau &= \{X\}
\end{aligned}$$

Definition 3.2.5. *An instantiation function* $\tau \in \mathbf{T}$ *can be applied to a right pattern* P_R*, giving a pattern* P*:*

$$\begin{aligned}
\lambda\tau &= \lambda \\
(PS)_{\langle f,m\rangle}\tau &= (PS)_{\langle f(\tau),m\rangle} \\
((PS)^L_{\langle f,m\rangle} \rfloor P_R)\tau &= (PS)^L_{\langle f(\tau),m\rangle} \rfloor P_R\tau \\
(P_R \mid P'_R)\tau &= P_R\tau \mid P'_R\tau \\
X\tau &= X
\end{aligned}$$

Finally, an instantiation function σ may be applied to a located pattern P, giving the term obtained by instantiating its variables; i.e. giving the term in which each each occurrence of each variable $v \in \mathcal{V}$ is replaced by $\sigma(v)$. The application of σ to a pattern is written as $P\sigma$.

3.2.2 Rewrite rules

Definition 3.2.6 (Rewrite Rule)**.** *A* rewrite rule *is a 5-tuple* (f_c, P_L, P_R, k, f)*, usually written as*

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

where $f_c : \mathbf{T} \to \{tt, ff\}$*,* $k \in \mathbb{R}^+$*,* $f : \Sigma \to \mathbb{N}$ *and* $\mathrm{Var}(P_R) \subseteq \mathrm{Var}(P_L)$.

The semantic meaning of a rewrite rule is that, given a term T representing the system at a certain time, if T contains a subterm that can be "matched" with the left part of the rule, then T may be rewritten to another term similar to T, but where the matched subterm is replaced with a "target" term derived from the right 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 objects (by checking their positions and/or radii). There is also a parameter k that specifies the *rate* of the reaction modelled by this rule, and a function f that counts the number of different ways that the rule can be applied to a given matching subterm, producing the same target term.

Specifically, there needs to be a position instantiation function τ and an instantiation function σ such that (where T' is the matching subterm of T):

$$P \in P_L\tau \tag{3.13}$$

$$T' = P\sigma \tag{3.14}$$

The function τ, that conceptually carries the binds between position variables appearing in left pattern of the rule and the exact positions and radii of the objects of T that matched, is then used for evaluating f_c. Hence, f_c uses the actual position and radii of matched objects for determining if the rule can be applied (i.e. its value is tt) or not.

Example 3.2.1. Let T be the following term (graphically represented in figure 3.3):

$$T = (a)_{\langle (2,4),0.5,m_1\rangle} \mid (b)_{\langle (2,1),0.5,m_2\rangle} \mid (b)_{\langle (3.5,4),0.5,m_2\rangle}$$

and R the following rewrite rule:

$$R_1 \ : \ [\ f_c\ ] \ \ (a)_u \mid (b)_v \overset{k_1,1}{\mapsto} (c)_{\langle u,m_3\rangle}$$

where f_C checks if the distance between the two reactants (a) and (b) is less than 2; formally:

$$f_c(\tau) = tt \qquad \Leftrightarrow \qquad \tau(u) = \langle p_1, r_1\rangle,\ \tau(v) = \langle p_2, r_2\rangle,\ \|p_1 - p_2\| \leq 2$$

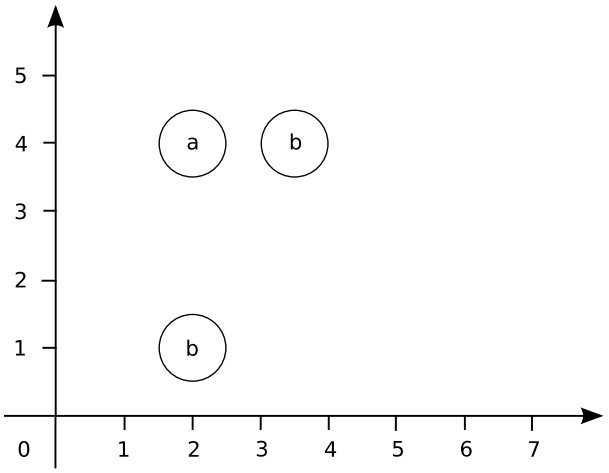


Figure 3.3: *Graphical representation of the term from example 3.2.1*

Intuitively, the rule can be applied only to the objects $(a)_{\langle(2,4),0.5,m_1\rangle}$ and $(b)_{\langle(3.5,4),0.5,m_2\rangle}$. In fact, in such case, the instantiation function τ_1 is such that

$$\tau_1(u) = \langle(2,4),0.5\rangle$$
$$\tau_1(v) = \langle(3.5,4),0.5\rangle$$

and it satisfies the constraint function $f_c(\tau_1) = tt$. The other possible instantiation function τ_2, that also matches two objects of the system, would be

$$\tau_2(u) = \langle(2,4),0.5\rangle$$
$$\tau_2(v) = \langle(2,1),0.5\rangle$$

but τ_2 does not satisfy the constraint, indeed $f_c(\tau_1) = ff$.

3.3 Informal semantics

Given a term T representing the initial system, and assuming a set of rewrite rules $\mathcal{R}$, the system evolves executing a sequence of steps, where each step is composed of two distinct phases:

1. a first phase where, considering a small time interval δt, at most one reaction among the ones available, is executed;

2. a second phase in which time passes, for a duration equal to δt; this involves moving all the objects to their new positions according to their movement functions.

After applying a rewrite rule, it may happen that the obtained term is not well-formed, i.e. two (or more) objects may overlap or some objects may not be correctly contained in a looping sequence. The same situation can happen after moving the objects. This problem is resolved by executing, as a last operation within both phases, a "rearrangement" of the objects of the system. The rearrangement is done by an algorithm, which may fail in determining a well-formed term: if such is the case when trying to apply a rule to a group of objects, then it is assumed that that group of objects cannot react (so the rule cannot be applied).

The length of the time interval δt depends on the number of ways all the rewrite rules in $\mathcal{R}$ could be applied to specific objects in the current state. Nevertheless, its maximum length is fixed to $\Delta t = 1/N$, where N is an arbitrary value such that (k_R denotes the rate associated with reaction R):

$$\forall R \in \mathcal{R}. \quad 0 < k_R/N \leq 1$$

The value k_R/N conceptually represents the probability that a reaction of kind R, among some specific objects in the system, happens in a time interval of length Δt.

It is worth noting that the choice of N may, indirectly, affect the precision in representing movement; in fact, the bigger the time interval Δt is, the bigger the covered distance will usually be in every single phase of movement (unless, for example, the object does not move).

3.3.1 Transition system

The semantics is given as a *Probabilistic Transition System* [28, 21], where two kinds of state are present, representing the two phases forming each step:

- $\langle T, t \rangle$, that represents the system at time t, and is associated with the first phase; from this state only a state of the second kind can be reached, and the probabilities associated with the transitions are computed from the rates of the rewrite rules;

- $\langle T, t, \delta t \rangle$, that is associated with the second phase, and which means that the current time t must be advanced to $t + \delta t$, updating the positions of all the objects according to their movement functions. From this state, a state in the form $\langle T', t + \delta t \rangle$ will be reached.

An example transition system is presented in figure 3.4

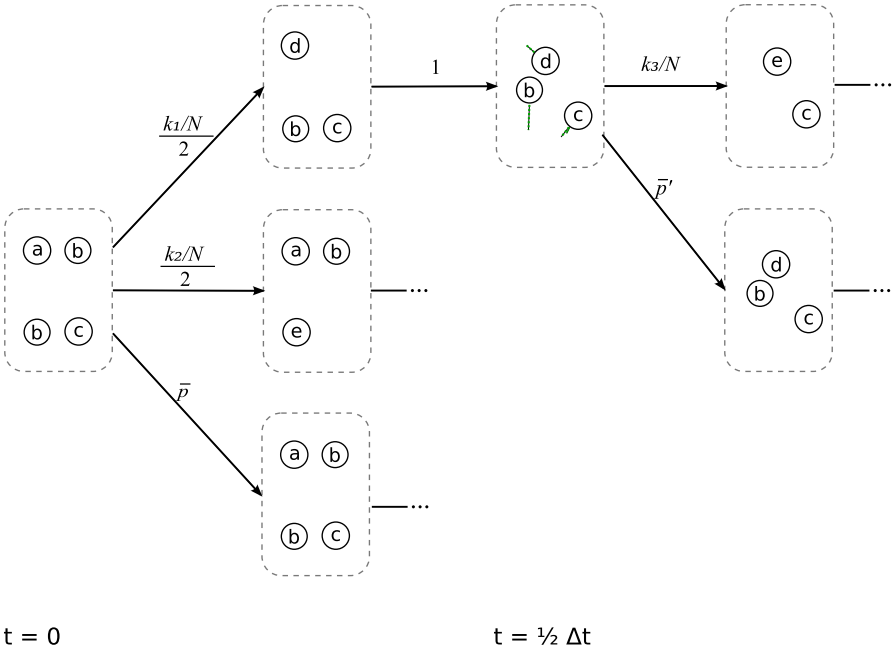


Figure 3.4: *An example transition system.*

The length of time interval δt, that corresponds to the amount of time in which the objects can only move before trying another reaction, is calcolated as $\delta t = \Delta t / m$, where m is the total number of reactions that can occur in current state.

Counting allowed reactions

Counting the number of allowed reactions involves determining every group of objects that matches the left part of the rule. Unfortunately, this is not enough when the rule models a reaction between an element on a looping sequence and

another object (either internal or external). Such situation is tipically modelled by a rule similar to the following:

$$[\ f_c\ ]\ \ (b)_u\ \mid (a \cdot \widetilde{x})^L_v \rfloor X \overset{k,f}{\mapsto} (c \cdot \widetilde{x})^L_{\langle v,m\rangle} \rfloor X$$

The example rule models a situation where an object (b) can react with an element a on the membrane, transforming it to c. Intuitively, if a membrane contains more than one element a, the reaction could happen between b and *any* element a on the membrane; that is, the number of different reactions is given by the number of a-elements on the membrane. For example, given the following term:

$$T = (b)_{\langle p,r,m\rangle}\ \mid (a \cdot a \cdot a)^L_{\langle p',r',m'\rangle} \rfloor \lambda$$

and supposing that the constraints in f_c are satisfied, then there is only one way in which T can match the left part of the rule; this happens with the following instantiation functions τ and σ:

$$\begin{aligned}\tau(u) &= \langle p, r\rangle\\ \tau(v) &= \langle p', r'\rangle\\ \sigma(\widetilde{x}) &= a \cdot a\\ \sigma(X) &= \lambda\end{aligned}$$

This problem is resolved by introducing a function $f : \Sigma \to \mathbb{N}$ associated with each rewrite rule, that counts, given an instantiation function σ, the number of conceptually different reactions that can occur. For the previous example, this function would have been defined as

$$f(\sigma) = 1 + occ(a, \sigma(\widetilde{x}))$$

where function $occ : \mathcal{E} \times \mathcal{S} \to \mathbb{N}$ counts the number of occurrences of a symbol in a sequence.

