Describing Membrane Computations with a Chemical Calculus *

Péter Battyányi, György Vaszil

Department of Computer Science, Faculty of Informatics
University of Debrecen
Kassai út 26, 4028 Debrecen, Hungary
{battyanyi.peter, vaszil.gyorgy}@inf.unideb.hu

Summary. Membrane systems are nature motivated computational models inspired by certain basic features of biological cells and their membranes. They are examples of the chemical computational paradigm which describes computation in terms of chemical solutions where molecules interact according to rules defining their reaction capabilities. Chemical models can be presented by rewriting systems based on multiset manipulations, and they are usually given as a kind of chemical calculus which might also allow non-deterministic and non-sequential computations. Here we study membrane systems from the point of view of the chemical computing paradigm and show how computations of membrane systems can be described by such a chemical calculus.

1 Introduction

The history of the chemical computing paradigm goes back to the introduction of the Gamma programming language by Bânatre and Le Métayer in [3, 4]. They describe computations in terms of a symbolic chemical solution of molecules with possible reactions between them. The molecules represent the data, the reactions represent their transformations, and the Brownian motion of molecules in the solution represents the execution model of the program.

The general idea behind the chemical paradigm (and the purpose of the introduction of the Gamma formalism) is to be able to express algorithms without the sequentiality which is not inherently necessary, which is not inherently "built into" the problem that the algorithm needs to solve, or in other words, the sequentiality which is the consequence of the given computational model and not related to the logic of the solution of the problem.

The idea was developed further in several different directions, on of them was realized by the chemical abstract machine of Berry and Boudol in [5] where they

* Research supported in part by the Hungarian Scientific Research Fund, "OTKA", grant no. K75952, and by the European Union through the TÁMOP-4.2.2.C-11/1/KONV-2012-0001 project which is co-financed by the European Social Fund.

also introduce the notion of membranes which encapsulate subsolutions forcing the reactions to occur in locally isolated ways. The role of membranes was underlined by Păun in [7] where membrane systems (P systems) were introduced emphasizing the importance of the hierarchical structure of different compartments or regions enclosing different molecules which react, evolve according to different rules and they only cross the membranes or region boundaries in a restricted and controlled manner.

Since its introduction, the theory of P systems became an extensive and well established research field, one of the most important and most popular areas of natural computing which also evolved in several different directions and into different subfields. The evolution of the area made its relationship to other chemical computing models and formalisms less apparent, so it might be of interest to examine them from the point of view of the chemical computing paradigm, and to point out the links between P systems and other chemical computational models, as we attempt in the present paper.

This approach could be beneficial in different ways. By being able to translate chemical programs, or chemical computing formalisms (like Gamma) to membrane systems, we could provide something like a high-level programming language for P systems which could serve as an elegant and efficient way of presenting P system algorithms. A preliminary study of turning certain simple Gamma programs to P systems was initiated in [11]. On the other hand, by being able to describe membrane systems using one of the chemical computing formalisms (as we attempt in this paper), we would be able to use the tools and techniques developed for the many different types of chemical calculi to reason about membrane systems and their computations.

In what follows we first give a short introduction to the Gamma formalism and to the notions of membrane systems and their computations. Then we present the γ-calculus from [1], and show how computations of certain membrane systems can be described in this formalism.

2 Preliminaries and Definitions

We assume that the reader is familiar with the basics of formal language theory and membrane computing; for more information, we refer to the monograph [10], and the handbooks [9] and [8].

In the following, we briefly review the notions and the notation we will use. An alphabet is a finite non-empty set of symbols. Given an alphabet V, we denote the set of strings over V by V^* . If the empty string, ε, is not included, then we use the notation V^+.

