Solving the ST-Connectivity Problem with Pure Membrane Computing Techniques

Zsolt Gazdag[1], Miguel A. Gutiérrez–Naranjo[2]

[1]Department of Algorithmics and their Applications
Faculty of Informatics
Eötvös Loránd University, Hungary
gazdagzs@inf.elte.hu

[2]Research Group on Natural Computing
Department of Computer Science and Artificial Intelligence
University of Sevilla, 41012, Spain
magutier@us.es

Summary. In Membrane Computing, the solution of a decision problem X belonging to the complexity class $\mathbf{P}$ via a polynomially uniform family of recognizer P systems is trivial, since the polynomial encoding of the input can involve the solution of the problem. The design of such solution has one membrane, two objects, two rules and one computation step. *Stricto sensu*, it is a solution in the framework of Membrane Computing, but it does not use Membrane Computing strategies. In this paper, we present three designs of uniform families of P systems that solve the decision problem STCON by using Membrane Computing strategies (*pure* Membrane Computing techniques): P systems with membrane creation, P systems with active membranes with dissolution and without polarizations and P systems with active membranes without dissolution and with polarizations. Since STCON is $\mathbf{NL}$-complete, such designs are *constructive* proofs of the belonging of $\mathbf{NL}$ to $\mathbf{PMC}_{\mathcal{MC}}$, $\mathbf{PMC}_{\mathcal{AM}^0_{+d}}$ and $\mathbf{PMC}_{\mathcal{AM}^+_{-d}}$.

1 Introduction

Membrane Computing [13] is a well-established model of computation inspired by the structure and functioning of cells as living organisms able to process and generate information. It starts from the assumption that the processes taking place in the compartmental structures as living cells can be interpreted as computations. The devices of this model are called *P systems*.

Among the different research lines in Membrane Computing, one of the most vivid is the search of frontiers between complexity classes of decision problems, i.e., to identify collections of problems that can be solved (or languages that can be decided) by families of P systems with *similar* computational resources. In order to settle the correspondence between complexity classes and P system families, recognizer P systems were introduced in [9, 10]. Since then, recognizer P systems are the natural framework to study and solve decision problems within Membrane Computing.

In the last years, many papers have been published about the problem of deciding if a uniform family of recognizer P systems of type $\mathcal{F}$ built in polynomial time is able to solve the decision problem X . This is usually written as the problem of deciding if X belongs to $\mathbf{PMC}_{\mathcal{F}}$ or not. It has been studied for many P system models $\mathcal{F}$ and for many decision problems X (see, e.g., [2, 3, 4, 5] and references therein).

The solution of a decision problem X belonging to the complexity class $\mathbf{P}$ via a polynomially uniform family of recognizer P systems is trivial[1], since the polynomial encoding of the input can involve the solution of the problem. On the one hand, by definition, $X \in \mathbf{P}$ if there exists a deterministic algorithm A working in polynomial time that *solves* X. On the other hand, the belonging of X to $\mathbf{PMC}_{\mathcal{F}}$ requires a polynomial time mapping *cod* that encodes the instances u of the problem X as multisets wich will be provided as inputs. Formally, given a decision problem X and an algorithm A as described above, two different functions *s (size)* and *cod (encoding)* can be defined for each instance u of the decision problem:

- $s(u) = 1$, for all u
- $cod(u) = \begin{cases} yes & \text{if } A(u) = yes \\ no & \text{if } A(u) = no. \end{cases}$

The family of P systems which solves X is $\boldsymbol{\Pi} = \{\Pi(n)\}_{n\in\mathbb{N}}$ with

$$\Pi(n) = \langle \Gamma, \Sigma, H, \mu, w, \mathcal{R}, i\rangle$$

- *Alphabet:* $\Gamma = \{yes, no\}$
- *Input alphabet:* $\Sigma = \Gamma$
- *Set of labels:* $H = \{skin\}$
- *Membrane structure:* $[\,]_{skin}$
- *Initial multisets:* $w = \emptyset$
- *Input label:* $i = skin$
- *Set of rules:* $[\,yes\,]_{skin} \to yes\,[\,]_{skin}$ and $[\,no\,]_{skin} \to no\,[\,]_{skin}$. Both are send-out rules.

Trivially, for all instance u of the problem, $\Pi(s(u)) + cod(u)$ provides the right solution in one computation step. *Stricto sensu*, it is a solution in the framework of Membrane Computing, but it does not use Membrane Computing strategies. All the work is done in the algorithm A and one can wonder if the computation itself can be performed by using *pure* Membrane Computing techniques.

We focus now on the well-known ST-Connectivity problem (known as STCON). It can be settled as follows: *Given a directed graph* $\langle V, E\rangle$ *and two*

[1] See [8, 11].

vertex s and t in V, the STCON *problem consists on deciding if t is reachable from s, i.e., if there exists a sequence of adjacent vertices (i.e., a path) which starts with s and ends with t.* It is known that it is an **NL**-complete problem, i.e., it can be solved by a nondeterministic Turing machine using a logarithmic amount of memory space and every problem in the class **NL** is reducible to STCON under a log-space reduction.

