A P-Lingua based simulator for tissue P systems

Miguel A. Martínez-del-Amor, Ignacio Pérez-Hurtado *, Mario J. Pérez-Jiménez, Agustín Riscos-Núñez

Research Group on Natural Computing, Dpt. of Computer Science and Artificial Intelligence, University of Seville, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

Keywords: Membrane computing
Tissue P systems
P-Lingua
Simulators

A B S T R A C T

Investigations within the field of tissue-like P systems are being conducted, on one hand studying their computational efficiency, and on the other hand exploring the possibilities to use them as a computational modelling framework to biological phenomena.
In both cases it is necessary to develop software that provides simulation tools (simulators) for the existing variety of tissue P systems. Such simulators allow us to carry on computations of solutions to computationally hard problems on certain (small) instances. Moreover, they also provide a way to verify tissue-like models for real biological processes, by means of experimental data.
The paper presents an extension of P-Lingua (a specification language intended to become a standard for software devoted to P systems), in order to cover the class of tissue-like P systems, that were not considered in the previous release. This extension involves on one hand defining the syntax to be used, and on the other hand introducing a new built-in simulation algorithm that has been added to the core library of P-Lingua.

1. Introduction

Natural Computing is characterized by the metaphorical use of concepts, principles and mechanisms underlying natural systems [16]. Membrane Computing is an emergent branch of Natural Computing that has received important attention from the scientific community in the last decade. This unconventional paradigm of computation provides a framework for designing distributed parallel models inspired by some basic features of biological membranes. Since Păun introduced it in [9], many different classes of such computing devices, called P systems, have already been investigated. Most of them are *computationally complete/universal*, in the sense that they are equivalent in power to Turing machines, as well as *computationally efficient*, i.e., are able to trade space for time and solve in this way computational hard problems in a feasible time.

These models of computation start from the assumption that the processes taking place in the compartmental structure of a living cell can be interpreted as computations. Computational devices in membrane computing can be roughly classified in two types: (a) with the membranes arranged in a hierarchical way and delimiting regions or compartments placed in the nodes of a rooted tree, inspired from the structure of the cell (*cell-like*), and (b) with the membranes placed in the nodes of a graph, inspired from the cell inter-communication in tissues (*tissue-like*).

Since the initial[1] definition of tissue P systems [8] several research lines have been developed and new variants have arisen. One of the most interesting variants of tissue P systems was presented in [11] where the definition of tissue P systems is combined with P systems with active membranes, yielding the model of *tissue P systems with cell division*. In [11] the first

* Corresponding author.
E-mail addresses: mdelamor@us.es (M.A. Martínez-del-Amor), perezh@us.es (I. Pérez-Hurtado), marper@us.es (M.J. Pérez-Jiménez), ariscosn@us.es (A. Riscos-Núñez).

[1] Actually, the first paper introducing P systems with membranes placed in the nodes of a graph is [12].

uniform and polynomial-time solution for the SAT problem through a family of tissue P systems is given. Since then, new interesting solutions for computationally hard problems have been given in that framework [3,4].

Recently, tissue-like P systems are being used as a modelling approach to cellular systems (the quorum sensing in Vibrio Fischeri [15]), and population biology (an ecosystem related to the bearded vultures in the Pyrenees [1]). These models integrate dynamics and structural details of the different components. Bearing in mind that tissue-like P systems provide distributed parallel computing devices, it is necessary to design software applications in order to simulate such devices and to validate the tissue-like based models.

In this paper, a simulator of tissue P systems with cell division is presented based on P-Lingua which is a programming language to define P systems [5,6,18]. The authors of P-Lingua have developed a Java library that provides several services, including e.g. parsers for input files and built-in simulators. In this paper we present the P-Lingua 2.1 release, whose main novelty is that the syntax has been extended to define tissue P systems with cell division, and also the library has been updated to handle P-Lingua input files defining tissue P systems. Finally, a new built-in simulator has been added to the library in order to simulate computations of such new models.

The paper is structured as follows. In Section 2, we introduce some definitions about tissue P systems with symport/antiport rules and cell division rules. Section 3 describes the extensions for the P-Lingua programming language in order to support tissue P systems. In Section 4, we introduce the simulator for tissue P systems used in this paper, including the simulation algorithm. Section 5 is devoted to an example of simulation. Finally, conclusions and future work are discussed in Section 6.

2. Preliminaries

One of the most important roles of cells is reproduction, and this is achieved e.g. through the division of a cell into two identical copies. In cell-like P systems with active membranes [10] the central role in the computation is played by the membranes and their capability to create an exponential workspace that allows us to efficiently (in terms of number of computational steps) solve computationally hard problems. By membrane division rules two new membranes are obtained, and their contents are replicated. Active membrane systems were introduced to analyse the computational complexity of nested membranes that have the ability to divide and dissolve.