A finite multiset over an alphabet V is a mapping $M : V \to \mathbb{N}$ where $\mathbb{N}$ denotes the set of non-negative integers, and $M(a)$ for $a \in V$ is said to be the multiplicity of a in M. The support of M is the set $supp(M) = \{a \in V \mid M(a) \geq 1\}$. If $supp(M)$ is a finite set, then M is called a finite multiset. The set of all finite

multisets over the set V is denoted by $\mathcal{M}(V)$. We say that $a \in M$ if $a \in supp(M)$, $M_1 \subseteq M_2$ if $supp(M_1) \subseteq supp(M_2)$ and for all $a \in V$, $M_1(a) \leq M_2(a)$. The union of two multisets over V is defined as $(M_1 \cup M_2)$ where for all $a \in V$, $(M_1 \cup M_2)(a) = M_1(a) + M_2(a)$, the difference is defined for $M_2 \subseteq M_1$ as $(M_1 \setminus M_2)$ where $(M_1 \setminus M_2)(a) = M_1(a) - M_2(a)$ for all $a \in V$.

In what follows, we usually enumerate the not necessarily distinct elements $a_1, \ldots, a_n$ of a multiset as $M = (a_1, \ldots, a_n)$, but the multiset M can also be represented by any permutation of a string $w = a_1^{M(a_1)} a_2^{M(a_2)} \ldots a_n^{M(a_n)} \in V^*$, where if $M(x) \neq 0$, then there exists j, $1 \leq j \leq n$, such that $x = a_j$. The empty multiset is denoted by $\emptyset$ as in the case of the empty set.

Both P systems and chemical calculi are implemented as rewriting systems. A rewriting system is a pair $\mathcal{A} = \{\Sigma, (\rightarrow_i)_{i \in I}\}$, where Σ is a set and $(\rightarrow_i)_{i \in I}$ is a set of binary relations defined on Σ. The relations $(\rightarrow_i)_{i \in I}$ are called reduction relations. In the rewriting systems considered in this paper the set § is a set of multisets of elements over an alphabet O or, in the case of the chemical formalism, multisets of terms of some calculus, and $\rightarrow$ is a single binary relation rendering multisets to multisets. It is supposed that the reduction relation $\rightarrow$ is compatible with the term formation rules in the γ-calculus, and, in the case of membrane systems, it extends itself to a relation on configurations.

2.1 The Gamma-formalism

Several attempts have been made to establish a programming language suitable for programming massively parallel architectures. The aim of the Gamma-formalism is to provide the programmer with a high-level programming language deprived of all artificial constraint for sequentiality, in which the parallel aspect is left implicit. The Gamma-formalism, as a programming language, is a tool for multiset manipulation. The only data structure, in the untyped version, is that of multisets, and the programs are collections of pairs consisting of reaction conditions and actions. The following presentation is based mainly on [2].

The meaning of the Γ-function can be defined as follows:

$$\Gamma(R, A)(M) = \begin{cases} \Gamma(R, A)((M \setminus (x_1, \ldots, x_n)) \cup A(x_1, \ldots, x_n)), \\ \qquad \text{if } x_1, \ldots, x_n \in M \text{ and } R(x_1, \ldots, x_n), \\ M \quad \text{otherwise.} \end{cases}$$

For example, the Γ-program below computes the maximal element of a set of integers M:

$$\begin{aligned} maxset(M) = \Gamma((R, A))(M) \text{ where} \\ R(x, y) = x \leq y \\ A(x, y) = (y) \end{aligned}$$

The Boolean function R describes a relation to be satisfied by the selected elements x and y. If $R(x, y)$ is true, then x and y are replaced by the result of the reaction defined by the function A. In this case, the element of the multiset

(x, y) is chosen which is not less than the other. The function R is called the reaction condition and the action A is the result of the reaction. The computation terminates when no more reactions are possible. In the present situation this leaves the computation with a one element multiset as the result, which is the maximal element of M.

