Causality in Membrane Systems

Nadia Busi

Dipartimento di Scienze dell'Informazione, Università di Bologna,
Mura A. Zamboni 7, I-40127 Bologna, Italy
busi@cs.unibo.it

Summary. P systems are a biologically inspired model introduced by Gheorghe Păun with the aim of representing the structure and the functioning of the cell. P systems are usually equipped with the maximal parallelism semantics; however, since their introduction, some alternative semantics have been proposed and investigated.

We propose a semantics that describes the causal dependencies occurring between the reactions of a P system. We investigate the basic properties that are satisfied by such a semantics. The notion of causality turns out to be quite relevant for biological systems, as it permits to point out which events occurring in a biological pathway are necessary for another event to happen.

1 Introduction

Membrane computing is a branch of natural computing, initiated by Gheorghe Păun with the definition of P systems in [22, 23, 24]. The aim is to provide a formal modeling of the structure and the functioning of the cell, making use especially of automata, languages and complexity theoretic tools.

Membrane systems are based upon the notion of *membrane structure,* which is a structure composed by several cell-membranes, hierarchically embedded in a main membrane called the *skin membrane.* A plane representation of a membrane structure can be given by means of a Venn diagram, without intersected sets and with a unique superset. The membranes delimit *regions* and we associate with each region a set of *objects*, described by some symbols over an alphabet, and a set of *evolution rules.*

In the basic variant, the objects evolve according to the evolution rules, which can modify the objects to obtain new objects and send them outside the membrane or to an inner membrane. The evolution rules are applied in a maximally parallel manner: at each step, all the objects which can evolve should evolve.

A computation device is obtained: we start from an initial configuration, with a certain number of objects in certain membranes, and we let the system evolve. If a computation *halts,* that is no further evolution rule can be applied, the result of the computation is defined to be the number of objects in a specified membrane (or expelled through the skin membrane). If a computation never halts (i.e. one or more object can be rewritten forever), then it provides no output.

Since their introduction, plenty of variants of P systems have been introduced, and a lot of research effort has been carried out, expecially concerned with the study of the expressivity and the universality of the proposed models and with the ability to solve NP-complete problems in polynomial time.

The aim of this work is to start an investigation of the causal dependencies arising in among reactions occurring in P systems. The main motivation for this work comes from system biology, as the understanding of the causal relations occurring between the events of a complex biological pathway could be of precious help, e.g., for limiting the search space in the case some unpredicted event occurs.

In this paper we concentrate on P systems with cooperative rules, namely systems whose evolution rules are of the form $u \rightarrow v$, representing the fact that the objects in u are consumed and the objects in v are produced.

The study of causal semantics in concurrency theory is quite old. For example, the study of a causal semantics for process algebras dates back to the early nineties for CCS [20] (see, e.g., [13, 11, 18]), and to the mid nineties for the π-calculus [21] (see, e.g., [3, 5, 14, 15]).

To the best of our knowledge, the only other works that deal with causality in bio-inspired calculi with membranes and compartments are the following. In [7] a causal semantics for the Mate/Bud/Drip Brane Calculus [9] is proposed. In [17] a causal semantics for Beta Binders [26, 27] – based on the π-calculus semantics and on the enhanced operational semantics approach of [15] – is defined. One of the main differences between Beta Binders on one side, and Brane Calculi and P systems on the other side, is that the membrane structure in Beta Binders is flat, whereas in Brane Calculi and in P systems the membranes are nested to form a hierarchical structure.

The paper is organized as follows. After providing some basic definitions in Section 2, in Section 3 we define (cooperative) P systems. Section 4 recalls a detailed definition of maximal parallelism semantics that will be used in the following to provide a comparison between the causal and the maximal parallelism semantics. Section 5 is devoted to the definition of the causal semantics; after an informal introduction, a formal definition is provided, and finally some result on the properties enjoyed by the causal semantics are given. Section 6 reports some conclusive remarks.

2 Basic Definitions

In this section we provide some definitions that will be used throughout the paper.