Calculating reaction probabilities

In the same way as the length of time interval δt is determined by the number of reactions allowed m, also the probability of executing a reaction among a specific

group of objects in time interval δt is determined from m. In fact, the probability that a specific group of objects, matching the left part of a rewrite rule R, reacts in a time interval δt is:

$$\frac{k_R/N}{m}\, n\,.$$

The value $n \in \mathbb{N}$ is obtained by evaluating the function f associated with the rewrite rule, and as such it counts the number of conceptually different reactions that this application of the rule describes. Finally, unless all of the probabilities k_R/N equals 1, there must be also a non-null probability of executing no reaction in the time interval δt. This probability is the sum of the probabilities of not executing each possible reaction.

It must be noted that, given a term T, there may be different rule applications that reduce to a same term T'. As such, the probabilities associated with each transition from a state $\langle T, t\rangle$ to a state $\langle T', t, \delta t\rangle$ must be calculated as the sum of the probabilities of all the different rule applications that lead up to T' from T. Indeed, a reaction could also lead up to term T, and as such its probability must be added to the one of executing no reactions.

3.3.2 Resolving physical overlaps

As previously indicated, during system evolution it may be needed to rearrange the objects because some of them collide. This rearrangement is done by the function DeepArrange, that takes a term (which may not be well-formed) and tries to compute a well-formed term based on it. If a well-formed term cannot be obtained, then the special value $\perp$ is returned.

Definition 3.3.1. *Function* DeepArrange : $\mathcal{T} \to \mathcal{T}_{wf} \cup \{\perp\}$*, where* $\perp \notin \mathcal{T}$*, is such that*

$$\text{DeepArrange}(T) = \begin{cases} T' & \text{if } T \to^{\infty} T' \\ \perp & \text{otherwise} \end{cases}$$

where relation $\rightarrow^r$ *is defined by:*

$$\frac{T \rightarrow I \qquad T' = \text{Arrange}(I, r) \qquad T' \neq \perp}{T \rightarrow^r T'} \tag{3.15}$$

$$(S)_d \rightarrow \{(S)_d\} \tag{3.16}$$

$$\frac{T \rightarrow^r T' \qquad T' \neq \perp}{(S)^L_{\langle p,r,m\rangle} \rfloor T \rightarrow \{(S)^L_{\langle p,r,m\rangle} \rfloor T'\}} \tag{3.17}$$

$$\frac{T_1 \rightarrow I_1 \qquad T_2 \rightarrow I_2}{T_1 \mid T_2 \rightarrow I_1 \uplus I_2} \tag{3.18}$$

The function DeepArrange simply walks down the containment hierarchy among objects and calls, for each looping sequence, the function Arrange (see section 4.3) with the multiset of all the parallel terms contained in the looping sequence and its radius (which indicates the available space). Arrange may fail, returning $\perp$, hence a well-formed term is obtained only if the objects inside all looping sequences can be successfully rearranged by function Arrange.

If there is not any conflict among the objects contained in set I, and between those objects and the container, then function Arrange is assumed to return a term representing the parallel of all the terms in I; that is, Arrange does not move any object if there is not any conflict. This implies that DeepArrange, when applied to a well-formed term $T \in \mathcal{T}_{wf}$, returns a well-formed term T' structurally congruent to it, i.e. such that $T' \equiv T$.

3.4 Representing movement

To each object of the calculus, either a simple sequence or a looping sequence, is associated a movement function that determines the autonomous movement of the object while time passes. A movement function $m : \mathbb{R}^n \times \mathbb{R} \cup \{\infty\} \times \mathbb{R}^+ \times \mathbb{R}^+ \rightarrow \mathbb{R}^n$ calculates a destination position p' as:

$$p' = m(p, l, t, \delta t)$$

where p is the current position of the object, t is the current time and l is the radius of the containing looping sequence, that constrains the possible destination

positions; δt is the amount of time that needs to pass after t.

Every movement function must satisfy the following property, that states that a movement function must never yield a destination position that causes the object to be beyond the limits of the containing membrane.

Property 3.4.1. Let $m \in \mathcal{M}$ be a movement function associated with an object with radius r. Then:

$$\forall p,\, l,\, t,\, \delta t. \quad p' = m(p, l, t, \delta t),\ \|p\| + r \leq l \ \Rightarrow\ \|p'\| + r \leq l \tag{3.19}$$

Example 3.4.1 (Brownian motion)**.** Brownian motion [32] identifies the irregular and random movement, that can be observed for small particles immersed in a fluid. This motion is due to the continuos collisions between the much smaller molecules of the fluid and the particle observed: considered individually, the effect of every single collision on the position of the particle observed is negligible but, as a whole, it becomes visible.

From physics, if the kind of fluid and temperature are known, and assuming that the particle is a small sphere with a certain radius, then the expected distance r covered by the particle in a small time interval t can be calculated as:

$$r = \sqrt{2\,D\,t}$$
$$D = \frac{k_B\,T}{6\,\pi\,\eta\,R}$$

where k_B is the *Boltzmann constant*. D, the *diffusion coefficient*, depends on:

- T, the current temperature;
- η, a constant that caracterizes the kind of fluid (but also depends on T);
- R, the radius of the particle.

The Brownian motion can hence be modelled by a movement function m_1 that depends only on current position p, and can return, with equal probability, every position p' whose distance from p is equal to r. It is worth noting that if some values needed for calculating the distance r with above formulae are unknown, then an estimated value may also be used; such value may be obtained, for example, by experimenting.

From an implementation point of view, first an angle α is chosen randomly, with equal probability in the interval $[0, 2\pi)$, and then the resulting position is obtained moving from p along the direction given by angle α for a distance of r. If the new position would be beyond the limits of the containing membrane, then, for example, the distance covered could be the limited by maximum allowed along the direction chosen.

Chapter 4

Formal semantics

This chapter presents, along with some supporting definitions, the formal semantics of the calculus.

4.1 Support definitions

The evolution of a term representing a biological system is described by a probabilistic transition system [28, 21], which is constructed according to a set of rewrite rules $\mathcal{R}$. For constructing the transition graph, given the term T representing the system at a certain time, it is initially needed to determine which rewrite rules are applicable, and how. This is done by the following functions.

Definition 4.1.1 (Applicable Rewrite Rules)**.** *Given a* rewrite rule $R : [\ f_c\ ]\ \ P_L \overset{k,f}{\mapsto} P_R$ *and a* term T*, all the groud rules applicable to term* T*, along with the context in which they can be applied, are contained in the following set of pairs:*

$$\begin{aligned}\mathrm{AR}(R,T) = \{\ (T_1 \overset{k,n}{\mapsto} T_2, C)\ \mid\ & \\ & \exists \tau \in \mathbf{T}, \sigma \in \Sigma, C \in \mathcal{C}. \\ & P_1 \in P_L\tau,\ P_2 = P_R\tau,\ f_c(\tau) = tt, \\ & T_1 = P_1\sigma,\ T_1 \not\equiv \lambda,\ T_2 = P_2\sigma, \\ & T \equiv C[T_1],\ n = f(\sigma), \\ & \mathrm{DeepArrange}(C[T_2]) \neq \bot\ \}\end{aligned} \tag{4.1}$$

The definition is extended to a set of rewrite rules $\mathcal{R}$

$$\mathrm{AR}(\mathcal{R}, T) = \bigcup_{R \in \mathcal{R}} (R, \mathrm{AR}(R, T))$$

Function $\mathrm{AR}(R, T)$ takes as arguments a rule R and a term T, and computes a set of pairs formed by (i) a ground rule, i.e. an instantiation of the original rewrite rule, and (ii) a context. The left and right parts of the ground rule are obtained by instantiating the corresponding patterns of the original rewrite rule; the ground rule also carries, instead of the function f, the actual value got by its evaluation over the instantiation function σ. The context identifies where the left part of the rule occurs inside T.

A tuple can be contained in $\mathrm{AR}(R, T)$ only if the left part of the rewrite rule, once instantiated, is not congruent to λ ($T_1 \not\equiv \lambda$). This effectively forbids rewrite rules in which the left part is empty, i.e. rules that could create objects from nothing. Also, each tuple is contained in it only if a well-formed term can be obtained after the application of the ground rule in its relative context; that is $\mathrm{DeepArrange}(C[T_2]) \neq \perp$.

Example 4.1.1. Let $\mathcal{R}$ be a set of rewrite rules containing only the following rewrite rules:

$$R_1 : [\ f_c\ ] \quad (a)_u \overset{k_1,1}{\mapsto} (a)_{\langle u,m \rangle}$$
$$R_2 : [\ f'_c\ ] \quad (a)_u \mid (b)_v \overset{k_2,1}{\mapsto} (c)_{\langle u,m'' \rangle}$$

for some rates r_1 and r_2, and where function f, for both rules, is the constant function $f(\sigma) = 1$ (for clarity, they are denoted by 1). Consider also the following term, (where m, m' are given movement functions):

$$T = \big((a)_{\langle (1,1),0.3,m \rangle} \mid (a)_{\langle (3,0),0.3,m \rangle}\big) \mid (b)_{\langle (3,3),0.3,m' \rangle}$$

Intuitively, supposing that objects are close enough for reacting, each rule can be applied in two different ways to T.