The next introductory example finds the prime numbers up to a given number n:

$$\begin{aligned} \mathit{sieve}(n) = \Gamma((R, A))(2, \ldots, n) \text{ where} \\ R(x, y) = x \mathit{\ divides\ } y \\ A(x, y) = (x) \end{aligned}$$

The reaction condition is fulfilled for x and y, if y is a multiple of x. In this case the multiset (x, y) is removed from the original multiset and is replaced by the multiset (x). In effect, this means that a stable condition is reached if there are no more pairs (x, y) such that x divides y, which is satisfied if and only if the resulting multiset consists of the prime numbers not greater than n.

The definition of Γ reflects the way the program reaches its halting point: if there is at least one multiset $(x_1, \ldots, x_n)$ such that $R(x_1, \ldots, x_n)$ holds, then $\Gamma(R, A)(M)$ is applicable and the result is the same as the value determined by application $\Gamma(R, A)((M \backslash (x_1, \ldots, x_n)) \cup A(x_1, \ldots, x_n))$, if it exists.

Remark 1. Of course, there can be Γ-programs which do not terminate. In simple cases like the above ones, termination can be proven by an application of the Dershowitz–Manna lemma, see [6]. This lemma asserts that if $(S, <)$ is a well-founded ordered set with the strict (that is, irreflexive and transitive) partial-order $<$, then $\mathcal{M}(S)$, the set of all finite multisets over S with the strict partial ordering $\ll$ is also well-founded, where $\ll$ is defined as follows. Let $M, N, \in \mathcal{M}(S)$. We say that $M \ll N$ if there are X, $Y \in \mathcal{M}(S)$ such that

- $X \neq \emptyset$,
- $N = (M \backslash X) \cup Y$,
- $(\forall y \in Y)(\exists x \in X)(y < x)$.

As the well foundedness of a set means that there exists no infinite decreasing sequence of its elements, the lemma applies to the examples above, because both by the *maxset* and the *sieve* programs, the number of elements of the multisets obtained as the intermediate results of the transformation forms a strictly decreasing sequence.

2.2 Membrane Systems

A P system is a structure of hierarchically embedded membranes, each having a label and enclosing a region containing a multiset of objects and possibly other membranes. The out-most membrane which is unique and usually labeled with 1, is called the skin membrane. The membrane structure is denoted by a sequence of matching parentheses where the matching pairs have the same label as the membranes they represent. If $x \in \{[_i,]_i \mid 1 \leq i \leq n\}^*$ is such a

string of matching parentheses of length $2n$, denoting a structure where membrane i contains membrane j, then $x = x_1\ [_i\ x_2\ [_j\ x_3\ ]_j\ x_4\ ]_i\ x_5$ for some $x_k \in \{[_l,]_l \mid 1 \leq l \leq n,\ l \neq i, j\}^*,\ 1 \leq k \leq 5$. If membrane i contains membrane j, and there is no other membrane, k, such that k contains j and i contains k (x_2 and x_4 above are strings of matching parentheses themselves), then we say that membrane i is the parent membrane of j. A membrane m is called elementary if it contains no membrane, in this case $x = x_1\ [_m\ ]_m\ x_2$.

The evolution of the contents of the regions of a P system is described by rules associated to the regions. Applying the rules synchronously in each region, the system performs a computation by passing from one configuration to another one. The rules are multiset rewriting rules given in the form of $u \to v$ where u, v are multisets, and they are applied in the maximal parallel manner, that is, as many rules are applied in each region as possible. The end of the computation is defined by halting: A P system halts when no more rules can be applied in any of the regions, the result is a number, the number of objects in an elementary membrane labeled as output.