In this paper, we study the STCON in the framework of P systems. As shown above, since STCON $\in$ **NL** $\subseteq$ **P**, there exist a trivial family of P systems in $\mathbf{PMC}_{\mathcal{F}}$ which solves it, regardless the model $\mathcal{F}$. It suffices that $\mathcal{F}$ deals with *send-out* rules. In this paper, we present three designs of uniform families of P systems that solve the decision problem STCON by *pure* Membrane Computing techniques, i.e., techniques where the features of the model $\mathcal{F}$ are exploited in the computation: P systems with membrane creation, P systems with active membranes with dissolution and without polarizations and P systems with active membranes without dissolution and with polarizations. We provide such designs and show the differences with previous studies found in the literature.

Since STCON is **NL**-complete, such designs are *constructive* proofs of the belonging of **NL** to $\mathbf{PMC}_{\mathcal{MC}}$, $\mathbf{PMC}_{\mathcal{AM}^0_{+d}}$ and $\mathbf{PMC}_{\mathcal{AM}^+_{-d}}$.

The paper is structured as follows: First of all, we recall some basic definitions used along the paper. In Section 3, previous works on **NL** are revisited. Next, our designs of solutions are provided and the paper finishes with some conclusions and presenting research lines for a future work.

2 Preliminaries

Next, some basic concepts used along the paper are recalled. We assume that the reader is familiar with Membrane Computing techniques (for a detailed description, see [13]).

A decision problem, X, is a pair (I_X, θ_X) such that I_X is a language over a finite alphabet (whose elements are called *instances*) and θ_X is a total Boolean function over I_X. A *P system with input* is a tuple (Π, Σ, i_Π), where Π is a P system, with working alphabet Γ, with p membranes labelled by $1, \ldots, p$, and initial multisets $\mathcal{M}_1, \ldots, \mathcal{M}_p$ associated with them; Σ is an (input) alphabet strictly contained in Γ; the initial multisets are over $\Gamma - \Sigma$; and i_Π is the label of a distinguished (input) membrane. Let (Π, Σ, i_Π) be a P system with input, Γ be the working alphabet of Π, μ its membrane structure, and $\mathcal{M}_1, \ldots, \mathcal{M}_p$ the initial multisets of Π. Let m be a multiset over Σ. The *initial configuration of* (Π, Σ, i_Π) *with input* m is $(\mu, \mathcal{M}_1, \ldots, \mathcal{M}_{i_\Pi} \cup m, \ldots, \mathcal{M}_p)$. We denote by I_Π the set of all inputs of the P system Π (i.e. I_Π is a collection of multisets over Σ). In the case of P systems with input and *with external output*, the above concepts are introduced in a similar way.

Definition 1. *A* recognizer P system *is a P system with input, (Π, Σ, i_Π), and with external output such that:*

1. *The working alphabet contains two distinguished elements yes, no.*
2. *All its computations halt.*
3. *If $\mathcal{C}$ is a computation of Π, then either some object yes or some object no (but not both) must have been released into the environment, and only in the last step of the computation. We say that $\mathcal{C}$ is an accepting computation (respectively, rejecting computation) if the object yes (respectively, no) appears in the external environment associated to the corresponding halting configuration of $\mathcal{C}$.*

3 Previous Works

The relation between the complexity class **NL** and Membrane Computing models has already been explored in the literature. In [6], Murphy and Woods claim that $\mathbf{NL} \subseteq \mathbf{PMC}_{\mathcal{AM}^0_{-d,-u}}$, i.e., every problem in the complexity class **NL** can be solved by a non-uniform family of recognizer P systems with active membranes without polarization and without dissolution.

The proof shows the design of a family of P systems with active membranes without polarization and without dissolution which solves STCON and considers the **NL**-completeness of STCON. Nonetheless, the authors use a non standard definition of recognizer P systems. According to the usual definition of *recognizer P system* (see, e.g., [4]), *either one object yes or one object no (but no both) must have been released into the environment, and only in the last step of the computation.* In the proposed family by Murphy and Woods, it is easy to find a P system which sends *yes* to the environment in an intermediate step of the computation and sends *no* to the environment in the last step of the computation, so their proof of $\mathbf{NL} \subseteq \mathbf{PMC}_{\mathcal{AM}^0_{-d,-u}}$ cannot be considered valid with respect to the standard definition of recognizer P systems.

Counterexample: Let us consider the instance (s, t, G) of STCON where G has only two vertices s and t and only one edge (s, t). According to [6], the P system of the cited model that solves this instance has $\Gamma = \{s, t, yes, no, c_0, \ldots, c_4\}$ as alphabet, h as unique label and $[\,]_h$ as membrane structure. The initial configuration is $[s\, c_4]_h$ and the set of rules consists of the following seven rules:

$$\begin{array}{ll} [s \to t]_h & [t]_h \to [\,]_h\, yes \\ [c_0]_h \to [\,]_h\, no & [c_i \to c_{i-1}]_h \text{ for } i \in \{1, \ldots, 4\}. \end{array}$$

It easy to check that this P system sends *yes* to the environment in the second step of computation and sends *no* in the fifth (and last) step, so, according to the standard definition, it is not a recognizer P system. In [7] Murphy and Woods revisited the solution of STCON by non-uniform families of recognizer P systems and considered three different ways of the acceptance in recognizer P systems, one of them was the standard one (Def. 1).