Rule R_1 can be instantiated in two ways, while being applicable to T, giving the following ground rules:

$$Rg_1 = (a)_{\langle (1,1),0.3,m \rangle} \overset{k_1,1}{\mapsto} (a)_{\langle (1,1),0.3,m \rangle}$$
$$Rg_2 = (a)_{\langle (3,0),0.3,m \rangle} \overset{k_1,1}{\mapsto} (a)_{\langle (3,0),0.3,m \rangle}$$

A context in which the ground rule Rg_1 can be applied is:

$$C_1 = (\Box \mid (a)_{\langle (3,0),0.3,m\rangle}) \mid (b)_{\langle (3,3),0.3,m'\rangle}$$

but it can also be applied in all other contexts that are structurally congruent to C_1 such as, for instance:

$$\begin{aligned} C_2 &= \Box \mid ((a)_{\langle (3,0),0.3,m\rangle} \mid (b)_{\langle (3,3),0.3,m'\rangle}) \\ C_3 &= ((a)_{\langle (3,0),0.3,m\rangle} \mid \Box) \mid (b)_{\langle (3,3),0.3,m'\rangle} \\ &\vdots \end{aligned}$$

Let all these structurally congruent contexts be denoted by $C_1, \cdots, C_{12}$. For the other ground rule Rg_2, there are another 12 syntactically different (but structurally congruent) contexts, one of which is the following:

$$C_{13} = (\Box \mid (a)_{\langle (1,1),0.3,m\rangle}) \mid (b)_{\langle (3,3),0.3,m'\rangle}$$

The other contexts congruent to C_{13} are denoted by $C_{14}, \cdots, C_{24}$.

To sum up, $\mathrm{AR}(R_1, T)$ corresponds to the following set:

$$\begin{aligned} AR_1 = \mathrm{AR}(R_1, T) = \{\ &(Rg_1, C_1), (Rg_1, C_2), \ldots, (Rg_1, C_{12}), \\ &(Rg_2, C_{13}), (Rg_2, C_{14}), \ldots, (Rg_2, C_{24})\ \} \end{aligned}$$

Similarly, considering rewrite rule R_2, we obtain the following two ground rules:

$$\begin{aligned} Rg_3 &= (a)_{\langle (1,1),0.3,m\rangle} \mid (b)_{\langle (3,3),0.3,m'\rangle} \overset{k_2,1}{\mapsto} (c)_{\langle (1,1),0.3,m''\rangle} \\ Rg_4 &= (a)_{\langle (3,0),0.3,m\rangle} \mid (b)_{\langle (3,3),0.3,m'\rangle} \overset{k_2,1}{\mapsto} (c)_{\langle (3,0),0.3,m''\rangle} \end{aligned}$$

Rg_3 can be applied to the following context C_{25}, while Rg_4 can be applied to C_{27}:

$$\begin{aligned} C_{25} &= \Box \mid (a)_{\langle (3,0),0.3,m\rangle} \\ C_{27} &= (a)_{\langle (1,1),0.3,m\rangle} \mid \Box \end{aligned}$$

There are also two other contexts: C_{26} congruent to C_{25}, and C_{28} congruent to C_{27}. The set $\mathrm{AR}(R_2, T)$ obtained is:

$$\begin{aligned} AR_2 = \mathrm{AR}(R_2, T) = \{\ &(Rg_3, C_{25}), (Rg_3, C_{26}), \\ &(Rg_4, C_{27}), (Rg_4, C_{28})\ \} \end{aligned}$$

and, finally, the set $\mathrm{AR}(\mathcal{R}, T)$ is constructed prepending, to each tuple from sets AR_1 and AR_2, the rewrite rule from which that tuple derives; that is:

$$\mathrm{AR}(\mathcal{R}, T) = \{R_1\} \times AR_1 \cup \{R_2\} \times AR_2$$

Definition 4.1.2. *The subset B_T of $\mathrm{AR}(\mathcal{R}, T)$ contains only one tuple (R, Rg, C) for each tuple of $\mathrm{AR}(R, T)$ with the rewrite rule R, the same ground rule Rg and with a context structurally congruent to C. Formally, B_T is such that:*

$$\begin{aligned} &\forall (R, R_g, C) \in \mathrm{AR}(\mathcal{R}, T). \\ &\quad \exists! (R, R_g, C') \in B_T. \quad C' \equiv C \end{aligned}$$

Example 4.1.2. Going on with previous example (4.1.1), a possible set B_T, for the given set of rewrite rules, is:

$$\begin{aligned} B_T = \{\ &(R_1, Rg_1, C_1), (R_1, Rg_2, C_{13}), \\ &(R_2, Rg_3, C_{25}), (R_2, Rg_4, C_{27})\ \} \end{aligned}$$

Definition 4.1.3. *Given the set B_T and a well-formed term $Tr \in \mathcal{T}_{wf}$, the function $\mathrm{g}(B_T, Tr)$ computes the set of all and only tuples of B_T whose rules (and contexts) lead to term Tr; formally:*

$$\mathrm{g}(B_T, Tr) = \{(R, T_1 \overset{k,n}{\mapsto} T_2, C) \in B_T \mid \mathrm{DeepArrange}(C[T_2]) \equiv Tr\}$$

Finally, the total number of reactions allowed in state T can be computed. Its value is simply the sum, over all ground rules appearing in tuples of set B_T, of the value n associated with each ground rule; in fact, value n represents the number of ways the reaction modelled by that ground rule can conceptually occur in the left part of the ground rule.

Definition 4.1.4. *Given a term T and a set B_T, constructed from a set of rewrite rules $\mathcal{R}$, the total number of different reactions that can occur in T is:*

$$m_{B_T} = \sum_{(R, T_1 \overset{k,n}{\mapsto} T_2, C) \in B_T} n$$

4.2 Formal semantics

The following definitions compute the probabilities that will be used for constructing the probabilistic transition system. It is assumed the value $N \in \mathbb{R}^+$, which can be chosen arbitrarly and must be such that:

$$\forall R \in \mathcal{R}. \quad 0 < k_R/N \leq 1$$

where k_R denotes the rate associated to rewrite rule R. Value N also determines the maximum length of the time interval which is advanced after each step, which is $\Delta t = 1/N$.

Definition 4.2.1 (Probabilities)**.** *Let $\overline{p}$ be the probability of executing no reaction in the next time interval $\delta t = \Delta t / m_{B_T}$, i.e.:*

$$\overline{p} = 1 - \sum_{(R, T_1 \overset{k,n}{\mapsto} T_2, C) \in B_T} \frac{k/N}{m_{B_T}} n$$

Given two terms T and Tr, the probability of reaching state Tr from T is:

$$P(T \to Tr) = \sum_{(R, T_1 \overset{k,n}{\mapsto} T_2, C) \in \mathrm{g}(B_T, Tr)} \frac{k/N}{m_{B_T}} n \; + \; \begin{cases} \overline{p} & \text{if } T \equiv Tr; \\ 0 & \text{otherwise.} \end{cases} \tag{4.2}$$

Intuitively, if $T \not\equiv Tr$, the probability $P(T \to Tr)$ is calculated as the sum of all the probabilities coming from each rule application which lead to the same term Tr. If Tr would have been congruent to T, then the probability $P(T \to T)$ would also have included $\overline{p}$, the one of not executing any reaction.

Example 4.2.1. Continuing with the ongoing example, the number of different reactions allowed in state T is $m_{B_T} = 4$. The probability $P(T \to Tr)$ can be computed for each term Tr reachable from T. All the terms reachable from T are obtained by at least one rule application, among those described by each tuple in B_T. Hence, the reachable terms are:

$$\begin{aligned}
Tr_1 &= \big((a)_{\langle (1,1), 0.3, m \rangle} \mid (a)_{\langle (3,0), 0.3, m \rangle}\big) \mid (b)_{\langle (3,3), 0.3, m' \rangle} && \text{by } (R_1, Rg_1, C_1) \in B_T; \\
Tr_2 &= \big((a)_{\langle (3,0), 0.3, m \rangle} \mid (a)_{\langle (1,1), 0.3, m \rangle}\big) \mid (b)_{\langle (3,3), 0.3, m' \rangle} && \text{by } (R_1, Rg_2, C_{13}) \in B_T; \\
Tr_3 &= (c)_{\langle (1,1), 0.3, m'' \rangle} \mid (a)_{\langle (3,0), 0.3, m \rangle} && \text{by } (R_2, Rg_3, C_{25}) \in B_T; \\
Tr_4 &= (a)_{\langle (1,1), 0.3, m \rangle} \mid (c)_{\langle (3,0), 0.3, m'' \rangle} && \text{by } (R_2, Rg_4, C_{27}) \in B_T.
\end{aligned}$$

It is important to point out that terms Tr_1 and Tr_2 are structurally congruent ($Tr_1 \equiv Tr_2$), and they are also structurally congruent to initial term T. Finally, the probabilities of reaching each of the above terms from T are:

$$q_1 = P(T \to Tr_1) = P(T \to Tr_2) = \frac{k_1/N}{4} \cdot 1 + \frac{k_1/N}{4} \cdot 1 + \overline{p}$$
$$q_2 = P(T \to Tr_3) = \frac{k_2/N}{4} \cdot 1$$
$$q_3 = P(T \to Tr_4) = \frac{k_2/N}{4} \cdot 1$$

As it can be seen, since $Tr_1, Tr_2 \equiv T$, probability q_1 includes the probability $\overline{p}$ of executing no reaction, whose value is:

$$\overline{p} = 1 - \left(2 \cdot \frac{k_1/N}{4} + 2 \cdot \frac{k_2/N}{4}\right)$$

Semantics is given by a probabilistic transition system [28, 21], in which states are represented as tuples in the form $\langle T, t\rangle$ or $\langle T, t, \delta t\rangle$; in both cases, T denotes the congruence class (with respect to $\equiv$) containing all term structurally congruent to T.