In many cases two chemicals compounds pass through a membrane at the same time, with the help of each other, either in the same direction (we have a *symport*), or in opposite directions (we have an *antiport*). Cell-like P systems with symport–antiport [7] capture these ideas.

2.1. Tissue P systems

In the framework of tissue P systems, membranes are seen as nodes of an undirected graph. The biological inspiration for this variant is twofold: cooperation between neurons and inter-cellular communication (actually, when dealing with tissue P systems we use the word *cell* instead of *membrane*). The common mathematical model of these two mechanisms is a net of processors dealing with symbols and communicating these symbols along channels specified in advance.

Communication between cells is based on symport/antiport rules. Symport rules move a number of objects present in a cell to an adjacent one (all of them are moved together and in a single step), whereas antiport rules move objects taken from two adjacent cells in opposite directions (cells interchange objects, also in a single step). Cell division rules are allowed analogously as in P systems with active membranes. The rules are used in the non-deterministic maximally parallel way, with the restriction that if a division rule is used for dividing a cell, then this cell does not participate in any other rule in this step, for division or communication.

Definition 2.1. A *tissue P system of order* $q \geqslant 1$ *with symport/antiport rules and cell division* is a tuple $\Pi = (\Gamma, \Sigma, \Omega, w_1, \ldots, w_q, R, i_{in}, i_{out})$, where:

1. Γ is a finite alphabet (called working alphabet) whose elements are called objects;
2. Σ is a finite alphabet (called input alphabet) strictly contained in Γ;
3. $\Omega \subseteq \Gamma \setminus \Sigma$ is a finite alphabet, describing the set of objects located in the environment in arbitrarily many copies each;
4. $w_1, \ldots, w_q$ are strings over Γ, describing the *multisets of objects* initially placed in the q cells of the system;
5. R is a finite set of rules, of the following forms:
   (a) *Communication rules:* $(i, u/v, j)$, for $i, j \in \{0, 1, \ldots, q\}$, $i \neq j$, and $u, v \in \Gamma^*$; $1, \ldots, q$ identify the cells of the system, while 0 is assigned to the environment.
   (b) *Division rules:* $[a]_i \to [b]_i[c]_i$, where $i \in \{1, \ldots, q\}$ and $a, b, c \in \Gamma$.
6. $i_{in} \in \{1, \ldots, q\}$ is the input cell, and $i_{out} \in \{0, 1, \ldots, q\}$ is the output cell.[2]

Rules are associated with labels, not with cells (note that there can be several cells with the same label). At each step, an object can be used by only one rule.

[2] In this paper, for the sake of simplicity, we shall consider that $i_{out} = 0$, and we will omit i_{out} in what follows.

When applying a communication rule $(i, u/v, j)$, the objects of the multiset represented by u are sent from cell i to cell j and, simultaneously, the objects of multiset v are sent from cell j to cell i (if either $i = 0$ or $j = 0$ then the communication takes place between a cell and the environment). We say that the sum of the lengths of u and v is the *length* of the rule.

When applying a division rule $[a]_i \to [b]_i[c]_i$, under the influence of object a, the cell with label i is divided in two cells with the same label; in the first copy the object a is replaced by b, in the second copy the object a is replaced by c; all other objects are replicated and copies of them are placed in the two new cells.

The rules of such a system are applied in a non-deterministic maximally parallel way as it is customary in membrane computing. In each step we apply a multiset of rules which is maximal (no further rule can be added) with the following important remark: if a cell divides, then the division rule is the only one which is applied for that cell at that step; its objects do not participate on any communication rule. In other words, before division a cell interrupts all its communication channels with the other cells and with the environment; the new cells resulting from division will interact with other cells or with the environment only at the next step – or they may divide once again. The label of a cell precisely identifies the rules which can be applied to it.

A configuration of Π is described by the multisets of objects over Γ associated with all the cells present in the system and the multiset over $\Gamma \setminus \Omega$ associated with the environment (the objects in the environment which are in finitely many copies). For two configurations C_1, C_2 of Π, we write $C_1 \Rightarrow_\Pi C_2$, and we say that we have a *transition* from C_1 to C_2, if we can pass from C_1 to C_2 by applying the rules from R according to the semantics described above.

The initial configuration of the system is $(\emptyset, w_1, \ldots, w_q)$. For each multiset m over the input alphabet, the initial configuration of the system associated with it is $(\emptyset, w_1, \ldots, w_{i_{in}} \cup m, \ldots, w_q)$. Then, m is an *input multiset* of every computation starting from such initial configuration of Π associated with m.