4 Three Designs for the STCON Problem

In this section, we provide three uniform families of P systems that solve the STCON problem in three different P system models. All these models use the same encoding for an instance of the problem. We do not loss generality if we consider the n vertices of the graph as $\{1, \ldots, n\}$. In this case, a concrete instance $I = (s, t, \langle V, E \rangle)$ of the STCON on a graph $\langle V, E \rangle$ with vertices $\{1, \ldots, n\}$, can be encoded as

$$cod(I) = \{x_s, y_t\} \cup \{a_{ij} \, : \, (i, j) \in E\},$$

i.e., x_s stands for the starting vertex, y_t for the ending vertex and a_{ij} for each edge (i, j) in the graph. By using this coding, *all* the instances of the STCON problem with n variables, can be encoded with the alphabet

$$\begin{aligned} \Sigma = & \{x_i \, : \, i \in \{1, \ldots, n\}\} \cup \\ & \{y_j \, : \, j \in \{1, \ldots, n\}\} \cup \\ & \{a_{ij} \, : \, i, j \in \{1, \ldots, n\}\} \end{aligned}$$

whose cardinality is $2n + n^2$.

Next we present three solutions of the STCON problem by P systems. The first two solutions are based on P systems with active membranes, the last one uses P systems with membrane creation. The first solution does not use membrane dissolution but uses the polarizations of the membranes. The second solution does not use polarizations but uses membrane dissolution instead. Moreover, none of these solutions use membrane division rules.

All the three solutions, roughly speaking, work in the following way. For a given directed graph $G = (V, E)$ and vertices s and t, the system creates/activates certain membranes in the initial configuration corresponding to the edges in E. Then, these membranes will be used to create those objects that represent the vertices reachable from s. Meanwhile, it is tested whether or not the vertex t is created or not. If yes, the system initiates a process which will send *yes* out to the environment. If the vertex t is not produced by the system, i.e., t is not reachable from s in G, then a counter will create the symbol *no* which is then sent out to the environment.

4.1 P Systems with Active Membranes, with Polarization and without Dissolution

As a first approach, we will provide the design of a uniform family $\mathbf{\Pi} = \{\Pi_n\}_{n \in \mathbb{N}}$ of P systems in $\mathbf{PMC}_{\mathcal{AM}_{-d}}$ which solves STCON. Each P system Π_n of the family decides on *all the possible* instances of the STCON problem on a graph with n nodes. Such P systems use two polarizations, but they do not use division or dissolution rules, so *not all the types of rules* of P systems with active membranes are necessary to solve STCON. Each Π_n will receive as input an instance of the STCON as described above and will release *yes* or *no* into the environment in the last step of the computation as the answer of the decision problem. The family presented here is

$$\Pi_n = \langle \Gamma_n, \Sigma_n, H_n, EC_n, \mu_n, w_n^a, w_n^1, \dots, w_n^n, w_n^{11}, \dots, w_n^{nn}, w_n^{skin}, \mathcal{R}_n, i_n \rangle.$$

For the sake of simplicity, thereafter we will omit the subindex n.

- **Alphabet:**

$$\begin{aligned} \Gamma = \{ & x_i, y_i, t_i \,:\, i \in \{1, \dots, n\}\} \cup \\ & \{a_{ij}, \, z_{ij} \,:\, i, j \in \{1, \dots, n\}\} \cup \\ & \{c_i \,:\, i \in \{0, \dots, 3n+1\}\} \cup \\ & \{k, yes, no\}. \end{aligned}$$

- **Input alphabet:** Σ, as described at the beginning of the section. Let us remark that $\Sigma \subset \Gamma$.
- **Set of labels:** $H = \{\langle i, j \rangle \,:\, i, j \in \{1, \dots, n\}\} \cup \{1, \dots, n\} \cup \{a, skin\}$.
- **Electrical charges:** $EC = \{0, +\}$.
- **Membrane structure:** $[\,[\,]_1^0 \dots [\,]_n^0 \, [\,]_{\langle 1,1 \rangle}^0 \dots [\,]_{\langle n,n \rangle}^0 \, [\,]_a^0 \,]_{skin}^0$.
- **Initial multisets:** $w^a = c_0$, $w^{skin} = w^{ij} = w^k = \lambda$ for $i, j, k \in \{1, \dots, n\}$.
- **Input label:** $i = skin$.