Definition 4.2.2 (Semantics)**.** *Given a set of rewrite rules $\mathcal{R}$, the semantics of TCLS is the least relation satisfying the following inference rules:*

$$\frac{\begin{array}{l}(R, T_1 \overset{k,n}{\mapsto} T_2, C) \in B_T \qquad Tr \equiv \text{DeepArrange}(C[T_2]) \qquad Tr \not\equiv T \\ q = P(T \to Tr) \qquad \delta t = \frac{1/N}{m_{B_T}}\end{array}}{\langle T, t\rangle \xrightarrow{q} \langle Tr, t, \delta t\rangle} \tag{4.3}$$

$$\frac{q = P(T \to T) \qquad \delta t = \frac{1/N}{m_{B_T}}}{\langle T, t\rangle \xrightarrow{q} \langle T, t, \delta t\rangle} \tag{4.4}$$

$$\frac{\langle T, t, \delta t\rangle \xrightarrow{\infty} T' \qquad T'' \equiv \text{DeepArrange}(T') \qquad T'' \neq \bot}{\langle T, t, \delta t\rangle \xrightarrow{1} \langle T'', t + \delta t\rangle} \tag{4.5}$$

$$\frac{\langle T, t, \delta t\rangle \xrightarrow{\infty} T' \qquad \text{DeepArrange}(T') = \bot}{\langle T, t, \delta t\rangle \xrightarrow{1} \langle T, t + \delta t\rangle} \tag{4.6}$$

and where relation $\xrightarrow{l}$*, used in inference rules 4.5 and 4.6, is the least relation satisfying:*

$$\frac{p' = m(p, l, t, \delta t)}{\langle (S)_{\langle p,r,m\rangle}, t, \delta t\rangle \xrightarrow{l} (S)_{\langle p',r,m\rangle}} \tag{4.7}$$

$$\frac{p' = m(p, l, t, \delta t) \qquad \langle T, t, \delta t\rangle \xrightarrow{r} T'}{\langle (S)^L_{\langle p,r,m\rangle} \rfloor T, t, \delta t\rangle \xrightarrow{l} (S)^L_{\langle p',r,m\rangle} \rfloor T'} \tag{4.8}$$

$$\frac{\langle T_1, t, \delta t\rangle \xrightarrow{l} T_1' \qquad \langle T_2, t, \delta t\rangle \xrightarrow{l} T_2'}{\langle T_1 \mid T_2, t, \delta t\rangle \xrightarrow{l} T_1' \mid T_2'} \tag{4.9}$$

As it has already been explained in section 3.3, the semantics of the calculus builds a probabilistic transition system, in which two kinds of state appear:

- $\langle T, t\rangle$: this state represents the phase in which at most one rewrite rule, among those available in state T, can be applied to T; transitions exiting from this state are derived from inference rules 4.3 and 4.4, and each of them represents the application of at most one rewrite rule.

- $\langle T, t, \delta t\rangle$: this kind of state is reached by every transition exiting from a state $\langle T, t\rangle$; next transition, derived from rule 4.5 or 4.6, represents time advancement and entailed movement of objects.

From rule 4.3 it can be derived one transition for each tuple in B_T. Given a tuple in B_T, representing a rule application that leads up to term Tr (which cannot be congruent to T), it can be derived a transition to Tr with probability $q = P(T \to Tr)$. It could happen that the same transition could be derived in different ways, when different tuples of B_T represent applications that lead up to the same term Tr. It is worth pointing out that DeepArrange($C[T_2]$), in the premises of the rule, can never yield $\perp$, because every tuple in $\mathrm{AR}(\mathcal{R}, T)$ (which is a superset of B_T) is such that DeepArrange($C[T_2]$) $\neq\perp$ (see 4.1).

Rule 4.4 is used to derive the transition representing the situation in which no reaction occurs in the time interval. Its probability is calculated as $P(T \to T)$ and, as such, by definition 4.2 its value is the sum of $\overline{p}$ and of all the probabilities coming from any rule application which leads to T.

Rules 4.5 and 4.6, which are used to advance time by interval δt and moving objects, use relation $\xrightarrow{l}$ defined by inference rules 4.7, 4.8, 4.9. Intuitively, these three rules are used to compute new positions for all objects, by applying their own movement function to their own positions: the obtained term (which may not be well-formed) is denoted by T' in rule preconditions. Function DeepArrange is applied to T' in order to rearrange the objects and obtain a well-formed term from it. If function succeeds returning a well-formed term T'', then T'' will represent the state of the system at advanced time $t + \delta t$; otherwise, for this time interval, objects are kept in their positions.

Example 4.2.2. Continuing with previous example, figure 4.1 shows the initial fragment of transition graph originated from state $\langle T, 0\rangle$, which represents term T (from example 4.1.1) at time $t = 0$. There are three transitions exiting from state $\langle T, 0\rangle$, each one derived from a different tuple of B_T. Their probabilities, q_1, q_2, q_3, are the ones calculated previously (example 4.2.1).

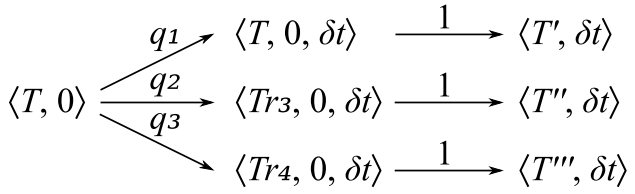


Figure 4.1: *An example transition system.*

4.3 Arranging objects

The Arrange algorithm is used to arrange all the objects contained in a looping sequence, in order to avoid collisions. Formally, Arrange takes two parameters:

1. a multiset I of terms, each of them corresponding to a single object (a simple sequence or a looping sequence) contained in the looping sequence;

2. a value $l \in \mathbb{R}$ representing the radius of the looping sequence in which the objects I are contained.

In the following, it is assumed that $p_1, \ldots, p_k$ denote the initial positions of the objects, $r_1, \ldots, r_k$ denote their radii and l is the radius of the container.

The problem is modelled by assuming an instant velocity associated with each object, whose direction and speed depends on the instant position of every object. Intuitively, these velocity are calculated considering the overlaps between each pair of objects and the collisions between each object and the containing membrane, so as to reduce the amount of conflicts. Precisely, given an object, the velocity it is subjected to is calculated as follows:

- for each other object it overlaps with, it is assumed a velocity directed opposite to the other object (the objects are trying to increase their distance) and whose magnitude is proportional to the length of the overlap; as such, the velocities due to a collision between two objects have the same magnitude for both of them.

- if the object collides with the border of the containing membrane, then a velocity directed towards the centre of the membrane is assumed; its magnitude will be proportional to the amount of the collision.

The conditions intuitively presented above can be modelled by the following system of differential equations, where $\mathbf{x_i}(t)$ denotes the position of object i at time t:

$$\begin{cases} \dfrac{d\mathbf{x_1}}{dt} &= \displaystyle\sum_{j=2}^{k} \left(s_{1j} \, \frac{\mathbf{x_1} - \mathbf{x_j}}{\|\mathbf{x_1} - \mathbf{x_j}\|} \right) - u_1 \, \frac{\mathbf{x_1}}{\|\mathbf{x_1}\|} \\ & \vdots \\ \dfrac{d\mathbf{x_i}}{dt} &= \displaystyle\sum_{\substack{j=1 \\ j \neq i}}^{k} \left(s_{ij} \, \frac{\mathbf{x_i} - \mathbf{x_j}}{\|\mathbf{x_i} - \mathbf{x_j}\|} \right) - u_i \, \frac{\mathbf{x_i}}{\|\mathbf{x_i}\|} \\ & \vdots \\ \dfrac{d\mathbf{x_k}}{dt} &= \displaystyle\sum_{j=1}^{k-1} \left(s_{kj} \, \frac{\mathbf{x_k} - \mathbf{x_j}}{\|\mathbf{x_k} - \mathbf{x_j}\|} \right) - u_k \, \frac{\mathbf{x_k}}{\|\mathbf{x_k}\|} \end{cases} \tag{4.10}$$

and where s_{ij} is the length of the overlap between the two objects i and j, and u_i is the amount of collision between an object i and its containing membrane. Their

values are defined by:

$$s_{ij} = \max\{0,\ r_i + r_j - \|\mathbf{x_i} - \mathbf{x_j}\|\} \tag{4.11}$$

$$u_i = \max\{0,\ r_i + \|\mathbf{x_i}\| - l\} \tag{4.12}$$

4.3.1 The algorithm

The Arrange algorithm is composed of two phases:

1. the objects are positioned in their initial positions, then their movement is simulated according to the system of differential equations 4.10; the simulation terminates when the objects stop moving, that is when the speed of each object becomes smaller than a constant value $\epsilon > 0$;

2. the arrangement obtained is checked to determine if there is still some collision: if there are none then the algorithm terminates returning the new arrangement, otherwise the algorithm fails returning the special value $\perp$.

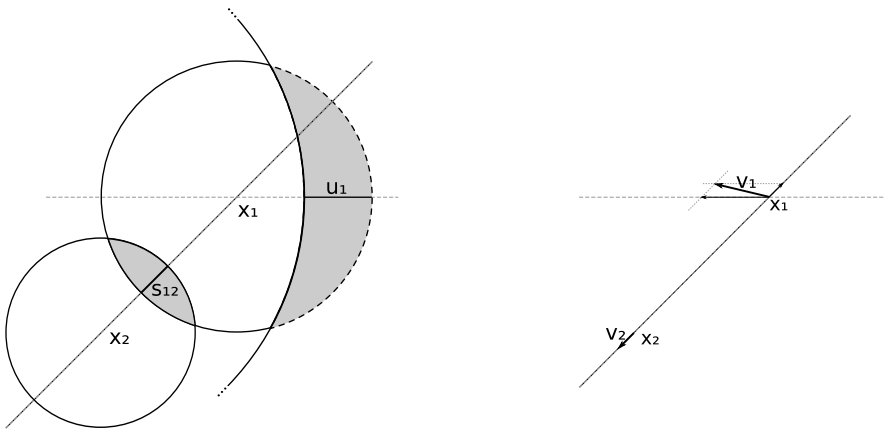
It is important to note that, during algorithm execution, it may happen that some velocities cannot be calculated. This happens in the following two situations:

1. two objects are in the same position;

2. an object bigger than the container is in its centre.

In the first case, the objects are moved along a fixed arbitrary direction. The second can simply be avoided by checking, before staring the algorithm, that the radius of every object is less than the one of the containing membrane since, if there is one, it can never be fully contained. Finally, it is worth noting that the second case can only happen if the contained object is bigger than the container; in fact, if it would be smaller, it would not collide with the container (because it is positioned in the centre of the container).

Example 4.3.1. Figure 4.2(a) shows two objects, whose centres are denoted by x_1 and x_2, that are contained in a bigger object (only partially shown). Object x_2 overlaps only with object x_2, and the amount of collision is s_{12}, while object



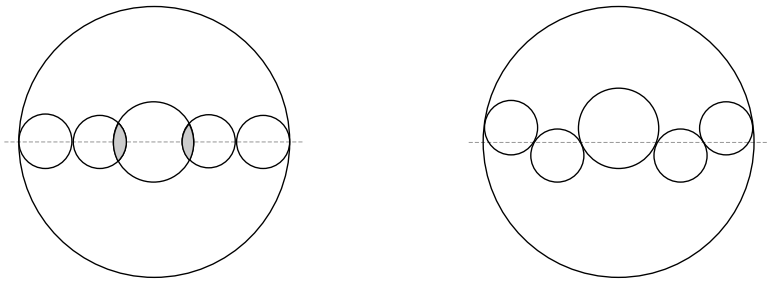
(a) (b)

Figure 4.2: *(a) An illustration of the not well-formed term from example 4.3.1, with highlighted the lengths of the two collisions; (b) the velocities associated with the objects.*

x_1 also collides with the containing membrane, and in this case the amount of collision is u_1.