All computations start from an initial configuration and proceed as stated above; only halting computations give a result, which is encoded by the objects from $\Gamma \setminus \Omega$ present in the environment in the last configuration.

2.2. Recognizer tissue P systems

Recognizer cell-like P systems were introduced in [14]. They are the natural framework to study and solve decision problems within Membrane Computing, since deciding whether an instance of a given problem has an affirmative or negative answer is equivalent to decide if a string belongs to the language associated with the problem.

In the literature, recognizer cell-like P systems are associated with P systems with *input* in a natural way. The data encoding an instance of the decision problem has to be provided to the P system in order to compute the appropriate answer. This is done by codifying each instance as a multiset placed in an *input membrane*. The output of the computation (`yes` or `no`) is sent to the environment in the last step of the computation. In this way, cell-like P systems with input and external output are devices which can be seen as black boxes, in the sense that the user provides the data before the computation starts, and then waits *outside* the P system until it sends to the environment the output in the last step of the computation.

In order to use these computational devices for solving decision problems, *recognizer tissue P systems* are introduced.

Definition 2.2. A *tissue P system of order* $q \geqslant 1$ *with symport/antiport rules and cell division*, $\Pi = (\Gamma, \Sigma, \Omega, w_1, \ldots, w_q, R, i_{in})$, is a *recognizer system* if the following holds:

1. The working alphabet Γ has two distinguished objects `yes` and `no`, present in at least one copy in some initial multisets w_1, ..., w_q, but not present in Ω.
2. All computations halt.
3. For every computation of Π, either the object `yes` or the object `no` (but not both) must have been released into the environment, and only in the last step of the computation.

A computation $\mathcal{C}$ of a recognizer tissue P system with cell division is an accepting computation (respectively, rejecting computation) if the object `yes` (respectively, `no`) appears in the environment in the halting configuration of $\mathcal{C}$. Note that the last condition of the definition forbids that `yes` or `no` are sent to the environment in any previous step.

Next, we define the concept of efficiency in the framework of tissue P systems.

Definition 2.3. We say that a decision problem $X = (I_X, \theta_X)$ is solvable in polynomial time by a family $\Pi = \{\Pi(n) : n \in \mathbb{N}\}$ of recognizer tissue P systems with cell division, in a uniform way, if the following hold:

- The family Π is *polynomially uniform* by Turing machines, that is, there exists a deterministic Turing machine which constructs the system $\Pi(n)$ from $n \in \mathbb{N}$ in polynomial time with respect to n.
- There exists a pair (cod, s) of polynomial-time computable functions over I_X (called a polynomial encoding of I_X in Π) such that:
  - For each instance $u \in I_X$, $s(u)$ is a natural number and $cod(u)$ is an input multiset of the system $\Pi(s(u))$.
  - The family Π is *polynomially bounded* with regard to (X, cod, s); that is, there exists a polynomial function p, such that for each $u \in I_X$ every computation of $\Pi(s(u))$ with input $cod(u)$ is halting and, moreover, it performs at most $p(|u|)$ steps.

– The family Π is *sound* with regard to (X, cod, s); that is, for each $u \in I_X$, if there exists an accepting computation of $\Pi(s(u))$ with input $cod(u)$, then $\theta_X(u) = 1$.
– The family Π is *complete* with regard to (X, cod, s); that is, for each $u \in I_X$, if $\theta_X(u) = 1$, then every computation of $\Pi(s(u))$ with input $cod(u)$ is an accepting one.

From the soundness and completeness conditions above we deduce that every P system $\Pi(n)$ is *confluent*, in the following sense: every computation of a system with the *same* input multiset must always give the *same* answer.

We denote by $\mathbf{PMC}_{TDC}$ the set of all decision problems which can be solved by means of recognizer tissue P systems with cell division in polynomial time. This class is closed under polynomial-time reduction and under complement (see [13] for a similar result for cell-like P systems).

3. P-Lingua syntax for tissue P systems

In the version of P-Lingua presented in [6] only P systems within the *cell-like* framework were allowed to be defined in P-Lingua files. This is the starting point for the current syntax, that has been extended as explained below, in order to support tissue P systems.

3.1. Reserved words

The set of reserved words has been updated in the current syntax of the P-Lingua language to the following text strings: `call, def, let, main, @model, @mu, @ms, @d, @debug, @ceil, @floor, @log, @round, -->, <-->, #`.

3.2. Numeric expressions

Some new operators have been added in order to construct numeric expressions: `@log` (logarithm base two), `@round` (rounding), `@ceil` (rounding up), `@floor` (rounding down). For example, the string `@ceil@log m` in a P-Lingua file represents the number $\lceil log\, m \rceil$.