Definition 1. *Given a set S, a* finite multiset *over S is a function $m : S \rightarrow I\!N$ such that the set $dom(m) = \{s \in S \,|\, m(s) \neq 0\}$ is finite. The* multiplicity *of an element s in m is given by the natural number $m(s)$. The set of all finite* *multisets over S, denoted by $\mathcal{M}_{fin}(S)$, is ranged over by m. A multiset m such that $dom(m) = \emptyset$ is called* empty*. The empty multiset is denoted by $\emptyset$.*

Given the multisets m and m', we write $m \subseteq m'$ if $m(s) \leq m'(s)$ for all $s \in S$ while $\oplus$ denotes their multiset union*, i.e., $m \oplus m'(s) = m(s) + m'(s)$. The operator $\setminus$ denotes* multiset difference*: $(m \setminus m')(s) =$ if $m(s) \geq m'(s)$ then $m(s) - m'(s)$ else 0. The* scalar product*, $j \cdot m$, of a number j with m is $(j \cdot m)(s) = j \cdot (m(s))$. The cardinality of a multiset is the number of occurrences of elements contained in the multiset: $|m| = \sum_{s \in S} m(s)$.*

The powerset of a set S is defined as $\mathcal{P}(S) = \{X \mid X \subseteq S\}$.

Definition 2. *Let m be a finite multiset over S and $X \subseteq S$. The multiset $m|_X$ is defined as follows: for all $s \in S$, $m|_X(s) = m(s)$ if $s \in X$, and $m|_X(s) = 0$ otherwise.*

Definition 3. *A string over S is a finite (possibly empty) sequence of elements in S. Given a string $u = x_1 \ldots x_n$, the length of u is the number of occurrences of elements contained in u and is defined by $|u| = n$.*

With S^* we denote the set of strings over S, and $u, v, w, \ldots$ range over S. Given $n \geq 0$, with S^n we denote the set of strings of length n over S.

Given a string $u = x_1 \ldots x_n$ and i such that $1 \leq i \leq n$, with $(u)_i$ we denote the i-th element of u, namely, $(u)_i = x_i$.

Given a string $u = x_1 \ldots x_n$, the multiset corresponding to u is defined as follows: for all $s \in S$, $m_u(s) = |\{i \mid x_i = s \wedge 1 \leq i \leq n\}|$. With abuse of notation, we use u to denote also m_u.

Definition 4. *With $S \times T$ we denote the Cartesian product of sets S and T, with $\times_n S$, $n \geq 1$, we denote the Cartesian product of n copies of set S and with $\times_{i=1}^n S_i$ we denote the Cartesian product of sets $S_1, \ldots, S_n$, i.e., $S_1 \times \ldots \times S_n$. The ith projection of $(x_1, \ldots, x_n) \in \times_{i=1}^n S_i$ is defined as $\pi_i(x) = x_i$, and lifted to subsets $X \subseteq \times_{i=1}^n S_i$ as follows: $\pi_i(X) = \{\pi_i(x) \mid x \in X\}$.*

Given a binary relation R over a set S, with R^n we denote the composition of n instances or R, with R^+ we denote the transitive closure of R, and with R^* we denote the reflexive and transitive closure of R.

3 P Systems

We recall the definition of catalytic P systems without priorities on rules. For a thorough description of the model, motivation, and examples see, e.g., [8, 12, 22, 23, 24]. To this aim, we start with the definition of *membrane structure*:

Definition 5. *Given the alphabet $\{[,]\}$, the set MS is the least set inductively defined by the following rules:*

- $[\,] \in MS$

- *if $\mu_1, \mu_2, \ldots, \mu_n \in MS$, $n \geq 1$, then $[\mu_1 \ldots \mu_n] \in MS$*

We define the following relation over MS: $x \sim y$ iff the two strings can be written as $x = [_1 \ldots [_2 \ldots]_2 \ldots [_3 \ldots]_3 \ldots]_1$ and $y = [_1 \ldots [_3 \ldots]_3 \ldots [_2 \ldots]_2 \ldots]_1$ (i.e., if two pairs of parentheses that are neighbors can be swapped together with their contents).

The set $\overline{MS}$ of membrane structures *is defined as the set of equivalence classes w.r.t. the relation $\sim^*$.*

We call a *membrane* each matching pair of parentheses appearing in the membrane structure. A membrane structure μ can be represented as a Venn diagram, in which any closed space (delimited by a membrane and by the membanes immediately inside) is called a *region* of μ.