In figure 4.2(b) are shown the resulting velocities, $\mathbf{v_1}$ and $\mathbf{v_2}$, to which the objects are subjected to. As it can be noted, velocity $\mathbf{v_1}$ derives from the sum of two velocities, the one due to the overlap with sibling object x_2, and the one due to collision with the containing membrane.

Example 4.3.2. This example shows a particular case in which the algorithm fails in finding a valid arrangement of the objects. The objects are initially positioned over the same line, as illustrated in figure 4.3(a). During the execution of the algorithm, the objects are moved only along the line so, since there is not enough space to put them side-by-side along the line, the algorithm eventually stops returning failure. This behaviour is correct, and an arrangement like the one shown in figure 4.3(b) could only be obtained if at least one of the object was not exactly positioned over the same line as the other objects. Finally, it's useful noting that, if the objects move (for example, by Brownian motion), then this case is extremely rare; also, the movement avoids that the objects may remain blocked forever.



(a) (b)

Figure 4.3: *(a) A particular case in which the algorithm fails in finding a valid arrangement of the objects (from example 4.3.2), because the objects are positioned along the same line; (b) a possible arrangement that can be obtained only if not all the objects are along the same line.*

4.4 Mean number of reactions

In this section, it will be shown that, for particular types of rewrite rules and when objects do not move, the mean number of reactions occurring in one time unit is equal to:

$$k_\mu \, n_{R_\mu}$$

where k_μ denotes the rate of the raction and n_{R_μ} denotes the number of ways the reaction can occur in the system. Specifically, it is assumed that:

1. The set of rewrite rules is such that, when a rule can be applied to a term, then the resulting term is the same (or at least structurally congruent to it); that is, even if a rule is applicated to the term representing the system, the system itself does not change. Formally, the set of rewrite rules $\mathcal{R}$ must be such that:

$$\forall \left([\, f_c \,] \quad P_L \overset{k,f}{\mapsto} P_R \right) \in \mathcal{R}.$$
$$\forall \tau \in \mathbf{T},\ \sigma \in \Sigma,\ T \in \mathcal{T}_{wf}. \quad T \in (P_L\tau)\sigma \Rightarrow (P_R\tau)\sigma \equiv T$$

   The rewrite rules contained in $\mathcal{R}$ are denoted by $R_1, \ldots R_M$.

2. Each object in system T does not move, that is its movement function m_0 is such that:

$$\forall p,\ l,\ t,\ \delta t. \quad m_0(p, l, t, \delta t) = p$$

The transition graph obtained for term T, since at each step the term representing the system is always the same, is:

$$\langle T, 0\rangle \xrightarrow{1} \langle T, 0, \delta t\rangle \xrightarrow{1} \langle T, 1\delta t\rangle \xrightarrow{1} \langle T, 1\delta t, \delta t\rangle \xrightarrow{1} \langle T, 2\delta t\rangle \cdots$$

At each step, time is always advanced by the same amount $\delta t = \frac{\Delta t}{m_{B_T}}$.

Given a state $\langle T, i\delta t\rangle$ and a rule R_μ, the probability that rule R_μ will be applied in the next time interval δt is independant of the exact time $i\delta t$ considered. Its value can be calculated as the sum of the probabilities deriving from each application of R_μ, each of one is described by a tuple in B_T. Hence, its value, denoted by p_μ, is equal to:

$$p_\mu = \sum_{(R_\mu, T_1 \overset{k,n}{\mapsto} T_2, C) \in B_T} \frac{k_\mu / N}{m_{B_T}}\, n$$

Let Y_t be a random variable representing the reaction occurred at time t, for $t \in \{0, 1\delta t, 2\delta t, 3\delta t, \ldots\}$, defined as:

$$Y_t = \begin{cases} \mu & \text{if at time } t \text{ reaction } R_\mu \text{ occurs, } \mu \in \{1, \ldots, M\} \\ 0 & \text{if no reaction occurs at time } t \end{cases}$$

The probability of executing a given reaction R_μ is independent of time, and equals to p_μ defined earlier; that is $P(Y_t = \mu) = p_\mu$.

Let X_μ be another random variable representing the number of reactions of type R_μ which occur in one time unit. X_μ can be seen as counting the number of successes in a sequence of independent success/failure experiment, in which the outcomes of each experiment in the sequence represent the execution, or not, of reaction R_μ at that step. Thus, X_μ follows a binomial distribution, whose success probability is p_μ and whose number of experiments is $N\, m_{B_T}$, because each time unit consists of such number of steps.

Finally, the mean number of reactions of type R_μ occurring in one time unit, is represented as the expected value of X_μ, which is:

$$E[X_\mu] = N\, m_{B_T}\, p_\mu = k_\mu\, n_{R_\mu}$$

where n_{R_μ} denotes the number of different ways in which a reaction of type R_μ can occur:

$$n_{R_\mu} = \sum_{(R_\mu, T_1 \overset{k,n}{\mapsto} T_2, C) \in B_T} n \,.$$

Chapter 5

Modelling cell cycle

This chapter shows the use of the Topological CLS to model the cell cycle in the eukaryote cells, this being the sequence of events which leads to the duplication of a mother cell into two sister cells. The model has been used for simulating, using a prototypical simulator, the proliferation of cells in a limited space.

5.1 The cell

Cells represent the main units of which living beings are made. Cells have an independent life and are able to undertake many activities, like, for example, undertaking metabolic processes (e.g. production of proteins) and responding to external stimuli. The instructions required to execute these activities are contained within the chromosomes (which are constituted by DNA molecules), and whose number may vary among different types of cells. Most cells are also able to reproduce themselves by a cell division, in which a mother cell generates two daughter cells.

Cells are separated from the external world by the cell membrane, that enables the interaction with other cells by allowing messages to pass through itself. Within the membrane there are several *organelles* scattered in the *cytoplasm*, whose characteristics depend on the cell type. In particular, there are two cell types:

- *Prokaryotic cells*: they form only unicellular beings, and are the simplest ones; their internal structure is characterized by the absence of a nucleus

(the genetic material lies around in the cytoplasm).

- *Eukaryotic cells*: they have a complex internal structure; in particular, there is a nucleus containing the chromosomes, which form the genetic material (figure 5.1).

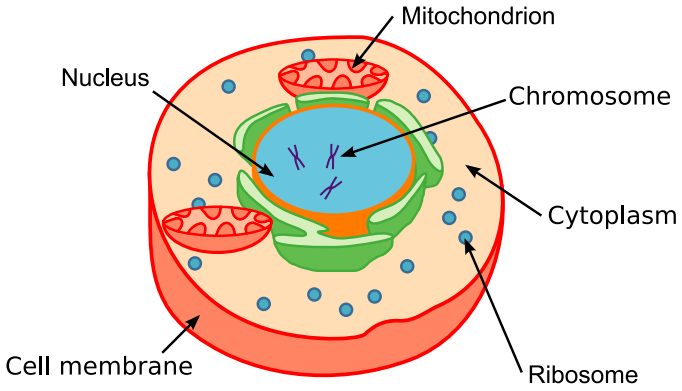


Figure 5.1: *An eukaryotic cell.*

5.1.1 Cell cycle

Cell cycle [5] represents the sequence of phases which happen within the cell, leading 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 example, the process may be stopped temporarily in the event of unfavourable ambient conditions.

Cell cycle comprises the following two phases:

1. *interphase*, is the period of time between two subsequent divisions, when a cell grows and duplicates its internal structures (in particular, this phase includes the replication of chromosomes);

2. *mitosis phase (M)*, when the nuclear division occurs and the division of the cell membrane (*cytokinesis*) generates two cells, both with their own nuclei. The mitosis phase is briefer than the interphase.

Interphase itself consists of 3 phases:

1. *G_1 phase*: the cell grows in size until it reaches the necessary size to start the division. During this phase the cell may stop its cell cycle entering a quiescent G_0 phase, in which it may remain indefinitely or temporarily;

2. *S phase (synthesis)*: during this period DNA, which is inside chromosomes, is duplicated; in particular from every chromosome there will originate two brother chromosomes;

3. *G_2 phase*: period of preparation for the following mitosis phase (M).

The duration of the various phases is not fixed, and in particular the G_1 phase is the one which may involve the most variability among the different cell types. Moreover, the mitosis phase is usually quite short in respect to the complete cell cycle. Finally, with respect to size, eucariotic cells usually range from $10\mu m$ to $100\mu m$ ([5]).

5.2 The model

The model of cell cycle that has been developed represents every cell as a membrane which contains the nucleus, which, in turn, contains the chromosomes. Only the characteristic elements involved in cell division have been covered, as they are useful for illustrating cell proliferation. This excludes the other components of the cell that do not have a central role in the process of replication.

The model describes the replication and proliferation of cells developing on a plain substrate, of limited size, and where the cells continue to duplicate as long as there is some space available. The objects are represented, therefore, as circles in space $\mathbb{R}^2$. When the cells occupy all the available space, they enter the quiescent phase (G_0 phase), stopping the cell cycle. They could re-enter into the cell cycle if some space becomes free, e.g. by removing some cells.

Different lengths of cell cycle characterize different kinds of cells. For simplicity, the model has been made assuming a total length of cell cycle process of about 24 hours, so partitioned (figure 5.2): G_1 phase 9h, S phase 8h, G_2 phase 4h, M phase 3h. The cell may also potentially enter into quiescent phase G_0 after 7h. from

the beginning of G_1 phase. The length of the various phases have been estimated according to their common relative lengths ([5]).

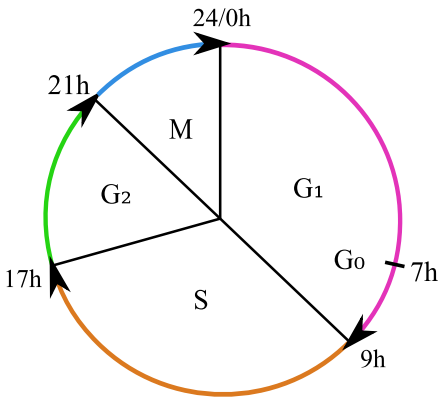


Figure 5.2: *The phases that compose cell cycle.*

5.2.1 Description

Cell membrane is modelled by a looping sequence $(m)^L$, where the elementary constituent m denotes the whole membrane. Inside it, there is the nucleus, modelled by the looping sequence $(n)^L$ or $(n_d)^L$. The two symbols n and n_d provide the ability to distinguish two different states in which the nucleus may be: n indicates that the nucleus has not yet started the duplication process, while n_d indicates that it is about to divide itself. The n_d status is kept during the entire phase of duplication of the chromosomes, until the division of the nucleus.