Definition 1 *A P system of degree $n \geq 1$ is a construct*

$$\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, out)$$

where

- *O is an alphabet of objects,*
- *μ is a membrane structure of n membranes,*
- *$w_i \in \mathcal{M}(O),\ 1 \leq i \leq n$, are the initial contents of the n regions,*
- *$R_i,\ 1 \leq i \leq n$, are the sets of evolution rules associated to the regions, they are of the form $u \to v$ where $u \in \mathcal{M}(O)$ and $v \in \mathcal{M}(O \times TAR)$ where $TAR = \{here, out\} \cup \{in_j \mid 1 \leq j \leq n\}$, and*
- *$out \in \{1, \ldots, n\}$ is the label of an elementary membrane, the output membrane.*

The evolution rules of the system are applied in the nondeterministic, maximally parallel manner to the n-tuple of multisets of objects constituting the configuration of the system. The n-tuple $(w_1, \ldots, w_n)$ is the initial configuration of Π. For two configurations $C_1 = (u_1, \ldots, u_n)$ and $C_2 = (v_1, \ldots, v_n)$, we can obtain C_2 from C_1 by applying the rules of $R_1, \ldots, R_n$ in the following way. The application of $u \to v \in R_i$ in the region i means to remove the objects of u from u_i and add the new objects specified by v to the system. The objects of v should be added to the regions as specified by the target indicators associated to them: If v contains a pair $(a, here) \in O \times TAR$, then a is placed in region i, the region where the rule is applied. If v contains $(a, out) \in O \times TAR$, then a is added to the contents of the parent region of region i; if v contains $(a, in_j) \in O \times TAR$ for some region j which is contained inside the region i (so region i is the parent region of region j), then a is added the contents of region j.

The objects evolve simultaneously, and the rules by which they evolve are chosen nondeterministically, but in a maximally parallel manner. This means that

the rules are chosen in such a way, that in each region objects are assigned to rules, as many objects to as many rules as possible, and the assignment is maximal in the sense that no other rule can be applied to the remaining (unassigned) objects which appear unchanged in the next configuration.

A sequence of transitions between configurations is called a computation. A computation is successful if it halts, that is, if it reaches a configuration where no application of any of the rules are possible. In this case, the result is the number of objects which are present in the output region in the halting configuration.

3 Writing Gamma Programs for Membrane Systems

The Γ-formalism has been the source of inspiration for the higher-order chemical model of Banâtre et al., the so-called γ-calculus. The following presentation is based on [1].

Two syntactical elements, molecules and patterns, are defined simultaneously in the following definition.

Definition 1. *The syntactical elements of molecules, denoted by M, and patterns, denoted by P, are defined as*

$$M = x \mid \gamma(P)[M_1].M_2 \mid (M_1, M_2) \mid \langle M \rangle$$
$$P = x \mid (P_1, P_2) \mid \langle P \rangle,$$

where x is a variable standing for any molecule, $\gamma(P)[M_1].M_2$ is a γ-abstraction with reaction condition M_1 and result M_2. The operator "," is commutative and associative, and $\langle M \rangle$ is called a solution.

A solution $\langle M \rangle$ encapsulates the molecule M: molecules inside the solution are insulated from molecules outside the solution. However, the contents of solutions can be changed by reactions which occur inside the solution.

The fundamental rule of the γ-calculus is that of a reaction. Prior to defining reactions, we need some auxiliary notions.

Definition 2. *A substitution is a mapping $\phi : Var \to \mathcal{M}$, if $\mathcal{M}$ is the set of all molecules and Var represents the set of variables. We can define the application of a substitution to a molecule as follows:*

$$\phi x = \phi(x)$$
$$\phi(M_1, M_2) = \phi M_1, \phi M_2$$
$$\phi\langle M \rangle = \langle \phi M \rangle$$
$$\phi(\gamma(P)[C].M) = \gamma(P)[C].\phi' M,$$

where ϕ' is obtained from ϕ by removing from the domain all the variables which occur in P.

A molecule is termed inert if no reaction can take place within it. The first argument of *match* is a pattern, the other one is a molecule. Its value is a substitution.

Definition 3. *Let x denote a variable, P a pattern, and M a molecule. Then we define*

$$\begin{aligned} match(x, M) &= \{x \mapsto M\} \\ match(\langle P\rangle, \langle M\rangle) &= match(P, M) \text{ provided } inert(M) \\ match((P_1, P_2), (M_1, M_2)) &= match(P_1, M_1) \circ match(P_2, M_2) \\ match(P, M) &= \boldsymbol{fail} \text{ in every other case.} \end{aligned}$$

The operation $\circ$ is the function composition with the additional stipulation that $\boldsymbol{fail} \circ x = x \circ \boldsymbol{fail} = \boldsymbol{fail}$.