3.3. Model specification

As this programming language supports more than one model, it is necessary to specify in the beginning of each file which is the model of the P system defined. In this sense, P-Lingua files defining tissue P systems must begin with the sentence: `@model<tissue_psystems>`.

3.4. Definition of the initial cells

In order to define the initial cells of a tissue P system, the following sentence must be written:

```
@mu = [[ ]'1 [ ]'q]'0;
```

For example: `@mu = [[ ]'1 [ ]'2]'0;`

3.5. Definition of multisets

Initial multisets of objects for *tissue-like* cells can be defined in the same manner as initial multisets of objects for *cell-like* membranes, as indicated in [6]. Similarly, the environment label can be used to define the alphabet Ω, whose elements are handled as if they had infinite multiplicity (recall Definition 2.1). For example: `@ms(0) = a,b,c;`

3.6. Definition of rules

Two types of rules can be written in P-Lingua files which begin with the `@model<tissue_psystems>` sentence.

1. Communication rules of type $(h_1, u/v, h_2)$ are written
   `[`u`]'`h_1 `<--> [`v`]'`h_2
2. Division rules of type $[a]_h \to [b]_h[c]_h$ are written `[a]'h --> [b][c]`
   where h_1, h_2 are cell labels or the environment label, h is a cell label, u and v are multisets of objects, and a, b and c are objects.

For example:

1. $(1, b_2c/b_3^2, 0) \equiv$ `[b{2},c]'1 <--> [b{3}*2]'0`
2. $[A_1]_2 \to [B_1]_2[C_1]_2 \equiv$ `[A{1}]'2 --> [B{1}] [C{1}]`

4. A simulator software for tissue P systems

In [6], a Java library called *pLinguaCore* was presented under GPL license. It includes parsers to handle input files and built-in simulators to generate P system computations. It can export several output file formats to represent P systems. It is not a closed product because developers with knowledge of Java can add new components to the library. In this paper we present how *pLinguaCore* has been upgraded to support tissue P systems. Now, the library is able to handle input P-Lingua files which define tissue P systems and it includes a new built-in simulator in order to simulate tissue P system computations. The current version of the library can be downloaded from `http://www.p-lingua.org`.

4.1. A simulation algorithm for tissue P systems

The simulation algorithm described below generates one possible computation for a tissue P system with an initial configuration C_0 containing n cells $c_1, \ldots, c_n$. Recall that when working with recognizer P systems all computations yield the same answer (confluence). In what follows, R_{sel} denotes a set of tuples corresponding to the rules applied at a given step.

I. Initialization

1. Let C_t be the current configuration
2. Let $R_{sel} = \emptyset$
3. Let c_0 be a virtual cell (with label 0) representing the environment where all of the initial objects have infinite multiplicity

II. Selection of communication rules

1. For each *communication rule* $(i, u/v, j)$ do
   (a) For each c_{k_1} with label i do
      i. Let N be the greatest number such that the multiset of c_{k_1} contains N copies of the multiset u.
      ii. For each c_{k_2} with label j, while $N > 0$ do
         A. Let M be the greatest number $M \leqslant N$ such that the multiset of c_{k_2} contains M copies of the multiset v.
         B. Remove M copies of u from the multiset of c_{k_1}
         C. Remove M copies of v from the multiset of c_{k_2}
         D. Add $\langle c_{k_1}, c_{k_2}, (i, u/v, j), M\rangle$ to R_{sel}
         E. Let $N = N - M$

III. Selection of division rules

1. For each *division rule* $[a]_i \rightarrow [b]_i[c]_i$ do
   (a) For each c_k with label i do
      i. If a is contained in the multiset of c_k and there is no tuple in R_{sel} where c_k appears, then:
         A. Remove one instance of a from the multiset of c_k
         B. Add $\langle c_k, [a]_i \rightarrow [b]_i[c]_i\rangle$ to R_{sel}

IV. Execution of rules

1. For each tuple $\langle c_{k_1}, c_{k_2}, (i, u/v, j), M\rangle$ from R_{sel} do
   (a) Add M copies of v to the multiset of c_{k_1}
   (b) Add M copies of u to the multiset of c_{k_2}
2. For each tuple $\langle c_k, [a]_i \rightarrow [b]_i[c]_i\rangle$ from R_{sel} do
   (a) Create a new cell c'_k with label i and the same multiset of c_k
   (b) Add 1 instance of b to the multiset of c_k
   (c) Add 1 instance of c to the multiset of c'_k

V. Ending

1. If R_{sel} is not empty then:
   (a) Let $C_{t+1} = C_t$
   (b) Goto **I**
2. else, END.

Before going on, it is important to note that we assume the defined P system to be free of syntax errors that could lead to an incorrect computation, since the P-Lingua parser checks for any possible programming errors.