The set of rules $\mathcal{R}$:

R1. $a_{ij} \, [\,]_{\langle i,j \rangle}^0 \to [a_{ij}]_{\langle i,j \rangle}^+$ for $i, j \in \{1, \dots, n\}$.
Each input object a_{ij} activates the corresponding membrane by changing its polarization. Notice that such a symbol a_{ij} represents an edge in the input graph.
R2. $y_j \, [\,]_j^0 \to [y_j]_j^+$ for $j \in \{1, \dots, n\}$.
The object y_j activates the membrane j by changing its polarization. As the input multiset always has exactly one object of the form y_j, Π_n will have a unique membrane with label in $\{1, \dots, n\}$ and polarization $+$.
R3. $[x_i \to z_{i1} \dots z_{in} t_i]_{skin}^0$ for $i \in \{1, \dots, n\}$.
The goal of these rules is to create $n+1$ copies of an object x_i. A copy z_{ij} will be able to produce an object x_j if the edge (i, j) belongs to E. The object t_i will be used to witness that vertex i is reachable.
R4. $\left.\begin{array}{l} z_{ij} \, [\,]_{\langle i,j \rangle}^+ \to [x_j]_{\langle i,j \rangle}^0 \\ t_j \, [\,]_j^+ \to [k]_j^0 \end{array}\right\}$ for $i, j \in \{1, \dots, n\}$.
If the membrane with label $\langle i, j \rangle$ has polarization $+$, then the symbol z_{ij} produces a symbol x_j inside this membrane. Meanwhile, the polarization of this membrane changes from $+$ to 0, i.e., the membrane is deactivated. Moreover, if the symbol t_j appears in the skin and the membrane with label j has positive polarization, then an object k is produced inside this membrane. Such object k will start the process to send *yes* out to the environment.
R5. $[k]_j^0 \to k \, [\,]_j^0 \qquad k \, [\,]_a^0 \to [k]_a^+$.
The object k is a witness of the success of the STCON problem. If it is produced, it goes into the membrane with label a and changes its polarization to $+$.
R6. $[x_j]_{\langle i,j \rangle}^0 \to x_j \, [\,]_{\langle i,j \rangle}^0$ for $i, j \in \{1, \dots, n\}$.

The produced object x_j is sent to the membrane *skin* in order to go on the computation by rules form **R3**.

R7. $\left.\begin{array}{l} [c_i \to c_{i+1}]_a^0 \ [c_{3n+1}]_a^0 \to no\,[\,]_a^0 \\ [c_i \to c_{i+1}]_a^+ \ [c_{3n+1}]_a^+ \to yes\,[\,]_a^0 \end{array}\right\}$ for $i \in \{0, \ldots, 3n\}$.

Object c_i evolves to c_{i+1} regardless of the polarization of the membrane a. If during the evolution the object k has gone inside such membrane, then the polarization changes to + and the object c_{3n+1} will produce *yes* in the membrane *skin*. Otherwise, if the object k is not produced, the polarization is not changed and the object c_{3n+1} will produce *no*.

R8. $[no]_{skin} \to no\,[\,]_{skin}$ $\quad [yes]_{skin} \to yes\,[\,]_{skin}$.

Finally, *yes* or *no* is sent out the P system in the last step of computation.

To see in more details how the computation of the presented P system goes, let us consider an instance $I = (s, t, G)$ of STCON where G is a graph $\langle\{1, \ldots, n\}, E\rangle$. The computation of Π_n on $cod(I)$ can be described as follows. During the first step, using rules in **R1**, every a_{ij} enters to the membrane with label $\langle i, j\rangle$ and changes its polarization to +. Thus, after the first step the edges in E are encoded by the positive polarizations of the membranes with labels of the form $\langle i, j\rangle$. During the same step, using the corresponding rule in **R2**, y_t enters to the membrane with label t and changes its polarization to +. This membrane will be used to recognize if an object representing that t is reachable from s is introduced by the system.

Now let $l \in \{1, 4, \ldots, 3(n-1)+1\}$ and consider an object x_i in the skin membrane. During the lth step, using rules in **R3**, x_i creates $n+1$ copies of itself. The system will try to use a copy z_{ij} ($j \in \{1, \ldots, n\}$) in the next step to create a new object x_j. The copy t_i will be used to decide if $i = t$.

During the $(l+1)$th step, using rules in **R4**, the systems sends z_{ij} into the membrane with label $\langle i, j\rangle$ if that membrane has a positive polarization. Meanwhile, z_{ij} evolves to x_j and the polarization of the membrane changes to neutral. During the same step, if $i = t$ and the membrane with label t has positive polarization, then the system sends t_i to this membrane. Meanwhile, t_i evolves to k and the polarization of membrane t changes to neutral.

During the $(l+2)$th step, using rules in **R6**, the object x_j is sent out from the membrane with label $\langle i, j\rangle$. Moreover, if the membrane with label t contains k, then this k is sent out from membrane t.

One can see that during the above three steps the system introduces an object x_j if and only if (i, j) is an edge in E. Using this observation we can derive that during the computation of the system, an object x_j appears in the skin if and only if there is a path in G from s to j. Thus, t is reachable from s in G if and only if there is a configuration of Π_n where the skin contains x_t. However, in this case an object k is introduced in the membrane with label t. It can also be seen that Π_n sends out to the environment *yes* if and only if k appears in membrane t. Moreover, if k does not appear in membrane t, then the systems sends out to the environment *no*. Thus, Π_n sends out to the environment *yes* or *no* according to that t is reachable from s or not. As Π_n stops in at most $3n+2$ steps, we can conclude that the family $\mathbf{\Pi}$ decides STCON in linear time in the number of vertices of the input graph.