We are in a position now to define the main rule of the calculus, the reaction rule.

Definition 4. *Let*

$$\gamma(P)[C].M, N \to \phi M, \qquad (\gamma)$$

where $match(P, N) = \phi$ and $\phi(C) \to^* true$.

Besides the γ-rule some additional rules are applied.

Definition 5. *Let M, M_1, and M_2 be any molecule. Then we have the following rules:*

$$\frac{M_1 \to M_2}{M, M_1 \to M, M_2} \quad (locality), \quad \frac{M_1 \to M_2}{\langle M_1\rangle \to \langle M_2\rangle} \quad (solution).$$

Moreover, we identify molecules with respect to commutativity and associativity, that is, any reduction is understood modulo $\equiv$, where

$$M_1, (M_2, M_3) \equiv (M_1, M_2), M_3 \quad \text{and} \quad M_1, M_2 \equiv M_2, M_1.$$

The γ-rule is a so-called one-shot rule: after taking part in a reduction, the γ-abstraction disappears. To obtain a syntactic tool resembling the Γ-operator of the previous section, we introduce a new operator which does not vanish in the course of the reduction.

Let $replace\ P\ by\ M\ if\ C$, or $replace(P, C, M)$ in short, be an operator obeying the following reduction rule:

$$replace\ P\ by\ M\ if\ C, N \to replace\ P\ by\ M\ if\ C, \phi(M), \quad (replace)$$

where $C \to^* true$ and $match(P, N) = \phi$.

With the notation introduced, we can construct the term finding the maximal element of a given set of numbers:

Example 1. Let $M = \{3, 5, 8, 10, 12\}$. Then

$$(replace\ \langle\langle x\rangle, \langle y\rangle\rangle \text{ by } \langle y\rangle \text{ if } x \leq y, \langle 3\rangle, \langle 5\rangle, \langle 8\rangle, \langle 10\rangle, \langle 12\rangle)$$

finds the maximum element of the set M.

Observe that in the example above the integers are represented as solutions. Otherwise the term "$replace\ x, y$ by y if $x \leq y$" would match any molecules x and y. In this case, it does not seem to be a problem since $x \leq y$ makes sense if x and y are integers. As another example we compute the largest prime number up to an integer n.

Example 2.

$$\begin{aligned} largestprime(n) = \\ \text{let } sieve &= replace\ \langle\langle x\rangle, \langle y\rangle\rangle \text{ by } \langle x\rangle \text{ if } x \text{ div } y \text{ in} \\ \text{let } max &= replace\ \langle\langle x\rangle, \langle y\rangle\rangle \text{ by } \langle x\rangle \text{ if } x \leq y \text{ in} \\ &(\langle\langle 2\rangle, \langle 3\rangle, \ldots, \langle n\rangle, sieve\rangle, \gamma\langle x\rangle(x, max)) \end{aligned}$$

Observe that in this example above the pattern standing in the last γ-term is a solution $\langle x\rangle$. By the definition of *match*, only inert solutions are able to match with this pattern. This amounts to a means of governing the sequence of reductions in a molecule: first the prime numbers are selected with the term *sieve*, then the term *max* chooses the maximum among them.

Now we can turn to the problem of establishing a relation between the computations in the γ-calculus and computations in membrane systems. We consider the systems defined in Section 2.2, that is P systems of the form

$$(O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, i_o)$$

where n is the number of membranes in the structure. We may assume $i_o = n$, which means the output membrane is the n-th membrane.