Definition 6. *A* P system *(of degree d, with $d \geq 1$) is a construct*

$$\Pi = (V, \mu, w_1^0, \ldots, w_d^0, R_1, \ldots, R_d, i_0), \text{ where:}$$

1. *V is a finite alphabet whose elements are called* objects*;*
2. *μ is a* membrane structure *consisting of d membranes (usually labelled with i and represented by corresponding brackets $[_i$ and $]_i$, with $1 \leq i \leq d$);*
3. *w_i^0, $1 \leq i \leq d$, are strings over V associated with the regions $1, 2, \ldots, d$ of μ; they represent multisets of objects present in the regions of μ at the beginning of computation (the multiplicity of a symbol in a region is given by the number of occurrences of this symbol in the string corresponding to that region);*
4. *R_i, $1 \leq i \leq d$, are finite sets of* evolution rules *over V associated with the regions $1, 2, \ldots, d$ of μ; these evolution rules are of the form $u \to v$, where u and v are strings from $(V \times \{here, out, in\})^*$;*
5. *$i_0 \in \{1, \ldots, d\}$ specifies the* output *membrane of Π.*

The membrane structure and the multisets represented by w_i^0, $1 \leq i \leq d$, in Π constitute the *initial state*[1] of the system. A transition between states is governed by an application of the evolution rules which is done in parallel; all objects, from all membranes, which *can be* the subject of local evolution rules *have to* evolve simultaneously.

The application of a rule $u \to v$ in a region containing a multiset m results in subtracting from m the multiset identified by u, and then in adding the multiset defined by v. The objects can eventually be transported through membranes due to the targets *in* and *out* (we usually omit the target *here*).

The system continues parallel steps until there remain no applicable rules in any region of Π; then the system halts. We consider the number of objects from V contained in the output membrane i_0 when the system halts as the *result* of the underlying computation of Π.

[1] Here we use the term *state* instead of the classical term *configuration* because we will define a (essentially equivalent but syntactically) different notion of configuration in Section 5.

We introduce a couple of functions on membrane structures that will be useful in the following:

Definition 7. *Let μ be a membrane structure consisting of d membranes, labelled with $\{1, \ldots, d\}$.*

Given two membranes i and j in μ, we say that i is contained in j if the surface delimited by the perimeter of i in the Venn diagram representation of μ is contained inside the perimeter of j.

We say that i is the father *of j (and j is a* child *of i) if the membrane j is contained in i, and no membrane exists that contains j and is contained in i.*

The partial function $father : \{1, \ldots, d\} \to \{1, \ldots, d\}$ returns the father of a membrane i, or is undefined if i is the external membrane.

The function $children : \{1, \ldots, d\} \to \mathcal{P}(\{1, \ldots, d\})$ returns the set of children of a membrane.

For example, take $\mu = [_1[_2[_3\ ]_3]_2\ [_4\ ]_4]_1$; then, $father(2) = father(4) = 1$, $father(3) = 2$ and $father(1)$ is undefined; moreover, $children(4) = \emptyset$ and $children(1) = \{2, 4\}$.

4 Maximal Parallelism Semantics for P Systems

In order to compare the classical maximal parallelism semantics with the causal semantics, in this section we recall a detailed definition of the computation of a P system, proposed in [4], where a maximal parallelism evolution step is represented as a (maximal) sequence of simple evolution steps, which are obtained by the application of a single evolution rule.

Throughout this section, we let $\Pi = (V, \mu, w_1^0, \ldots, w_d^0, R_1, \ldots, R_d, i_0)$ be a P system.

To represent the states of the system reached after the execution of a non maximal sequence of simple evolution rules, we introduce the notion of *partial configuration* of a system. In a partial configuration, the contents of each region is represented by two multisets:

- The multiset of *active objects* contains the objects that were in the region at the beginning of the current maximal parallelism evolution step. These objects can be used by the next simple evolution step.
- The multiset of *frozen objects* contains the objects that have been produced in the region during the current maximal parallelism evolution step. These objects will be available for consumption in the next maximal parallelism evolution step.