5. An example: a family of tissue P systems solving SAT

In order to illustrate the simulator, a uniform and efficient solution to SAT by means of a family of tissue P system of order $q \geqslant 1$ with symport/antiport rules and cell division is described in this section. This solution is a slightly modified version of the one presented in [11].

Let us consider a propositional formula $\varphi = C_1 \wedge \cdots \wedge C_m$ over n variables $x_1 \cdots x_n$, consisting of m clauses $C_j = y_{j,1} \vee \cdots \vee y_{j,k_j}$, $1 \leqslant j \leqslant m$, where $y_{j,i} \in \{x_l, \neg x_l \mid 1 \leqslant l \leqslant n\}$, $1 \leqslant i \leqslant k_j$. Without loss of generality, we may assume that no clause

contains two occurrences of some x_i or two occurrences of some $\neg x_i$ (the formula is not redundant at the level of clauses), or both x_i and $\neg x_i$ (otherwise such a clause is trivially satisfiable, hence can be removed).

We codify φ, which is an instance of SAT with size parameters n and m, by the multiset

$$cod(\varphi) = \{s_{i,j} \mid y_{j,r} = x_i,\ 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant m,\ 1 \leqslant r \leqslant k_j\} \cup \{s'_{i,j} \mid y_{j,r} = \neg x_i,\ 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant m,\ 1 \leqslant r \leqslant k_j\}.$$

In other words, we replace each variable x_i from each clause C_j with $s_{i,j}$ and each negated variable $\neg x_i$ from each clause C_j with $s'_{i,j}$, then we remove all parentheses and connectives. In this way we pass from φ to $cod(\varphi)$ in a number of steps which is linear with respect to $n \cdot m$.

The instance φ will be processed by the tissue P system $\Pi(s(\varphi))$ with input $cod(\varphi)$, where $s(\varphi) = \frac{(n+m)\cdot(n+m+1)}{2} + n$ is denoted by $\langle n, m\rangle$. We construct the recognizer tissue P system (of order 2)

$$\Pi(\langle n, m\rangle) = (\Gamma, \Sigma, \Omega, w_1, w_2, R, 2),$$

with the following components:

$$\begin{aligned}
\Gamma &= \Sigma \cup \{f, q, \texttt{yes}, \texttt{no}\} \cup \{a_i, t_i, f_i \mid 1 \leqslant i \leqslant n\} \cup \{r_i \mid 1 \leqslant i \leqslant m\} \\
&\quad \cup \{T_i, F_i \mid 1 \leqslant i \leqslant n\} \cup \{T_{i,j}, F_{i,j} \mid 1 \leqslant i \leq n, 1 \leqslant j \leqslant m+1\} \\
&\quad \cup \{b_i \mid 1 \leqslant i \leqslant 3n+m+1\} \cup \{c_i \mid 1 \leqslant i \leqslant n+1\} \\
&\quad \cup \{d_i \mid 1 \leqslant i \leqslant 3n+nm+2m+1\} \\
&\quad \cup \{e_i \mid 1 \leqslant i \leqslant 3n+nm+2m+3\}, \\
\Sigma &= \{s_{i,j}, s'_{i,j} \mid\ 1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant m\}, \\
\Omega &= \Gamma - \{\texttt{yes}, \texttt{no}\}, \\
w_1 &= \texttt{yes no}\ b_1\ c_1\ d_1\ e_1, \\
w_2 &= f\ a_1\ a_2\ \ldots\ a_n.
\end{aligned}$$

The set of rules R consists of:

- **Division rules:**
  $r_{1,i} \equiv [a_i]_2 \to [T_i]_2[F_i]_2$ for $1 \leqslant i \leqslant n$
- **Communication rules:**
  $r_{2,i} \equiv (1, b_i/b_{i+1}^2, 0)$ for $1 \leqslant i \leqslant n$
  $r_{3,i} \equiv (1, c_i/c_{i+1}^2, 0)$ for $1 \leqslant i \leqslant n$
  $r_{4,i} \equiv (1, d_i/d_{i+1}^2, 0)$ for $1 \leqslant i \leqslant n$
  $r_{5,i} \equiv (1, e_i/e_{i+1}, 0)$ for $1 \leqslant i \leqslant 3n+nm+2m$
  $r_6 \equiv (1, b_{n+1}c_{n+1}d_{n+1}/f, 2)$
  $r_{7,i} \equiv (2, c_{n+1}T_i/c_{n+1}T_{i,1}, 0)$ for $1 \leqslant i \leqslant n$
  $r_{8,i} \equiv (2, c_{n+1}F_i/c_{n+1}F_{i,1}, 0)$ for $1 \leqslant i \leqslant n$
  $r_{9,ij} \equiv (2, T_{i,j}/t_iT_{i,j+1}, 0)$ for $1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant m$
  $r_{10,ij} \equiv (2, F_{i,j}/f_iF_{i,j+1}, 0)$ for $1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant m$
  $r_{11,i} \equiv (2, b_i/b_{i+1}, 0)$ for $n+1 \leqslant i \leqslant 3n+m$
  $r_{12,i} \equiv (2, d_i/d_{i+1}, 0)$ for $n+1 \leqslant i \leqslant 3n+m$
  $r_{13,ij} \equiv (2, b_{3n+m+1}t_is_{i,j}/b_{3n+m+1}r_j, 0)$ for $1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant m$
  $r_{14,ij} \equiv (2, b_{3n+m+1}f_is'_{i,j}/b_{3n+m+1}r_j, 0)$ for $1 \leqslant i \leqslant n,\ 1 \leqslant j \leqslant m$
  $r_{15,i} \equiv (2, d_i/d_{i+1}, 0)$ for $3n+m+1 \leqslant i \leqslant 3n+nm+m$
  $r_{16,i} \equiv (2, d_{3n+nm+m+i}r_i/d_{3n+nm+m+i+1}, 0)$ for $1 \leqslant i \leqslant m$
  $r_{17} \equiv (2, d_{3n+nm+2m+1}/q\ \texttt{yes}, 1)$
  $r_{18} \equiv (1, e_{3n+nm+2m+1}/e_{3n+nm+2m+2}q, 0)$
  $r_{19} \equiv (1, e_{3n+nm+2m+2}/e_{3n+nm+2m+3}, 0)$
  $r_{20} \equiv (2, \texttt{yes}/\lambda, 0)$
  $r_{21} \equiv (1, e_{3n+nm+2m+3}q\ \texttt{no}/\lambda, 2)$
  $r_{22} \equiv (2, \texttt{no}/\lambda, 0)$

5.1. A P-Lingua source code

This section shows the P-Lingua source code that defines a tissue P system belonging to the family specified above. In particular, we shall consider for example the following formula as the instance of SAT that we try to solve:

$$\varphi = (x_1 \vee x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee x_2) \wedge (x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_2 \vee \neg x_3 \vee x_4)$$
$$\wedge(\neg x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \neg x_3 \vee x_4) \wedge (x_1 \vee x_4) \wedge (\neg x_1 \vee x_2 \vee x_3 \vee x_4).$$

Of course, the `main` module can be easily modified in order to define any other P system of the family.

The source code is structured as follows:

1. Module `main()` that defines a recognizer tissue P system solving the SAT problem for the formula described above with 4 variables and 8 clauses. Firstly, it calls the module `sat_tissue(n,m)` for $(n, m) \equiv (4, 8)$. Secondly, it extends the initial multiset of the input cell with $cod(\varphi)$ where objects $s_{i,j}$ are written `s{i,j}` representing that variable x_i is present in clause C_j, and objects $s'_{i,j}$ are written `sp{i,j}` representing that variable $\neg x_i$ is present in clause C_j.
2. Module `sat_tissue(n,m)` that defines the skeleton for a tissue P system solving the SAT problem for any instance with n variables and m clauses.
3. Module `init_cells()` that defines the initial cells of the tissue P system, determining as well the label 0 for the environment.
4. Module `init_rules(n,m)` that defines the rules of the tissue P system.
5. Module `init_multisets(n)` that defines the initial multisets of the cells.
6. Module `init_environment(n,m)` that defines the initial multiset of the environment.