4.2 P Systems with Active Membranes, with Dissolution and without Polarization

Based on the solution presented in the previous sub-section, we give here a uniform family $\mathbf{\Pi} = \{\Pi_n\}_{n\in\mathbb{N}}$ in $\mathbf{PMC}_{\mathcal{AM}^0}$ which solves STCON. As here we cannot use the polarizations of the membranes, we use membrane dissolution to select those membranes of the initial configuration that correspond to the edges of the input graph. Next we will describe the mentioned family $\mathbf{\Pi}$. Since we do not use polarizations, we do not indicate it at the upper-right corner of the membranes. The family presented here is

$$\Pi_n = \langle \Gamma, \Sigma, H, EC, \mu, W, \mathcal{R}, i\rangle.$$

- **Alphabet:**

$$\begin{aligned}\Gamma = \{&x_i, v_{1i}, v_{2i}, v_{3i}, v_i, y_i, t_i \,:\, i \in \{1,\dots,n\}\}\cup\\ \{&a_{ij},\, z_{ij} \,:\, i,j \in \{1,\dots,n\}\}\cup\\ \{&c_i \,:\, i \in \{0,\dots,3n+4\}\}\cup\\ \{&k, yes, no\}.\end{aligned}$$

- **Input alphabet:** Σ, as described at the beginning of the section.
- **Set of labels:** $H = \{\langle i,j,in\rangle, \langle i,j,out\rangle \,:\, i,j \in \{1,\dots,n\}\} \cup \{\langle i,in\rangle, \langle i,out\rangle \,:\, i \in \{1,\dots,n\} \cup \{a, skin\}$.
- **Electrical charges:** $EC = \varnothing$.
- **Membrane structure:** $[\,[[\,]_{\langle 1,in\rangle}]_{\langle 1,out\rangle}\cdots[[\,]_{\langle n,in\rangle}]_{\langle n,out\rangle}\,[[\,]_{\langle 1,1,in\rangle}]_{\langle 1,1,out\rangle}\cdots[[\,]_{\langle n,n,in\rangle}]_{\langle n,n,out\rangle}\,[\,]_a\,]_{skin}$.
- **Initial multisets:** $W = \{w^a, w^{\langle 1,in\rangle},\dots,w^{\langle n,in\rangle}, w^{\langle 1,out\rangle},\dots,w^{\langle n,out\rangle}, w^{\langle 1,1,in\rangle},\dots,w^{\langle n,n,in\rangle}, w^{\langle 1,1,out\rangle},\dots,w^{\langle n,n,out\rangle}, w^{skin}\}$, where $w^a = c_0$, $w^{skin} = w^{\langle i,j,out\rangle} = w^{\langle k,out\rangle} = \lambda$, $w^{\langle i,j,in\rangle} = w^{\langle k,in\rangle} = f_0$, for $i,j,k \in \{1,\dots,n\}$.
- **Input label:** $i = skin$.

The set of rules $\mathcal{R}$:

R0. $[x_i \to v_{1i}]_{skin},\ [v_{ji} \to v_{j+1,i}]_{skin},\ [v_{3i} \to v_i]_{skin}$ for $i \in \{1,\dots,n\}$ and $j \in \{1,2\}$.
In this solution we cannot use the objects x_i in the same role as we did in the previous sub-section because of the following reason. The system needs four steps to select those membranes in the initial membrane configuration that correspond to the edges in E. Thus, the system introduces in four steps the objects v_i which will act in this solution as the objects x_i did in the previous one.

R1. $$\left.\begin{array}{l}[f_m \to f_{m+1}]_{\langle i,j,in\rangle}\\ [f_3]_{\langle i,j,in\rangle} \to f_4\\ [f_4]_{\langle i,j,out\rangle} \to f_4\end{array}\right\}\text{ for } i,j \in \{1,\dots,n\},\ m \in \{0,1,2\}.$$

These rules dissolve the membranes with label $\langle i,j,in\rangle$ and $\langle i,j,out\rangle$ if the input symbol a_{ij} is not present in the system. On the other hand, if a_{ij} is in the system, then it prevents the dissolution of the membrane with label $\langle i,j,out\rangle$ using the following rules.

R2. $\left.\begin{array}{l} a_{ij}\,[\,]_{\langle i,j,m\rangle} \to [a_{ij}]_{\langle i,j,m\rangle} \\ [a_{ij}]_{\langle i,j,in\rangle} \to a_{ij} \end{array}\right\}$ for $i,j \in \{1,\dots,n\}$, $m \in \{in, out\}$.

By these rules the input symbol a_{ij} goes into the membrane with label $\langle i,j,in\rangle$ and dissolves that. This way the second rule in **R1** cannot be applied, thus the membrane with label $\langle i,j,out\rangle$ cannot be dissolved by the third rule.