Definition 8. *A* partial configuration *of Π is a tuple $((w_1, \bar{w}_1), \ldots, (w_d, \bar{w}_d)) \in \times_d(V^* \times V^*)$.*

We use $\times_{i=1}^{d}(w_i, \bar{w}_i)$ to denote the partial configuration above.

The set of partial configurations of Π is denoted by $Conf_\Pi$. We use $\gamma, \gamma', \gamma_1, \ldots$ to range over $Conf_\Pi$.

In the above definition, $w_1, \ldots, w_d$ represent the active multisets, whereas $\bar{w}_1, \ldots, \bar{w}_d$ represent the frozen multisets.

A *configuration* is a partial configuration containing no frozen objects; configurations represent the states reached after the execution of a maximal parallelism computation step.

Definition 9. *A* configuration *of Π is a partial configuration $\times_{i=1}^{d}(w_i, \bar{w}_i)$ satisfying the following: $\bar{w}_i = \emptyset$ for $i = 1, \ldots, d$.*

The initial configuration *of Π is the configuration $\times_{i=1}^{d}(w_i^0, \emptyset)$.*

The size of a partial configuration is the number of active objects contained in the configuration.

Definition 10. *Let $\gamma = \times_{i=1}^{d}(w_i, \bar{w}_i)$ be a partial configuration. The* size *of γ is $\#(\gamma) = \sum_{i=1}^{d} |w_i|$.*

The execution of a simple evolution rule is formalized by the notion of reaction relation, defined as follows:

Definition 11. *The* reaction relation $\mapsto$ *over $Conf_\Pi \times Conf_\Pi$ is defined as follows:*

$\times_{i=1}^{d}(w_i, \bar{w}_i) \mapsto \times_{i=1}^{d}(w_i', \bar{w}_i')$ iff there exist k, with $1 \leq k \leq d$, an evolution rule $u \to v \in R_k$ and a migration string *$\rho \in \{1, \ldots, d\}^{|v|}$ such that*

- $u \subseteq w_k$
- $w_k' = w_k \setminus u$
- $\forall i : 1 \leq i \leq d$ *and* $i \neq k$ *implies* $w_i' = w_i$
- $\forall j : 1 \leq j \leq |v|$ *the following holds:*
  - *if* $\pi_2((v)_j) = here$ *then* $(\rho)_j = k$
  - *if* $\pi_2((v)_j) = out$ *then* $(\rho)_j = father(k)$[2]
  - *if* $\pi_2((v)_j) = in$ *then* $(\rho)_j \in children(k)$[3]
- $\forall i, 1 \leq i \leq d : \bar{w}_k' = \bar{w}_k \oplus \bigoplus_{1 \leq j \leq |v|, (\rho)_j = k} (v)_j$

Note that the size of a configuration represents an upper bound to the length of the sequences of reactions starting from that configuration. Hence, infinite sequences of reactions are not possible.

Proposition 1. *Let γ be a configuration. If $\gamma \mapsto^n \gamma'$ then $n \leq \#(\gamma)$.*

The *heating function heated* transforms the frozen objects of a configuration in active objects, and will be used in the definition of the maximal parallelism computation step.

Definition 12. *Let $\times_{i=1}^{d}(w_i, \bar{w}_i)$ be a partial configuration of Π.*

The heating function heated $: Conf_\Pi \to Conf_\Pi$ *is defined as follows:*

$heated(\times_{i=1}^{d}(w_i, \bar{w}_i)) = \times_{i=1}^{d}(w_i \oplus \bar{w}_i, \emptyset)$

[2] As $\rho \in \{1, \ldots, d\}^{|v|}$, this implies that $father(k)$ is defined.

[3] This implies that $children(k)$ is not empty.