The membrane structure can be uniquely described by setting the parent member to each element. To this end, let $par : \{1, \ldots, n\} \to \{1, \ldots, n\} \cup \{nil\}$ be the function with the interpretation: if $par(j) = k$, then the k-th membrane is the parent of the membrane numbered j. The parent of the outer membrane is nil, that is, $par(1) = nil$.

We assume, as in Definition 1, that the rules R_i belonging to membrane i are of the form $u \to v$, where $u \in \mathcal{M}(O)$ and $v \in \mathcal{M}(O \times \{here\}) \cup \mathcal{M}(O \times \{out\}) \cup \mathcal{M}(O \times \{in_j\})$, where $par(j) = i$.

First, we label the elements of each membrane, displaying explicitly the number of the membrane the element belongs to. Thus, let $lab : O \to \{(k : a) \mid 1 \leq k \leq n, a \in O\}$ be such that

$$lab(a) = (k : a),$$

if a is contained by the regions enclosed by membrane k. Let us determine how these labels are inherited through a rule $u \to v$.

Definition 6. *Let* $a \in O \times \{here\} \cup O \times \{out\} \cup O \times \{in_j\}$, *assume* $tar \in \{here, out, in_j\}$, *where* $1 \leq k \leq n$ *and* $par(j) = k$. *Then*

$$lab(k, a, tar) = \begin{cases} k & \text{if } tar = here \\ j & \text{if } tar = in_j \\ par(k) & \text{if } tar = out \text{ and } k > 1 \\ undefined & \text{if } tar = out \text{ and } k = 1. \end{cases}$$

Let k be a membrane label for some $1 \leq k \leq n$. We define a transformation $h_k(u \to v)$ of a rule $u \to v$ in region k into some γ-term. First, settle h_k for the elements in u and v. Let O be the alphabet of the membrane system. We define the image of an element of O in the corresponding γ-calculus. If $a \in O$, then

$$h_k(a) = (k : a),$$

otherwise

$$h_k(a, tar) = \begin{cases} \underline{(i : a)} & \text{where } i = lab(k, a, tar) \text{ and } tar \neq out \text{ or } k > 1, \\ \lambda & \text{if } tar = out \text{ and } k = 1. \end{cases}$$

The function h_k propagates itself to multisets, thus

$$h_k(w_1 \dots w_l) = h_k(w_1) \dots h_k(w_l),$$

if $w = w_1 \dots w_l$ and $w \in \mathcal{M}(O)$ or $w \in \mathcal{M}((O \times \{here\}) \cup \mathcal{M}(O \times \{out\}) \cup \mathcal{M}(O \times \{in_j\})$. For technical reasons we add a new constant, say, 1 to each multiset obtained from u and v, thus

$$h_k(u \to v) = replace\ (h_k(u), 1)\ with\ (h_k(v), 1).$$

Accomplishing this for every membrane labeled by k, we assign the multiset $((k : u_1), \dots, (k : u_{i_k}), (k : 1), h_k(r_1), \dots, h_k(r_{j_k}))$ to region k, where $u_1, \dots, u_{i_k}$ were the elements and $r_1, \dots, r_{j_k}$ were the rules corresponding to the region.

Let us denote by $gamma(\Pi)$ the gamma formalism assigned to the membrane system Π. The transformation above is injective, thus, given a γ-structure Γ obtained in this way, we can uniquely identify the membrane structure Π such that $gamma(\Pi) = \Gamma$. In the sequel, if no confusion occurs, we write a_k instead of $(k : a)$ and c instead of $(k : c)$, if c is a fixed element of the base set.

Example 3. Let Π be the membrane system $\Pi = (\{a, b, c\}, \mu, aa, \emptyset, R_1, \emptyset, 2)$ where $\mu = [\ [\ ]_2\ ]_1$, and $R_1 = \{a \to (a, here)(b, in_2)(c, in_2)^2,\ a^2 \to (a, out)^2\}$.