Finally, inside the nucleus there are the chromosomes, modelled as simple sequences of the form $(cr \cdot g1 \cdot g2 \cdot g3)$, where the g_is represent generic genes. Different chromosomes may have different number of genes. A duplicated chromosome is represented in a similar way to a normal chromosome, except that the first element of the sequence is $cr2$, and therefore it is in the form $(cr2 \cdot g1 \cdot g2 \cdot g3)$.

During system evolution, cells may change their size. In particular in the model cells have three different sizes, depending on their state, which have been chosen according to the usual size of an eukariotic cell. In detail:

- one cell in G_1 phase of its cell cycle, for which the duplication process has not yet started, has a size of $10\mu m$;
- when a cell begins the duplication process, therefore near the end of the G_1 phase, its size grows to $14\mu m$;
- after the division, that is at the end of M phase, the two sister cells have a size of $7\mu m$; this size grows during the first part of the G_1 phase up to $10\mu m$.

To simplify the model, the nucleus always keeps the same size of $3\mu m$, even after nuclear division. Chromosome size is $0.1\mu m$ (their position and size are not relevant for the purposes of the model).

In regard to object movement, only the whole cells are subject to Brownian motion, denoted by the movement function m_1 (see example 3.4.1). The nucleus and the chromosomes are, by contrast, considered motionless: as the purpose of the model is to underline the proliferation of cells in a limited space, it is unimportant to know their exact positions inside the cell and representing their movement. The Brownian motion of the cells, which is very small, is by contrast useful in order to obtain new placements of cells, which permit the growth of cells previously stopped because of lack of space. Also, Brownian motion avoids that system evolution could stop completely when situations analogous to that shown in example 4.3.2 occur.

For the nucleus and the chromosomes, as it is assumed that they do not move, the movement function associated with them, denoted by m_0, is such that:

$$\forall p,\ l,\ t,\ \delta t.\quad m_0(p, l, t, \delta t) = p\,.$$

5.3 Formalization

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

$$\begin{aligned} T = (b)^{L}_{\langle(0,0),50,m_0\rangle} \rfloor (m)^{L}_{\langle(0,0),10,m_1\rangle} \rfloor (n)^{L}_{\langle(0,-6),3,m_0\rangle} \rfloor \\ ((cr \cdot g_1 \cdot g_2 \cdot g_3)_{\langle(-1,0),0.1,m_0\rangle} \mid (cr \cdot g_4 \cdot g_5)_{\langle(-1,0),0.1,m_0\rangle}) \end{aligned} \tag{5.1}$$

Looping sequence $(b)^L$, in which the single cell is contained, allows the representation of the limited space within which the proliferation may take place. In particular the available space corresponds to a circle with a $50\mu m$ radius.

Looping sequence $(m)^L$ represents the single cell positioned in $(0,0)$ and with a radius of $10\mu m$, which indicates that it is in the G_1 phase and has not yet started the duplication process. Inside it there is the nucleus $(n)^L$, which contains, in turn, two chromosomes: $(cr \cdot g_1 \cdot g_2 \cdot g_3)$ and $(cr \cdot g_4 \cdot g_5)$.

5.3.1 Rewrite rules

The evolotion of the system is modelled by the following rewrite rules:

$$
\begin{aligned}
R_1 &: [r=7] \quad (m)^L_{\langle p,r\rangle} \rfloor X \overset{0.33}{\mapsto} (m)^L_{\langle p,10,m_1\rangle} \rfloor X \\
R_2 &: [r=10] \quad (m)^L_{\langle p,r\rangle} \rfloor X \overset{0.25}{\mapsto} (m)^L_{\langle p,14,m_1\rangle} \rfloor X \\
R_3 &: [r=14] \quad (m)^L_{\langle p,r\rangle} \rfloor (n)^L_u \rfloor X \overset{0.5}{\mapsto} (m)^L_{\langle p,r,m_1\rangle} \rfloor (n_d)^L_{\langle u,m_0\rangle} \rfloor X \\
R_4 &: \quad (n_d)^L_u \rfloor ((cr \cdot \widetilde{x})_v \mid X) \overset{0.125}{\mapsto} (n_d)^L_u \rfloor ((cr2 \cdot \widetilde{x})_v \mid X) \\
R_5 &: \quad (n_d)^L_{\langle (x,y),r\rangle} \rfloor ((cr2 \cdot \widetilde{x_1})_{v_1} \mid (cr2 \cdot \widetilde{x_k})_{v_k}) \overset{0.17}{\mapsto} \\
&\qquad (n)^L_{\langle (x-3,y),r,m_0\rangle} \rfloor ((cr \cdot \widetilde{x_1})_{\langle v_1,m_0\rangle} \mid (cr \cdot \widetilde{x_k})_{\langle v_k,m_0\rangle}) \mid \\
&\qquad (n)^L_{\langle (x+3,y),r,m_0\rangle} \rfloor ((cr \cdot \widetilde{x_1})_{\langle v_1,m_0\rangle} \mid (cr \cdot \widetilde{x_k})_{\langle v_k,m_0\rangle}) \\
R_6 &: \quad (m)^L_{\langle (x,y),r\rangle} \rfloor ((n)^L_u \rfloor X \mid (n)^L_v \rfloor Y) \overset{1}{\mapsto} \\
&\qquad ((m)^L_{\langle (x-5,y),7,m_1\rangle} \rfloor (n)^L_{\langle u,m_0\rangle} \rfloor X) \mid \\
&\qquad ((m)^L_{\langle (x+5,y),7,m_1\rangle} \rfloor (n)^L_{\langle v,m_0\rangle} \rfloor Y)
\end{aligned}
$$

The rates associated to each rewrite rule are expressed assuming a 1 hour time unit and a value of $N = 1$, so $\Delta t = 1$.

The first three rules model phase G_1 of cell cycle, rule R_4 models phase S and the last two rules model phases G_2 and M. The rates associated with the rules express the expected number of reactions occurring in each time interval Δt, when the rule is applicable. As such, a rule with rate r occurs, on average, every $\Delta t/r = 1/r$ hours (as long as it is applicable).

Rule R_1 models the first part of phase G_1, in which the cell grows after being created from its parent cell in phase M. R_1 can be applied only to a cell whose radius is 7, and increases it to 10. Its rate is 0.33, which means that this growth is expected to last for about 3 hours.

Rule R_2 also increases the radius of the cell, this time from 10 to 14. The growth to 14 means that the division process has started, and the cell will eventually

divide. It is expected to last for about 4 hours. A cell blocked in phase G_0, that cannot grow, is represented by a cell to which neither rule R_1 nor R_2 can be applied but only because there is not enough space for the bigger cell.

The application of rule R_3 signals the start of the division process for the nucleus, hence it represents the last part of phase G_1, from which the cell reaches phase S. The rule states that if the cell contains exactly one nucleus in its normal state $(n)^L$, then the nucleus may change state becoming $(n_d)^L$. Its expected duration is about $2h$.

Rule R_4 models chromosome replication which happens during phase S of cell cycle. In particular, it states that if a nucleus $(n_d)^L$ that has started replication contains a simple sequence such as $(cr \cdot \widetilde{x})$ then that sequence may become $(cr2 \cdot \widetilde{x})$. $(cr2 \cdot \widetilde{x})$ models the replicated chromosome. The associated rate is 0.125, meaning an expected duration of about 8 hours.

The next rule, R_5, models the division of the nucleus into two nuclei. This happens only when all chromosomes have been replicated. In fact, in order to be applicable, the nucleus must contain exactly two duplicated chromosomes[1]. The two nuclei obtained are like $(n)^L$, which means that they are not going to replicate further before cell division. This rule represents phase G_2 and the initial part of phase M, and as such its rate is 0.17, meaning an expected duration of 6 hours.

Finally, rule R_6 models cytokinesis, which takes place as the last operation during phase M, and from which two daughter cells originate. The rule states that if the cell contains two nuclei, then two distinct, smaller, looping sequences $(m)^L$ (with radius 7) can be obtained, both of which include only one of the two nuclei. This process is expected to last for about 1 hour.

5.3.2 Extending the model

In the model presented, nuclei inside the cells do not move, and their division occurs in a single step. Actually, the division of a nucleus into two nuclei involves the following steps:

1. after the chromosomes have been duplicated, the nuclear membrane dissolves;

[1]If nucleus would have had a different number of chromosomes, then that number of chromosomes would have appeared in the rule.

2. each duplicated chromosome is divided into two *chromatid* (single chromosomes, i.e. not duplicated), then each of two brother chromatids moves towards opposite positions of the cell;

3. a nuclear membrane is created around each group of single chromosomes, thus originating two distinct nuclei.

Finally, the process continues with the citokinesis.

As an example, the model can be extended by associating a motion to the nuclei. In particular, after nuclear division (modelled by rule R_5), each nucleus starts moving towards opposite positions of the cell. Then cytokinesis (modelled by rule R_6) only happen when the two nuclei are sufficiently distant.

This behaviour can be modelled by associating a different movement function to the nuclei which generated with rule R_5, and by constraining the applicability of rule R_6 according to the positions of the nuclei. The movement functions for the nuclei, denoted by m_W and m_E, model a linear motion, with constant speed v, directed towards, repectively, the two opposite positions q_W and q_E inside the cell (figure 5.3), defined as:

$$q_W = (-l + 4,\ 0)$$
$$q_E = (l - 4,\ 0)$$

where $l = 14$ is the radius of the container membrane. The distance between q_W and q_E is 20, while the distance from the membrane, for both positions, is 4. The updated rules are:

$$\begin{aligned} R'_5 \ : \quad & (n_d)^L_{\langle (x,y),r\rangle} \rfloor ((cr2 \cdot \widetilde{x_1})_{v_1} \mid (cr2 \cdot \widetilde{x_k})_{v_k}) \overset{0.17}{\mapsto} \\ & (n)^L_{\langle (x-3,y),r,m_W\rangle} \rfloor ((cr \cdot \widetilde{x_1})_{\langle v_1,m_0\rangle} \mid (cr \cdot \widetilde{x_k})_{\langle v_k,m_0\rangle}) \mid \\ & (n)^L_{\langle (x+3,y),r,m_E\rangle} \rfloor ((cr \cdot \widetilde{x_1})_{\langle v_1,m_0\rangle} \mid (cr \cdot \widetilde{x_k})_{\langle v_k,m_0\rangle}) \\ R'_6 \ : \ & [\, \|p_1 - p_2\| \geq 19 \,] \\ & (m)^L_{\langle (x,y),r\rangle} \rfloor ((n)^L_{\langle p_1,r_1\rangle} \rfloor X \mid (n)^L_{\langle p_2,r_2\rangle} \rfloor Y) \overset{r_6}{\mapsto} \\ & ((m)^L_{\langle (x-5,y),7,m_1\rangle} \rfloor (n)^L_{\langle u,m_0\rangle} \rfloor X) \mid \\ & ((m)^L_{\langle (x+5,y),7,m_1\rangle} \rfloor (n)^L_{\langle v,m_0\rangle} \rfloor Y) \end{aligned}$$

Constraint for rule R'_6 specifies that it becomes applicable as soon as the distance between the nuclei becomes greater than 19.