R3. $\left.\begin{array}{l} [f_m \to f_{m+1}]_{\langle j,in\rangle} \\ [f_3]_{\langle j,in\rangle} \to f_4 \\ [f_4]_{\langle j,out\rangle} \to f_4 \end{array}\right\}$ for $j \in \{1,\dots,n\}$, $m \in \{0,1,2\}$.

These rules dissolve the membranes with label $\langle j,in\rangle$ and $\langle j,out\rangle$ if the input symbol y_j is not present in the system. However, if y_j is in the system, then it prevents the dissolution of the membrane with label $\langle j,out\rangle$ using the following rules.

R4. $\left.\begin{array}{l} y_j\,[\,]_{\langle j,m\rangle} \to [y_j]_{\langle j,m\rangle} \\ [y_j]_{\langle j,in\rangle} \to y_j \end{array}\right\}$ for $j \in \{1,\dots,n\}$ and $m \in \{in, out\}$.

By these rules the input symbol y_j goes into the membrane with label $\langle j,in\rangle$ and dissolves that. With this it is achieved that the membrane with label $\langle j,out\rangle$ is not dissolved by the rules in **R3**.

R5. $[v_i \to z_{i1}\dots z_{in}t_i]_{skin}$ for $i \in \{1,\dots,n\}$.

The role of these rules is the same as that of the rules in **R3** in Section 4.1.

R6. $\left.\begin{array}{l} z_{ij}\,[\,]_{\langle i,j,out\rangle} \to [v_j]_{\langle i,j,out\rangle} \\ t_j\,[\,]_{\langle j,out\rangle} \to [k]_{\langle j,out\rangle} \end{array}\right\}$ for $i,j \in \{1,\dots,n\}$.

The role of these rules is similar to that of the rules in **R4** in Section 4.1: If the membrane with label $\langle i,j,out\rangle$ has not been dissolved, then the object z_{ij} produces a symbol v_j inside this membrane. Analogously, if the symbol t_j appears in the skin and the membrane with label $\langle j,out\rangle$ is not dissolved, then an object k is produced inside this membrane. Such object k will start the process to send *yes* out to the environment.

R7. $[k]_{\langle j,out\rangle} \to k\,[\,]_{\langle j,out\rangle}$ $\quad k\,[\,]_a \to [k]_a$ $\quad [k]_a \to k$.

The object k is a witness of the success of the STCON problem. If it is produced, it goes into the membrane with label a and dissolves it.

R8. $[v_j]_{\langle i,j\rangle} \to v_j$ for $i,j \in \{1,\dots,n\}$.

The produced object v_j dissolves the membrane with label $\langle i,j\rangle$ as the computation does not need any more this membranes. This way the object v_j gets to the skin and the computation can go on using the rules in **R5**.

R9. $\left.\begin{array}{ll} [c_i \to c_{i+1}]_a & [c_{3n+4}]_a \to no\,[\,]_a \\ & [c_{i+1}]_{skin} \to [yes]_{skin} \end{array}\right\}$ for $i \in \{0,\dots,3n+3\}$.

Object c_i evolves to c_{i+1} in membrane with label a. If during the evolution the object k has gone inside this membrane, then it dissolves it and the object c_{i+1} gets to the membrane s where it produces *yes*. Otherwise, if the object k is not produced, c_{3n+4} remains in membrane with label a and produces *no*.

R10. $[no]_{skin} \to no\,[\,]_{skin}$ $\quad [yes]_{skin} \to yes\,[\,]_{skin}$.
Finally, *yes* or *no* is sent out the P system in the last step of computation.

One can observe that during the first four steps of Π_n a membrane with label $\langle i, j, out\rangle$ is not dissolved if and only if a_{ij} is in the input. Thus, Π_n has a membrane with label $\langle i, j, out\rangle$ after the first four steps if and only if Π_n defined in Section 4.1 has a membrane $\langle i, j\rangle$ with positive polarization after the first step. Similar observations apply in the case of membranes with label $\langle j, out\rangle$. Thus, the correctness of Π_n defined in this section follows from the correctness of Π_n defined in Section 4.1. One can also observe that Π_n stops after at most $3n+5$ steps, which means that the family $\mathbf{\Pi}$ defined in this section decides STCON in linear time.

4.3 P Systems with Membrane Creation

Here we provide the design of a uniform family of P systems in the framework of *P systems with Membrane Creation* which solves the problem STCON. Since STCON is **NL**-complete, we have a direct proof of $\mathbf{NL} \subseteq \mathbf{PMC}_{\mathcal{MC}}$. This result is well-know, since $\mathbf{NL} \subset \mathbf{NP}$ and $\mathbf{NP} \subseteq \mathbf{PMC}_{\mathcal{MC}}$ (see [4]). Nonetheless, to the best of our knowledge, this is the first design of a P system family which solves STCON in $\mathbf{PMC}_{\mathcal{MC}}$.

Next we will describe the family $\mathbf{\Pi} = \{\Pi_n\}_{n\in\mathbb{N}}$ of P systems in $\mathbf{PMC}_{\mathcal{MC}}$. Each Π_n will receive as input an instance of the STCON as described at the beginning of the section and will release *yes* or *no* into the environment in the last step of the computation as the answer of the decision problem.