Now we are ready to define the maximal parallelism computational step $\Mapsto$:

Definition 13. *The* maximal parallelism computational step $\Mapsto$ *over (nonpartial) configurations of Π is defined as follows: $\gamma_1 \Mapsto \gamma_2$ iff there exists a partial configuration γ' such that $\gamma_1 \mapsto^+ \gamma'$, $\gamma' \not\mapsto$ and $\gamma_2 = heated(\gamma')$.*

An operational semantics for P systems with maximal parallelism semantics has been defined for P systems in [1, 2, 10]. The main difference w.r.t. our approach is concerned with the fact that, while in this Section a maximal parallelism computational step is defined as a maximal sequence of reactions, in [1, 2, 10] no notion of reaction is provided, and the notion equivalent to the maximal parallelism computational step is defined directly by SOS rules [25]. A detailed comparison of the two approaches is beyond the scope of the present paper and deserves further investigation.

5 A Causal Semantics for P Systems

In this section we provide a causal semantics for cooperative P systems. To define a causal semantics, we follow the approach used in [18] for CCS, and in [3] for the π-calculus.

5.1 An informal explanation

The idea consists in decorating the reaction relation with two pieces of information:

- a fresh name k, that is associated to the reaction and it is taken from the set of causes $\mathcal{K}$;
- a set $H \subseteq \mathcal{K}$, containing all the names associated to the already occurred reactions, that represent a cause for the current reaction.

To keep track of the names of the already occurred reactions that may represent a cause for the reactions that may happen in the future, we introduce a notion of causal configuration that associates to each object an information on its causal dependencies. As in [3], for the sake of clarity we only keep track of the so called immediate causes, as the set of general causes can be reconstructed by transitive closure of the immediate causal relation. We will provide more explanation on this point with an example in the following part of the paper.

Now we start with an informal introduction of causality in P systems. Consider the following system with a unique membrane:

$$\Pi_1 = (\{a, b, c, d, e, f\}, [_1\ ]_1, ae, \{a \to bc, c \to d, e \to f\}, 1).$$

If we consider the reaction relation $\mapsto$ defined in the previous section, we have that the system Π_1 can perform either a reaction obtained by the application of the rule $a \to bc$ followed by a reaction obtained by the application of rule $e \to f$, or a sequence of two reactions where the application the rule $e \to f$ is followed by the application of $a \to bc$. The applications of the two rules are independent, in the sense that all the objects consumed by both the rules are already present in the initial configuration. Hence, the two rules can be applied in the same maximal parallelism step, and no one of the rules is causally dependent on the other one.

Consider now the system

$$\Pi_2 = (\{a, b, c, d, e, f\}, [_1\ ]_1, a, \{a \to bc, c \to d, e \to f\}, 1),$$

obtained from Π_1 by removing object e from the initial state. In this case, only rule $a \to bc$ can be applied. After the application of such a rule, an instance of object c is created by the application of rule $a \to bc$. Now, a further reduction step can be performed, consisting in applying rule $c \to d$. However, the applications of the two rules $a \to bc$ and $c \to d$ cannot be swapped, and the two rules cannot be applied in the same maximal parallelism computational step. This is because the object c consumed by rule $c \to d$ has been produced by rule $a \to bc$. In this case, we say that the reduction step consisting in the application of rule $c \to d$ *causally depends* on the reduction step consisting in the application of rule $a \to bc$.

If we consider again system Π_1, we have that, after the application of the two rules $a \to bc$ and $e \to f$, the rule $c \to e$ can be applied, and it is caused by the application of rule $a \to bc$.

We would like to note that the causal semantics is in some sense "finer" than the maximal parallelism step semantics, as it permits to identify exactly which rule(s) represent a cause for the execution of another rule. Consider, e.g., the system

$$\Pi_3 = (\{a, b, c, d, e, f\}, [_1\ ]_1, ae, \{a \to bc, cf \to d, e \to f\}, 1).$$

According to the maximal parallelism semantics, the two systems cannot be distinguished, as both can perform a maximal parallelism step containing two rules (i.e., $\{a \to bc, e \to f\}$), followed by a maximal parallelism step containing a single rule (resp. $\{c \to d\}$ for Π_1 and $\{cf \to d\}$ for Π_3). On the other hand, if we consider the causal semantics, we have that the application of rule $c \to d$ in Π_1 causally depends only on one of the two rules applied in the previous maximal parallelism step, i.e., $a \to bc$, whereas the application fo the rule $cf \to d$ in Π_3 causally depends on both the rules applied in the previous maximal parallelism step.