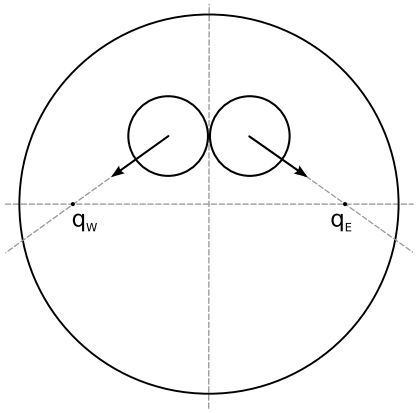


Figure 5.3: *The movement function for the nuclei.*

The following function, used for defining m_W and m_E, computes the new position reached from position p after a δt-long time interval, assuming a constant velocity directed towards position q.

$$m'(p, q, \delta t) = \begin{cases} q & \text{if } v\,\delta t \leq \|q - p\| \\ v\,\delta t\,\dfrac{q-p}{\|q-p\|} & \text{if } v\,\delta t > \|q - p\| \end{cases}$$

where v denotes the constant speed of the motion. Finally, functions m_W and m_E can be defined as:

$$m_E(p, l, t, \delta t) = m'(p, q_E, \delta t)$$
$$m_W(p, l, t, \delta t) = m'(p, q_W, \delta t)$$

Finally, it is useful to point out that the rate r_6 for rule R'_6 has to be increased from the value 1 of rule R_6, since it does no more include the time spent while nuclei are moving. Therefore, the speed of the nuclei v and the rate of rule r_6 must be chosen such that the expected time spent from the division of the nucleus to the citokinesis is still 1 hour, which is the time that R_6 was expected to last for.

5.4 Simulation

The TCLS model of cell cycle, presented in the previous section, has been used to simulate, using a prototypical simulator, the evolution of the biological system

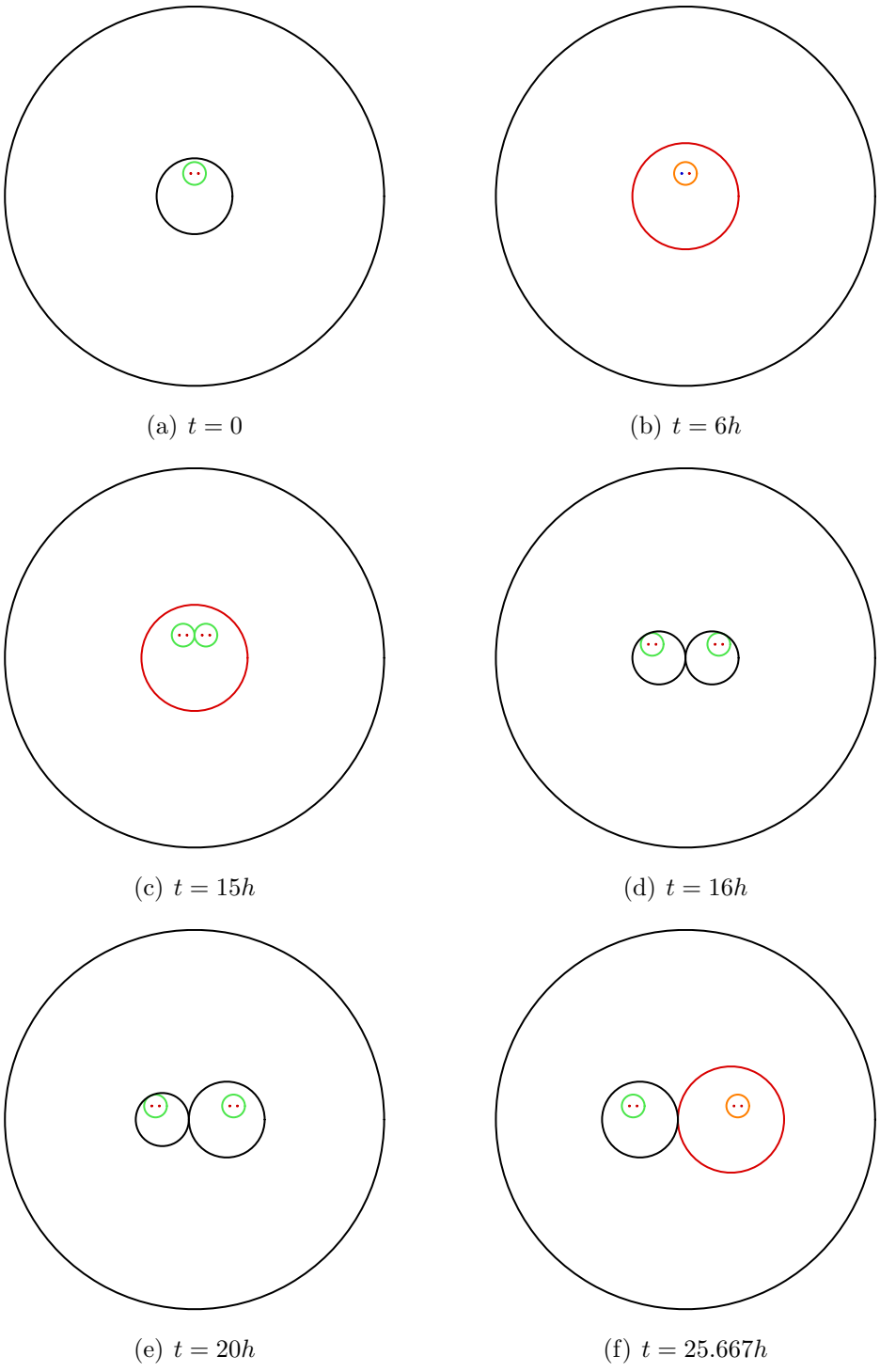
represented by term T from 5.1. The simulator is able to give a graphical representation of any well-formed term, and as such it has been used to obtain a graphical representation of the biological system after each simulation step (that is, after each time advancement).

Figure 5.4 shows the evolution of the system at certain times during the beginning of the simulation. In particular, figure 5.4(a) shows the initial status. The outer circle represents the limiting membrane, which contains a single circle representing the only cell. Inside the cell there is the nucleus, within which the two chromosomes can be found. The next figure (5.4(b)) shows the cell when it has already increased its radius to 14 and it has just replicated one of its chromosomes. Cell, nucleus and chromosomes are painted in different colours to highlight these changes: the bigger cell is red; the nuclues that has started its division process is orange; normal chromosomes are red; duplicated chromosomes are blue. Figure 5.4(c) represents the system when mitosis has just happened; the two nuclei are painted green, indicating their normal state (i.e. they are not going to replicate again during this cycle). Figure 5.4(d) shows the two daughter cells the have been generated, hence the first cell cycle is completed. Cell cycle then recommences for both cells, and in the last two figures (5.4(e) and 5.4(f)), the right cell can be seen growing and starting its replication process, while the other cell has not grown yet.

5.4.1 Results

The graph in figure 5.5 shows how the number of cells increases as time passes. This simulation shows a slow proliferation during the first 75 hours, then almost constant growth is achieved, until it completely stops after about 140 hours.

Figure 5.6(a) shows the graphical representation of the system near time 102. It can be seen that the space available is almost full and, intuitively, there are 8 cells which cannot grow because there is no room for expansion. These cells, whose cycle is blocked because the space for growing is not sufficient, can easily re-enter the cycle as some space is freed, as it will happen when other cells divide. For example, in figure 5.6(b), the cell indicated by the arrow is one of the cells that were blocked from time 102. At time 107, as the bigger central cell divides, the pointed cell eventually re-enters cell cycle (figure 5.6(c)), when it grows and



(a) $t = 0$ (b) $t = 6h$

(c) $t = 15h$ (d) $t = 16h$

(e) $t = 20h$ (f) $t = 25.667h$

Figure 5.4: *The graphical representation of the system at certain times during the beginning of the simulation.*

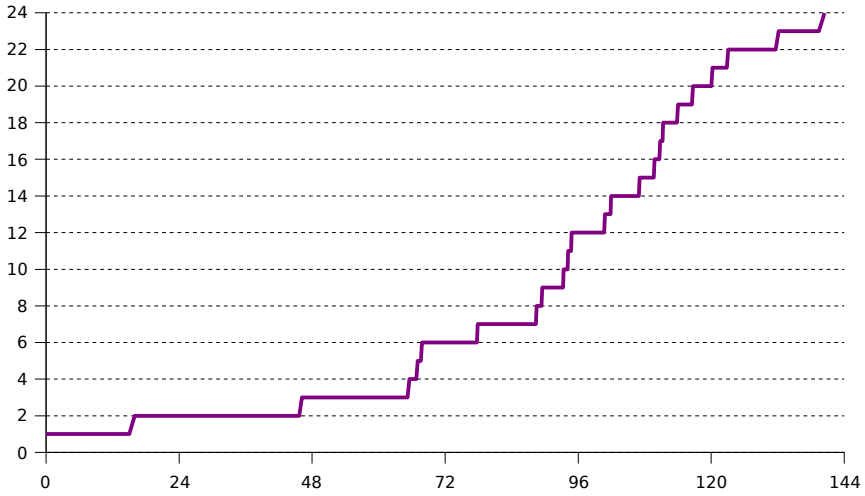
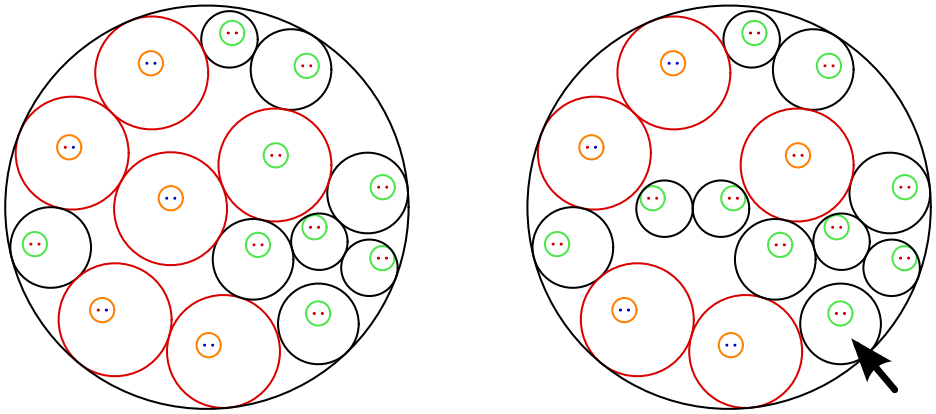


Figure 5.5: *A graph showing the number of cells in the system as time passes.*

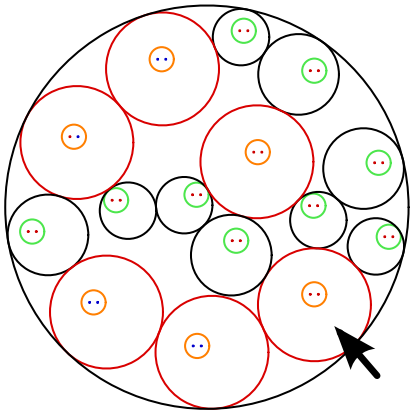
starts the division process.