Now

$$h_1(a) = a_1,$$
$$h_1(a \to (a, here)(b, in_2)(c, in_2)^2) = replace\ (a_1, 1)\ with\ (\underline{a_1}, \underline{b_2}, \underline{c_2}^2, 1),$$
$$h_1(aa \to (a, out)^2) = replace\ (a_1, a_1, 1)\ with\ 1.$$

In what follows we simulate maximal parallel reduction sequences of the membrane system with reduction sequences in the chemical calculus γ. Besides the translations of the multiset reduction steps the process must be prepared to simulate the next maximal parallel sequence of reductions, as well. To this end, the underlined symbols must be transformed back into usual ones. Some care is needed though in the process, the replace operators assigned to rules of the membrane system should not interfere with this stage. The introduction of the additional constants 1 and 0 serves exactly this synchronization purpose. When 1 is present in the corresponding molecule, then the operators corresponding to the rules of the P system are active, and, in the case when 1 is exchanged for 0, the operators restoring the original elements of the alphabet O come into effect.

Let us define the terms giving us back the elements of the original alphabet O after simulating a maximal parallel computation sequence.

Definition 7. *Let O_{lab} be the set of elements obtained from O by completing the labeling process for every $1 \leq k \leq n$. Then*

$$d(O_{lab}) = \bigcup \{replace\ \underline{a}, 0\ with\ a, 0 \mid a \in O_{lab}\}.$$

Now we determine the term controlling the whole process.

Theorem 2. *Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, i_n)$ be a membrane system, assume $gamma(\Pi)$ is the multiset defined as above, and $d(O_{lab})$ is the set of operators presented in the previous definition. Then, for the term*

$$M = (\langle gamma(\Pi)\rangle, replace\ \langle x, 1\rangle\ with\ \langle x, 0, d(O_{lab})\rangle,$$
$$replace\ \langle x, 0, d(O_{lab})\rangle\ with\ \langle x, 1\rangle),$$

Π comes to a halt with result $(a_1^{i_1}, a_2^{i_2}, \ldots, a_j^{i_j})$ if and only if M normalizes with $(n : a_1)^{i_1}, (n : a_2)^{i_2}, \ldots, (n : a_j)^{i_j}$ as its elements with label n.

Proof. First of all, observe that the mapping *gamma* is a bijection between membrane systems having the structure $(O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, i_n)$ and chemical structures over the alphabet $\bigcup_{i=1}^{n} \{h_i(O) \cup h_i(O \times \{tar\})\}$, where $tar \in \{here, out, in_j\}$ if $par(j) = i$. A little more is true. Namely, let $M \to M_1 \to \ldots$ be a reduction sequence starting from M. We call a reduction step $M_i \to M_j$ a 1-step, if the maximal solution in M_i contains 1, otherwise it is a 0-step. Let $\langle S_0\rangle$, $\langle S_1\rangle, \ldots$ be the sequence of the maximal solutions of $M \to M_1 \to M_2 \to \ldots$ and assume, for Π_0, $\Pi_1, \ldots$, $\Pi_k, \ldots$, $gamma(\Pi_0) = S_0$, $gamma(\Pi_1) = S_1, \ldots$, $gamma(\Pi_k) = S_k, \ldots$. The notion of 1-step and 0-step transforms to S_0, $S_1, \ldots$ analogously. We denote by $S' \to_1 S''$ ($S' \to_0 S''$) if a 1-step reduction from S' yields S''. Then we have the following statement.

$$S_i \to^1 S_j \text{ if and only if } \Pi_i \to \Pi_j. \tag{1}$$

The proof can be done by induction on the lengths of the reduction sequences on the two sides. Assume now $\Pi_0 \to \Pi_1 \to \Pi_2 \to \ldots \to \Pi_n$ is a reduction sequence

yielding a maximal parallel reduction step from Π_0. Then, by (1), there can be no 1-reductions starting from $gamma(\Pi_n) = S_n$. But this means S_n has arrived to a normal form. Below, for a multiset S, let $\overline{S}$ denote the multiset $S - (0,1)$. Moreover, if S is a multiset possibly with underlined elements, then let $plain(S)$ denote the multiset consisting of the same elements with the underlining removed. Thus