5.2 The formal definition of causal semantics

In this section we provide a formal definition of the notions introduced in the previous section.

Let $\mathcal{K}$ be a denumerable set of cause names, disjoint from the set V of objects.

Throughout this section, we let $\Pi = (V, \mu, w_1^0, \ldots, w_d^0, R_1, \ldots, R_d, i_0)$ be a P system.

To be able to define the set of causes of a reaction, we proceed in the following way: we associate a fresh (i.e., never used before) cause name to each reaction performed in the system. Then, each instance of object in a configuration of the system is decorated with the causal name of the reaction that produced it, or with $\emptyset$ if the object is already present in the initial configuration.[4] To keep track of such causal information, we introduce the notion of *causal configuration* fo a system.

Definition 14. *A* causal configuration *of Π is a tuple $z_1, \ldots, z_d$, where $z_i \in (V \times \mathcal{P}(\mathcal{K}))^*$ for $i = 1, \ldots, d$.*

We use $\times_{i=1}^{d} z_i$ to denote the causal configuration above.

The set of causal configurations of Π is denoted by $CConf_\Pi$.

We use $\gamma, \gamma', \gamma_1, \ldots$ to range over $CConf_\Pi$.[5]

Let $w_i^0 = o_{i,1} o_{i,2} \ldots o_{i,n_i}$ for $i = 1, \ldots, d$. The initial causal configuration *of Π is the configuration $\times_{i=1}^{d} (o_{i,1}, \emptyset)(o_{i,2}, \emptyset) \ldots (o_{i,n_i}, \emptyset)$.*

For example, $((a, \emptyset)(e, \emptyset))$ represents the initial causal configuration of the P system Π_1 in the previous subsection, and $((b, k_1)(c, k_1)(e, \emptyset))$ represents another configuration of Π_1, reached after the firing of rule $a \to bc$ (for the sake of clarity we omit the surrounding braces if the set of causes is a singleton).

Now we are ready to define the causal semantics for P systems. We write $\gamma \xrightarrow{h;H} \gamma'$ to denote the fact that system Π in configuration γ performs an action – to which we associate the cause name h – that is caused by the (previously occurred) actions whose action names form the set H. The cause name h is a fresh name: this means that it has not been used yet in the current computation.

The execution of an evolution rule is formalized by the notion of causal reaction relation.

Before providing the definition of causal reaction relation, we need some auxiliary definitions.

Definition 15. *The function drop : $(V \times \mathcal{P}(\mathcal{K}))^* \to V^*$ removes the causality information:*

$$\begin{aligned} &drop(\varepsilon) = \varepsilon \\ &drop((o, H)w) = o\ drop(w) \end{aligned}$$

The function drop is extended to configurations in the obvious way:

$$drop(\times_{i=1}^{d} z_i) = \times_{i=1}^{d} drop(z_i)$$

The function causes : $V \times \mathcal{P}(\mathcal{K}))^* \to \mathcal{P}(\mathcal{K})$ produces the set of causal labels in a string:

$$\begin{aligned} &causes(\varepsilon) = \emptyset \\ &causes((o, H)w) = H \cup causes(w) \end{aligned}$$

[4] For homogeneity with other classes of P systems, actually we decorate each object with a – possibly empty – set of cause names, even if, in the class of P systems considered in this paper, a single cause name is sufficient.

[5] With abuse of notation, we use $\gamma, \gamma', \gamma_1, \ldots$ to denote both partial configurations and causal configurations. It will be clear from the context to which kind of configuration we are referring to.