Cell proliferation definitely stops at time 141 (figure 5.6(d)) when the cell cycle of all cells is blocked because they cannot grow. As it is capable of being noted, in this case there are 9 cells that have not yet reached the normal size, but their size is still the one obtained just after division.

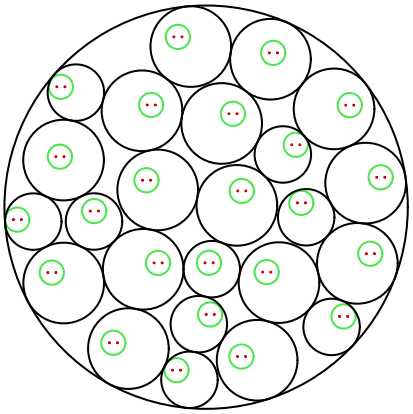


(a) $t = 102h$

(b) $t = 107h$



(c) $t = 108h$



(d) $t = 141h$

Figure 5.6: *The graphical representation of the system at certain times during simulation.*

Chapter 6

Conclusions

This thesis has presented the Topological Calculus of Looping Sequences (TCLS), an extension of the CLS that deals explicitly with space and time, and allows a faithful representation of those biological processes where knowing the position of the elements plays a central role in describing their dynamics. In particular, every object described in TCLS is represented as a *hard-sphere* with a definite position in space. Objects may also move autonomously as time passes and they are able to react when given conditions involving their positions are satisfied; for instance, an object contained inside a membrane may react with the membrane itself only when it is sufficiently close to it. Since objects are represented as hard-spheres, conflicts among them may arise and these are resolved by assuming that objects push each other if they occur to be too close.

Topological CLS, as for CLS, is based on term rewriting, so its models are described by a term, representing the biological system, and a set of rewrite rules for modelling reactions. Each rewrite rule is also enriched with a parameter specifying its *rate*, which represents the speed of the reaction described by the rule. The semantics of the calculus has been given by a *Probabilistic Transition System*, in which each state represents the status of the biological system at a certain time.

Finally, as an example application, a TCLS model of *cell cycle* has been presented and this has been used for simulating the proliferation of cells in a limited space.

As a future work, it would be useful to develop an efficient and complete simulator of TCLS which would make it possible to describe systems in its full

syntax. Nevertheless, to obtain an efficient simulator, it is likely to be necessary to restrict the permitted movement functions and to restrict the type of functions which constrain rule application; this would permit the limitation of the space in which the object may have some effect during a time interval of a certain length.

Another approach could be considered for the development of a simulator for TCLS, which involves the use of a *multi-agent system* to simulate the evolution of the biological system. This approach has been already studied in the Orion Framework [4], which permits the simulation of metabolic pathways. A multi-agent system consists of computational units, called *agents*, which are autonomous and independent from one another but able to interact among themselves. The Orion Framework is based on Hermes [12], an agent-based middleware. In Orion, each molecule composing the system is represented as an agent; moreover, there is also an agent representing the environment (a three-dimensional space) within which the molecules can move. A reaction between an enzyme and a metabolite is then represented by an interaction between the two agents involved, while the movement of a molecule involves an interaction with the agent representing the environment.

A simulator for TCLS could construct, given a term and a set of rewrite rules, a group of agents, where each agents simulates a portion of the system (for instance, it could represent a looping sequence, either with or without the contained objects). The workflow of each agent would be determined by the interaction capabilities of the portion of the system it simulates, i.e. it would be determined by the given set of rewrite rules. In this way, we could hence obtain a simulator for TCLS which exploits the inherent scalability of an agent-based approach.

Another interesting aspect to be investigated involves the ability to abstract some portions of the system if knowing the exact positions of the objects is unnecessary. This may involve developing an extension of TCLS which deeply integrates SCLS; for example, this would allow to abstract the content of a looping sequence as a SCLS term, hence avoiding the need to represent exactly, during time passage, the position and movement of the objects contained.

Bibliography

[1] The BioSPI Project. `http://www.wisdom.weizmann.ac.il/~biospi/`.

[2] The Stochastic Pi Machine (SPiM). `http://research.microsoft.com/~aphillip/spim/`.

[3] R. Alur, C. Belta, F. Ivančić, V. Kumar, M. Mintz, G. J. Pappas, H. Rubin, and J. Schug. Hybrid Modeling and Simulation of Biomolecular Networks. *Lecture Notes in Computer Science*, 2034:19–32, 2001.

[4] M. Angeletti, A. Baldoncini, N. Cannata, F. Corradini, R. Culmone, C. Forcato, M. Mattioni, E. Merelli, , and R. Piergallini. Orion: A spatial multi agent system framework for computational cellular dynamics of metabolic pathways. In *Bioinformatics ITalian Society (BITS)*, Lecture Notes in Computer Science, pages 234–270, 2006.

[5] C. J. Avers. *Molecular Cell Biology.* Addison-Wesley, 1986.

[6] P. Baldi. *Calcolo delle probabilità e statistica.* McGraw-Hill, second edition, 1998.

[7] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, P. Tiberi, and A. Troina. Stochastic Calculus of Looping Sequences for the Modeling and Simulation of Cellular Pathways.

[8] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and A. Troina. An Alternative to Gillespie's Algorithm for Simulating Chemical Reactions. In *Computational Methods in Systems Biology*, 2005.

[9] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and A. Troina. A Calculus of Looping Sequence for Modelling Microbiological Systems. *Fundamenta Informaticae*, 72(1-3):21–35, 2006.

[10] R. Barbuti, A. Maggiolo-Schettini, P. Milazzo, and A. Troina. Bisimulation Congruences in the Calculus of Looping Sequences. In *International Colloquium on Theoretical Aspects of Computing*, volume 4281 of *Lecture Notes in Computer Science*, pages 93–107. Springer, 2006.

[11] L. Cardelli. Brane Calculi. Interactions of Biological Membranes. *Lecture Notes in Computer Science*, 3082:257–280, 2005.

[12] F. Corradini and E. Merelli. Hermes: Agent-Based Middleware for Mobile Computing. In *Formal Methods for Mobile Computing*, Lecture Notes in Computer Science, pages 234–270, 2005.

[13] M. Curti, P. Degano, C. Priami, and C. T. Baldari. Modelling biochemical pathways through enhanced π-calculus. *Theor. Comput. Sci.*, 325(1):111–140, 2004.

[14] V. Danos and C. Laneve. Formal molecular biology. *Theor. Comput. Sci.*, 325(1):69–110, 2004.

[15] D. T. Gillespie. Exact Stochastic Simulation of Coupled Chemical Reactions. *The Journal of Phisical Chemistry*, 81(25), 1977.

[16] M. John, R. Ewald, and A. M. Uhrmacher. A Spatial Extension to the π-Calculus. To appear, 2007.

[17] M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic Symbolic Model Checking with PRISM: A Hybrid Approach. *International Journal on Software Tools for Technology Transfer (STTT)*, 6(2):128–142, 2004. `http://www.prismmodelchecker.org/`.

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

[19] P. Milazzo. *Qualitative and Quantitative Formal Modeling of Biological Systems*. PhD thesis, Università di Pisa, 2007.

[20] R. Milner. *Communicating and Mobile Systems: the π-Calculus*. Cambridge University Press, 1999.

[21] G. E. Monahan. A Survey of Partially Observable Markov Decision Processes: Theory, Models, and Algorithms. *Management Science*, 28(1):1–16, 1982.

[22] X. Nicollin and J. Sifakis. An Overview and Synthesis on Timed Process Algebras. In *Proceedings of the Real-Time: Theory and Practice, REX Workshop*, 1991.

[23] G. Paun. *Membrane Computing: An Introduction*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2002.

[24] D. Prandi, C. Priami, and P. Quaglia. Process Calculi in a Biological Context. *Bulletin of the EATCS*, 85:53–69, 2005.

[25] C. Priami. Stochastic π-Calculus. *The Computer Journal*, 38(7):578–589, 1995.

[26] C. Priami and P. Quaglia. Beta Binders for Biological Interactions. *Lecture Notes in Computer Science*, 3082:20–33, 2005.

[27] C. Priami, A. Regev, E. Shapiro, and W. Silverman. Application of a stochastic name-passing calculus to representation and simulation of molecular processes. *Information Processing Letters*, 80(1):25–31, October 2001.

[28] M. O. Rabin. Probabilistic Automata. *Information and Control*, 6(3):230–245, 1963.

[29] A. Regev, E. M. Panina, W. Silverman, L. Cardelli, and E. Shapiro. BioAmbients: an abstraction for biological compartments. *Theor. Comput. Sci.*, 325(1):141–167, 2004.

[30] A. Regev and E. Y. Shapiro. Cells as Computation. In *CMSB '03: Proceedings of the First International Workshop on Computational Methods in Systems Biology*, pages 1–3, London, UK, 2003. Springer-Verlag.

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

[32] S. Rosati. *Fisica generale*. Casa editrice Ambrosiana, second edition, 1994.

[33] S. Ross. *Stochastic Processes*. John–Wiley, 1983.

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