Source code:

```
@model<tissue_psystems>

def main()
{
 /* tissue P system skeleton */
 call sat_tissue(4,8);
 /* formula */
 @ms(2) += s{1,1},s{2,1},sp{3,1},s{4,1},
           sp{1,2},s{2,2},
           s{1,3},sp{2,3},sp{3,3},
           s{2,4},sp{3,4},s{4,4},
           sp{1,5},s{2,5},s{4,5},
           s{2,6},sp{3,6},s{4,6},
           s{1,7},s{4,7},
           sp{1,8},s{2,8},s{3,8},s{4,8};
}

def sat_tissue(n,m)
{
  call init_cells();
  call init_multisets(n);
  call init_environment(n,m);
  call init_rules(n,m);
}

def init_cells()
{
  @mu = [[]'1 []'2]'0;
}

def init_rules(n,m)
{
  /* r1 */ [a{i}]'2 --> [T{i}] [F{i}] : 1<=i<=n;
  {
    /* r2 */ [b{i}]'1 <--> [b{i+1}*2]'0;
    /* r3 */ [c{i}]'1 <--> [c{i+1}*2]'0;
    /* r4 */ [d{i}]'1 <--> [d{i+1}*2]'0;
  } : 1<=i<=n;
  /* r5 */ [e{i}]'1 <--> [e{i+1}]'0   : 1<=i<=3*n+n*m+2*m;
  /* r6 */ [b{n+1},c{n+1},d{n+1}]'1 <--> [f]'2;
```

```
  {
    /* r7 */ [c{n+1},T{i}]'2 <--> [c{n+1},T{i,1}]'0;
    /* r8 */ [c{n+1},F{i}]'2 <--> [c{n+1},F{i,1}]'0;
  } : 1<=i<=n;
  {
    /* r9 */  [T{i,j}]'2 <--> [t{i},T{i,j+1}]'0;
    /* r10 */ [F{i,j}]'2 <--> [f{i},F{i,j+1}]'0;
  } : 1<=i<=n,1<=j<=m;
  {
    /* r11 */ [b{i}]'2 <--> [b{i+1}]'0;
    /* r12 */ [d{i}]'2 <--> [d{i+1}]'0;
  } : n+1<=i<=(n+1)+(2*n+m)-1;
  {
    /* r13 */ [b{3*n+m+1},t{i},s{i,j}]'2 <--> [b{3*n+m+1},r{j}]'0;
    /* r14 */ [b{3*n+m+1},f{i},sp{i,j}]'2 <--> [b{3*n+m+1},r{j}]'0;
  } : 1<=i<=n,1<=j<=m;
  /* r15 */ [d{i}]'2 <--> [d{i+1}]'0 : 3*n+m+1<=i<=3*n+n*m+m;
  /* r16 */ [d{3*n+n*m+m+i},r{i}]'2 <--> [d{3*n+n*m+m+i+1}]'0 : 1<=i<=m;
  /* r17 */ [d{3*n+n*m+2*m+1}]'2 <--> [yes,q]'1;
  /* r18 */ [e{3*n+n*m+2*m+1}]'1 <--> [e{3*n+n*m+2*m+2},q]'0;
  /* r19 */ [e{3*n+n*m+2*m+2}]'1 <--> [e{3*n+n*m+2*m+3}]'0;
  /* r20 */ [yes]'2 <--> [#]'0;
  /* r21 */ [e{3*n+n*m+2*m+3},no,q]'1 <--> [#]'2;
  /* r22 */ [no]'2 <--> [#]'0;
}

def init_multisets(n)
{
  @ms(1) = yes,no,b{1},c{1},d{1},e{1};
  @ms(2) = f;
  @ms(2) += a{i} : 1<=i<=n;
}

def init_environment(n,m)
{
  @ms(0) = f,q;
  @ms(0) += s{i,j},sp{i,j} : 1<=i<=n,1<=j<=m;
  @ms(0) += a{i},t{i},f{i} : 1<=i<=n;
  @ms(0) += r{i} : 1<=i<=m;
  @ms(0) += T{i},F{i} : 1<=i<=n;
  @ms(0) += T{i,j},F{i,j} : 1<=i<=n,1<=j<=m+1;
  @ms(0) += b{i} : 1<=i<=3*n+m+1;
  @ms(0) += c{i} : 1<=i<=n+1;
  @ms(0) += d{i} : 1<=i<=3*n+n*m+2*m+1;
  @ms(0) += e{i} : 1<=i<=3*n+n*m+2*m+3;
}
```

5.2. Simulation results

The *pLinguaCore* Java library includes a command-line interface in order to parse P-Lingua input files and simulate the defined P systems. The recognizer tissue P system defined above can be simulated by writing the next command[3] in a system console:

```
java -jar plinguacore.jar plingua_sim -pli sat_tissue.pli -o output.txt
```

A complete explanation of commands for *pLinguaCore* can be found in [6]. In this case, the P-Lingua input file is called `sat_tissue.pli` and it contains the source code presented in Section 5.1. The file `output.txt` is a text file where information about the parser process and the generated computation is stored:

[3] A Java runtime environment 1.6.0 or better must be installed. It can be downloaded from `http://www.java.com`.

1. Initial cells
2. Initial multisets
3. Rules set
4. For each configuration:
   (a) Multiset of objects in the environment
   (b) Multiset of objects for each cell
   (c) Rules selected to be executed in the next step

The simulator runs until reaching a halting configuration, where no rule can be selected to be executed in the next step. At this stage, the environment contains an object `yes` or `no`.

For the P system defined in the file `sat_tissue.pli`, an object `yes` is sent to the environment after 63 steps of computation and it halts. That is, it is an *accepting* computation.