The function $deco : V^* \to V \times \mathcal{P}(\mathcal{K}))^*$ *decorates each object in a string with a given set of causal labels:*

$$\begin{aligned} &deco(\varepsilon, H) = \emptyset \\ &deco(ow, H) = (o, H)deco(w, H) \end{aligned}$$

Definition 16. *The* causal reaction relation $\xrightarrow{h;H}$ *over* $CConf_\Pi \times CConf_\Pi$ *is defined as follows:*

$\times_{i=1}^{d} z_i \xrightarrow{h;H} \times_{i=1}^{d} z_i'$ *iff there exist* k, *with* $1 \leq k \leq d$, *a string* $w \in (V \times \mathcal{P}(\mathcal{K}))^*$, *an evolution rule* $u \to v \in R_k$ *and a* migration string $\rho \in \{1, \ldots, d\}^{|v|}$ *such that*

- $u = drop(w)$
- $H = causes(w)$
- $w \subseteq z_k$
- $z_k' = z_k \setminus w \oplus deco(v, \{h\})$
- $\forall j : 1 \leq j \leq |v|$ *the following holds:*
  - *if* $\pi_2((v)_j) = here$ *then* $(\rho)_j = k$
  - *if* $\pi_2((v)_j) = out$ *then* $(\rho)_j = father(k)$[6]
  - *if* $\pi_2((v)_j) = in$ *then* $(\rho)_j \in children(k)$[7]
- $\forall i, 1 \leq i \leq d$ *and* $i \neq k$: $z_i' = z_i \oplus \bigoplus_{1 \leq j \leq |v|, (\rho)_j = i}((v)_j, h)$

5.3 Properties of the causal semantics

The causal semantics for the class of P systems considered in this paper enjoys some nice properties.

The first property is the *retrievability* of the maximal parallelism step semantics from the causal semantics. According to such a property, there is no loss of information when moving from the maximal parallelism to the causal semantics, as we can reconstruct the maximal parallelism semantics of a system by looking at its causal execution:

Theorem 1. $\times_{i=1}^{d}(w_i, \emptyset) \mapsto\!\!\!> \times_{i=1}^{d}(w_i', \emptyset)$ *is a maximal parallelism computational step if and only if there exist* $\gamma, \gamma' \in CConf(\Pi)$, $h_1, \ldots, h_n$, $H_1, \ldots H_n$ *such that*

- $drop(\gamma) = \times_{i=1}^{d}(w_i, \emptyset)$
- $drop(\gamma') = \times_{i=1}^{d}(w_i', \emptyset)$
- $\gamma \xrightarrow{h_1;H_1} \ldots \xrightarrow{h_n;H_n} \gamma'$
- $h_i \notin H_j$ *for all* i, j: $1 \leq i, j \leq n$
- *if there esist* h, H *such that* $\gamma' \xrightarrow{h;H}$ *then there exists* i *such that* $1 \leq i \leq n$ *and* $h_i \in H$

The other property is the so-called diamond property, stating that if two non-causally related actions can happen one after the other, then they can happen also in the other order, and at the end they reach the same system.

[6] As $\rho \in \{1, \ldots, d\}^{|v|}$, this implies that $father(k)$ is defined.
[7] This implies that $children(k)$ is not empty.

Theorem 2. *If* $\gamma \xrightarrow{h_1;H_1}_{r_1} \gamma' \xrightarrow{h_2;H_2}_{r_2} \gamma''$ *and* $h_1 \notin H_2$ *then there exists a causal configuration* γ''' *such that* $\gamma \xrightarrow{h_2;H_2}_{r_2} \gamma''' \xrightarrow{h_1;H_1}_{r_1} \gamma''$.

6 Conclusion

In this paper we tackled the problem of defining a causal semantics for a basic class of P systems. We think that the study of the causal dependencies that arise between the actions performed by a system is of primary importance for models inspired by the biology, because of its possible application to the analysis of complex biological pathways.

This paper represents a first step in this direction, but a lot of work remains to be done. For example, if we move to other classes of membrane systems, such as, e.g., P systems with promoters and inhibitors, we have to deal with more involved causal relations among reactions, and it could happen that some of the properties enjoyed by the causal semantics for basic P systems presented in this work no longer hold. Another interesting research topic is the investigation of the causal semantics for classes of P systems whose membrane structure is dynamically evolving (e.g., we can consider dissolution rules, duplication, gemmation or either brane-like operations). Once we have completed the definition of a causal semantics for systems with an evolving structure, we will start investigating the causal dependencies arising in biological pathways involving membranes, such as, e.g., the LDL Cholesterol Degradation Pathway [19], that was modeled in P systems in [6].