The family presented here is

$$\Pi_n = \langle \Gamma, \Sigma, H, \mu, w^a, w^b, w^c, \mathcal{R}, i\rangle.$$

- **Alphabet:**

$$\begin{aligned}\Gamma = \{&x_i, y_i, t_i \,:\, i \in \{1, \dots, n\}\} \cup \\ \{&a_{ij},\, z_{ij} \,:\, i, j \in \{1, \dots, n\}\}\cup \\ \{&no_i \,:\, i \in \{0, \dots, 3n+3\}\}\cup \\ \{&yes_i \,:\, i \in \{1, \dots, 4\}\}\cup \\ \{&yes, no\}.\end{aligned}$$

- **Input alphabet:** Σ, as it is described at beginning of the section.
- **Set of labels:** $H = \{\langle i, j\rangle \,:\, i, j \in \{1, \dots, n\}\} \cup \{1, \dots, n\} \cup \{a, b, c\}$.
- **Membrane structure:** $[\,[\,]_a\,[\,]_b\,]_c$.
- **Initial multisets:** $w^a = no_0$, $w^b = w^c = \lambda$.
- **Input label:** $i = b$.

The set of rules $\mathcal{R}$:

R1. $[[a_{ij} \to [\,\lambda\,]_{\langle i,j\rangle}\,]_b$ for $i, j \in \{1, \dots, n\}$.
Each input symbol a_{ij} creates a new membrane with label $\langle i, j\rangle$. Recall that such a symbol a_{ij} represents an edge in the directed graph.

R2. $[y_j \to [\,\lambda\,]_j\,]_b$ for $j \in \{1, \ldots, n\}$.
By these rules an input symbol y_j creates a new membrane with label j.
R3. $[x_i \to z_{i1} \ldots z_{in} t_i]_b$ for $i \in \{1, \ldots, n\}$.
The role of these rules is the same as those of the rules in **R3** in Section 4.1.
R4. $\left.\begin{array}{l} z_{ij}\,[\,]_{\langle i,j\rangle} \to [x_j]_{\langle i,j\rangle} \\ t_j\,[\,]_j \to [yes_0]_j \end{array}\right\}$ for $i, j \in \{1, \ldots, n\}$.
The role of these rules is similar to that of the rules in **R4** in Section 4.1 except that here an object t_j introduces an object yes_0 in the membrane with label j. This new object yes_0 will evolve with the rules in **R6** and **R7** until the final object yes is produced in the environment.
R5. $[x_j]_{\langle i,j\rangle} \to x_j$ for $i, j \in \{1, \ldots, n\}$.
The object x_j dissolves the membrane with label $\langle i, j\rangle$. The useful information is that x_j is reachable. We keep this information, but the membrane can be dissolved. This way x_j gets to the membrane b and the computation can go on using the rules in $\mathbf{R_3}$.
R6. $[yes_0]_j \to yes_1$ for $j \in \{1, \ldots, n\}$.
For each possible value of j, if yes_0 is produced, the corresponding membrane is dissolved and yes_1 appears in the membrane with label b.
R7. $[yes_1]_b \to yes_2,\ yes_2\,[\,]_a \to [yes_3]_a,$
$[yes_3]_a \to yes_4,\ [yes_4]_c \to yes\,[\,]_c\,.$
The evolution of the objects yes_i firstly produces the dissolution of the membrane b. If this membrane is dissolved, the rules from **R3** will be no longer applied. In a similar way, object yes_3 also dissolves membrane a and this stops the evolution of the objects inside such membrane.
R8. $[no_i \to no_{i+1}]_a$ for $i \in \{1, \ldots, 3n+2\}$.
The object no_i evolves inside the membrane a. If this evolution is not halted by the dissolution of the membrane a, these objects will produce the object no in the environment.
R9. $[no_{3n+3}]_a \to no \qquad [no]_c \to no\,[\,]_c\,.$
If the evolution of no_i is not stopped, the object no_{3n+3} dissolves the membrane a and creates a new object no. This object will be sent to the environment in the next step of the computation.

It is not difficult to see using the comments given after the rules that this solution works essentially in the same way as our first solution. The main difference is that while in Section 4.1 an input symbol a_{ij} is used to change the polarization of a membrane $\langle i, j\rangle$, here this symbol is used to create such a membrane. Thus, the correctness of the solution presented here can be seen using the correctness of the solution given in Section 4.1. It is also clear that the P systems presented here work in linear time in the number of vertices in the input graph.

As we have mentioned, in solutions of problems in **P** via uniform families of P systems it is important to use such input encoding and P system constructing devices that are not capable to compute the correct answer. It is easy to see that the decision processes in the solutions of STCON presented in this paper are entirely done by the P systems themselves. Thus our solutions could be easily modified so that the construction of the used families and the computation of the input encoding can be carried out by reasonable weak computational devices, for example, by logarithmic-space deterministic Turing machines.