6. Conclusions and future work

In this paper we present the new release P-Lingua 2.1, that extends significantly the previous version by incorporating the possibility to work with tissue P systems. Besides, a new simulation algorithm has been designed and implemented, taking into account the special features of tissue P systems with symport/antiport rules and cell division. This new simulator has been included into the library *pLinguaCore*, and it has been checked by simulating two families of tissue P systems taken from the literature, namely, uniform solutions to `SAT` and `3-COL`. The first one has been included in the paper, and the second one is omitted because of the limitation of space.

A possible line for future work is to design a simulator for the probabilistic variants of tissue P systems, where each rule has a constant (probability) associated with it. Furthermore, we are also interested in passing from the classical sequential architecture available in most computers to a parallel architecture such as the modern programmable GPUs.

Acknowledgements

The authors acknowledge the support of the project TIN2009-13192 of the Ministerio de Ciencia e Innovación of Spain, cofinanced by FEDER funds, and the support of the Project of Excellence with *Investigador de Reconocida Valía* of the Junta de Andalucía, grant P08-TIC-04200.

References

[1] M. Cardona, M.A. Colomer, M.J. Pérez-Jiménez, D. Sanuy, A. Margalida, Modeling ecosystem using P systems: the bearded vulture, a case study, Lecture Notes in Comput. Sci., 5391 (2009) 137–156.
[2] D. Díaz-Pernil, Sistemas celulares de tejidos: formalización y eficiencia computacional, Ph.D. Thesis, University of Sevilla, 2008.
[3] D. Díaz-Pernil, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, Solving the independent set problem by using tissue-like P systems with cell division, Lecture Notes in Comput. Sci. 5601 (2009) 213–222.
[4] D. Díaz-Pernil, M.A. Gutiérrez-Naranjo, M.J. Pérez-Jiménez, A. Riscos-Núñez, A uniform family of tissue P systems with cell division solving 3-COL in a linear time, Theoret. Comput. Sci. 404 (1–2) (2008) 76–87.
[5] D. Díaz-Pernil, I. Pérez-Hurtado, M.J. Pérez-Jiménez, A. Riscos-Núñez, A P-Lingua programming environment for Membrane Computing, Lecture Notes in Comput. Sci. 5391 (2009) 187–203.
[6] M. García-Quismondo, R. Gutiérrez-Escudero, I. Pérez-Hurtado, M.J. Pérez-Jiménez, A. Riscos-Núñez, An overview of P-Lingua 2.0, Lecture Notes in Comput. Sci. 5957 (2010) 264–288.
[7] C. Martín-Vide, Gh. Păun, G. Rozenberg, Membrane systems with carriers, Theoret. Comput. Sci. 270 (2002) 779–796.
[8] C. Martín-Vide, J. Pazos, Gh. Păun, A. Rodríguez-Patón, Tissue P systems, Theoret. Comput. Sci. 296 (2003) 295–326.
[9] Gh. Păun, Computing with membranes, J. Comput. Syst. Sci. 61 (1) (2000) 108–143 (and Turku Center for Computer Science-TUCS Report No. 2008, 1998).
[10] Gh. Păun, P systems with active membranes: attacking NP-complete problems, J. Automat. Lang. Comb. 6 (1) (2001) 75–90.
[11] Gh. Păun, M.J. Pérez-Jiménez, A. Riscos-Núñez, Tissue P systems with cell division, Int. J. Comput. Commun. Control III (3) (2008) 295–303 (A preliminary version in Gh. Păun, A. Riscos-Núñez, A. Romero-Jiménez, F. Sancho-Caparrini (Eds.), Second Brainstorming Week on Membrane Computing, Sevilla, Report RGNC 01/2004, 2004, pp. 380–386).
[12] Gh. Păun, Y. Sakakibara, T. Yokomori, P systems on graphs of restricted forms, Publ. Math. Debrecen 60 (2002) 635–660.
[13] M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, Complexity classes in cellular computing with membranes, Nat. Comput. 2 (3) (2003) 265–285.
[14] M.J. Pérez-Jiménez, A. Romero-Jiménez, F. Sancho-Caparrini, A polynomial complexity class in P systems using membrane division, J. Automat. Lang. Comb. 11 (4) (2006) 423–434.
[15] F.J. Romero, M.J. Pérez-Jiménez, A model of the Quorum Sensing System in Vibrio Fischeri using P systems, Artificial Life 14 (1) (2008) 95–109.
[16] G. Rozenberg, DNA processing in ciliates. The wonders of DNA computing in vivo, in: I. Antoniou, C.S. Calude, M.J. Dinneen (Eds.), Unconventional Models of Computation, UMC'2K, Springer, 2000, pp. 116–118.
[17] The P Systems. <http://www.ppage.psystems.eu/>.
[18] The P-Lingua. <http://www.p-lingua.org/>.