References

1. O. Andrei, G. Ciobanu, and D. Lucanu. Operational semantics and rewriting logic in membrane computing. Proceedings SOS Workshop, ENTCS, 2005.
2. O. Andrei, G. Ciobanu, and D. Lucanu. A rewriting logic framework for operational semantics of membrane systems. *Theoretical Computer Science*, 373(3):163–181, 2007.
3. M. Boreale and D. Sangiorgi. A Fully Abstract Semantics for Causality in the π-Calculus. Acta Inf. 35(5): 353-400, 1998. An extended abstract appeared in Proc. STACS 1995: 243-254.
4. N. Busi. Using well-structured transition systems to decide divergence for catalytic P systems. *Theoretical Computer Science*, 372(2-3): 125-135, 2007.
5. N. Busi and R. Gorrieri. A Petri Net Semantics for π- calculus. In Proc. Concur'95 , LNCS 962, Springer, 145-159, 1995.
6. N. Busi and C. Zandron. Modeling and analysis of biological processes by mem(brane) calculi and systems. Proceedings of the Winter Simulation Conference (WSC 2006), ACM, 2006.
7. N. Busi. Towards a causal semantics for Brane Calculi. Proc. Fifth Brainstorming Week on Membrane Computing, Sevilla, 2007, to appear.

8. C.S. Calude and G. Păun. *Computing with Cells and Atoms.* Taylor & Francis, London, 2001.
9. L. Cardelli. Brane Calculi - Interactions of biological membranes. Proc. Computational Methods in System Biology 2004 (CMSB 2004), LNCS 3082, Springer, 2005.
10. G. Ciobanu, O. Andrei, and D. Lucanu. Structural Operational Semantics of P Systems. Proc. of Sixth International Workshop on Membrane Computing (WMC6), LNCS, 2005.
11. Ph. Darondeau and P. Degano. Causal Trees. in Proc. *ICALP'89*, LNCS 372, Springer, 234-248, 1989.
12. J. Dassow and G. Păun. On the Power of Membrane Computing. J. Univ. Comput. Sci. 5(2), 1999.
13. P. Degano, R. De Nicola, and U. Montanari. Partial ordering descriptions and observations of nondeterministic concurrent processes. in Proc. REX School/Workshop on Linear Time, Branching Time and Partial Order in Logic and Models of Concurrency, LNCS 354, 438-466, 1989.
14. P. Degano and C. Priami. Causality for Mobile Processes. in Proc. *ICALP'95*, LNCS 944, Springer, 660-671, 1995.
15. P. Degano and C. Priami. Non Interleaving Semantics for Mobile Processes. Theoretical Computer Science 216(1-2): 237-270, 1999.
16. V. Danos and S. Pradalier. Projective Brane Calculus. Proc. Computational Methods in System Biology 2004 (CMSB 2004), LNCS 3082, Springer, 2005.
17. M.L. Guerriero and C. Priami. Causality and Concurrency in Beta-binders TR-01-2006 The Microsoft Research - University of Trento Centre for Computational and Systems Biology, 2006.
18. A. Kiehn. Proof Systems for cause based equivalences. In Proc. MFCS'93, LNCS 711, Springer, 1993.
19. H. Lodish, A. Berk, P. Matsudaira, C.A. Kaiser, M. Krieger, M. P. Scott, S. L. Zipursky and J. Darnell. *Molecular cell biology.* W.H. Freeman and Company, 4th edition, 1999.
20. R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.
21. R. Milner, J. Parrow, and D. Walker. A Calculus of Mobile Processes. *Information and Computation* 100, 1-77, 1992.
22. G. Păun. Computing with membranes: an Introduction. Bull. EATCS 67, 1999.
23. G. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61(1):108–143, 2000.
24. G. Păun. *Membrane Computing. An Introduction.* Springer, 2002.
25. G.D. Plotkin. Structural operational semantics. *Journal of Logic and Algebraic Programming* 60, 17–139, 2004.
26. C. Priami and P. Quaglia. Beta binders for biological interactions. In Proc. of Computational Methods in Systems Biology, LNCS 3082, 2033, 2005.
27. C. Priami and P. Quaglia. Operational patterns in beta-binders. T. Comp. Sys. Biology, 1:5065, 2005.