5 Conclusions

The design of a uniform family of recognizer P systems working in polynomial time which solves a decision problem with *pure* Membrane Computing techniques is a hard task, regardless the complexity class of the problem. The difficulty comes from the hard restrictions imposed to such family. Firstly, the use of *input* P systems implies that each instance of the problem must be encoded as a multiset and such multiset must be introduced at the starting configuration in *one* input membrane. The multiset encoding the instance cannot be distributed in several membranes in the starting configuration. Secondly, in *uniform* families, each P system must solve *all* the instances of the problem of the same *size* (regardless of whether the answer is positive or not). This means that the set of rules which leads to send *yes* to the environment and the set of rules which leads to send *no* must be present in the design of the P system; and thirdly, the standard definition of recognizer P systems claims that an object *yes* or *no* (but no both) is sent to the environment in the *last* step of computation.

A deep study of these constraints shows that it is not sufficient to implement a design of P system with the control scheme "*if* the restrictions of the decision problem are satisfied, *then* an object *yes* must be sent to the environment". Instead of such scheme, the design must consider the following structure: "*if* the restrictions are satisfied, *then* an object *yes* must be sent to the environment, *else* an object *no* must be sent". This scheme *if-then-else* must be controlled with the ingredients of the P system model. In the three presented designs, this *if-then-else* scheme is implemented via dissolution, polarization, or membrane creation.

These ideas lead us to consider the necessity of revisiting the complexity classes under **P** and adapt the definition of recognizer P systems for these classes. Some papers in this new research line can be found in the literature (see, e.g., [12]), but further research is needed.

Acknowledgements

This work was partially done during Zsolt Gazdag's visit at the Research Institute of Mathematics of the University of Sevilla (IMUS) partially supported by IMUS. Miguel A. Gutiérrez–Naranjo acknowledges the support of the project TIN2012-37434 of the Ministerio de Economía y Competitividad of Spain.

References

1. Csuhaj-Varjú, E., Gheorghe, M., Rozenberg, G., Salomaa, A., Vaszil, G. (eds.): Membrane Computing - 13th International Conference, CMC 2012, Budapest, Hungary, August 28-31, 2012, Revised Selected Papers, Lecture Notes in Computer Science, vol. 7762. Springer (2013)
2. Díaz-Pernil, D., Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Riscos-Núñez, A.: A Linear Time Solution to the Partition Problem in a Cellular Tissue-Like Model. Journal of Computational and Theoretical Nanoscience 7(5), 884–889 (MAY 2010)
3. Gazdag, Z., Kolonits, G.: A new approach for solving SAT by P systems with active membranes. In: Csuhaj-Varjú et al. [1], pp. 195–207
4. Gutiérrez-Naranjo, M.A., Pérez-Jiménez, M.J., Romero-Campero, F.J.: A uniform solution to SAT using membrane creation. Theoretical Computer Science 371(1-2), 54–61 (2007)
5. Leporati, A., Zandron, C., Ferretti, C., Mauri, G.: Solving numerical NP-complete problems with spiking neural P systems. In: Eleftherakis, G., Kefalas, P., Păun, Gh., Rozenberg, G., Salomaa, A. (eds.) Workshop on Membrane Computing. Lecture Notes in Computer Science, vol. 4860, pp. 336–352. Springer, Berlin Heidelberg (2007)
6. Murphy, N., Woods, D.: A characterisation of NL using membrane systems without charges and dissolution. In: Calude, C.S., Costa, J.F., Freund, R., Oswald, M., Rozenberg, G. (eds.) Unconventional Computing, Lecture Notes in Computer Science, vol. 5204, pp. 164–176. Springer Berlin Heidelberg (2008)
7. Murphy, N., Woods, D.: On acceptance conditions for membrane systems: characterisations of L and NL. In: Proceedings International Workshop on The Complexity of Simple Programs. Cork, Ireland, 6-7th December 2008. pp. 172–184. Electronic Proceedings in Theoretical Computer Science. Vol 1 (2009)
8. Pérez-Jiménez, M.J., Riscos-Núñez, A., Romero-Jiménez, A., Woods, D.: Complexity - membrane division, membrane creation. In: Păun et al. [13], pp. 302 – 336
9. Pérez-Jiménez, M.J., Romero-Jiménez, A., Sancho-Caparrini, F.: A polynomial complexity class in P systems using membrane division. In: Csuhaj-Varjú, E., Kintala, C., Wotschke, D., Vaszyl, G. (eds.) Proceeding of the 5th Workshop on Descriptional Complexity of Formal Systems. DCFS 2003. pp. 284–294 (2003)
10. Pérez-Jiménez, M.J., Romero-Jiménez, Á., Sancho-Caparrini, F.: A polynomial complexity class in P systems using membrane division. Journal of Automata, Languages and Combinatorics 11(4), 423–434 (2006)
11. Porreca, A.E.: Computational Complexity Classes for Membrane System. Master's thesis, Univertita´ di Milano-Bicocca, Italy (2008)
12. Porreca, A.E., Leporati, A., Mauri, G., Zandron, C.: Sublinear-space P systems with active membranes. In: Csuhaj-Varjú et al. [1], pp. 342–357
13. Păun, Gh., Rozenberg, G., Salomaa, A. (eds.): The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford, England (2010)