$$\begin{aligned}
M = {} & (\langle S_0\rangle, replace\ \langle x,1\rangle\ with\ \langle x,0,d(O_{lab})\rangle,\\
& replace\ \langle x,0,d(O_{lab})\rangle\ with\ \langle x,1\rangle) \rightarrow\\
& \ldots\\
\rightarrow {} & (\langle S_n\rangle, replace\ \langle x,1\rangle\ with\ \langle x,0,d(O_{lab})\rangle,\\
& replace\ \langle x,0,d(O_{lab})\rangle\ with\ \langle x,1\rangle) \rightarrow\\
\rightarrow {} & (\langle \overline{S}_n,0,d(O_{lab})\rangle, replace\ \langle x,1\rangle\ with\ \langle x,0,d(O_{lab})\rangle,\\
& replace\ \langle x,0,d(O_{lab})\rangle\ with\ \langle x,1\rangle) \rightarrow\\
& \ldots\\
\rightarrow {} & (\langle plain(\overline{S}_n),0,d(O_{lab})\rangle, replace\ \langle x,1\rangle\ with\ \langle x,0,d(O_{lab})\rangle,\\
& replace\ \langle x,0,d(O_{lab})\rangle\ with\ \langle x,1\rangle) \rightarrow\\
\rightarrow {} & (\langle plain(\overline{S}_n),1)\rangle, replace\ \langle x,1\rangle\ with\ \langle x,0,d(O_{lab})\rangle,\\
& replace\ \langle x,0,d(O_{lab})\rangle\ with\ \langle x,1\rangle)
\end{aligned}$$

Here, $\langle \overline{S}_n,0,d(O_{lab})\rangle$ has no choice but to reduce to $\langle plain(\overline{S}_n),0,d(O_{lab})\rangle$, which cannot be reduced any further. Then the action of $\langle plain(\overline{S}_n),0,d(O_{lab})\rangle$ and $replace\ \langle x,0,d(O_{lab})\rangle\ with\ \langle x,1\rangle$ yields $\langle plain(\overline{S}_n),1)\rangle$, which means a new maximal parallel reduction step of the membrane system obtained so far can be simulated again. This gives an account of the correspondence set up between membrane system reductions and reductions in chemical structures. □

4 Conclusion

The chemical computational paradigm describes computations as reactions between molecules which freely interact in a symbolic chemical solution. We have given a short overview of a chemical calculus from [1], and shown how computations of membrane systems can be described in this setting. Thus, we have taken some initial steps in the direction of being able to describe membrane systems using the chemical computing formalisms and being able to use the tools and techniques developed for chemical calculi to reason about membrane systems and their computations.

References

1. J.P. Banâtre, P. Fradet, Y. Radenac, Principles of chemical computing. *Electronic Notes in Theoretical Computer Science* 124 (2005) 133–147.

2. J.P. Banâtre, P. Fradet, Y. Radenac, Generalized multisets for chemical programming. *Mathematical Structures in Computer Science* 16(4) (2006), 557 – 580
3. J.P. Banâtre, D. Le Métayer, A new computational model and its discipline of programming. *Technical Report RR0566*, INRIA (1986).
4. J.P. Banâtre, D. Le Métayer, Programming by multiset transformation. *Communications of the ACM* 36 (1993), 98–111.
5. G. Berry, G. Boudol, The chemical abstract machine. *Theoretical Computer Science* 96 (1992), 217–248.
6. N. Dershowitz, Z. Manna, Proving termination with multiset orderings. *Communications of the ACM* 22(8) (1979), 465–476.
7. Gh. Păun, Computing with membranes. *Journal of Computer and System Sciences* 61 (2000), 108–143.
8. Păun, G., Rozenberg, G., Salomaa, A. (eds): The Oxford Handbook of Membrane Computing, Oxford University Press (2010)
9. Rozenberg, G. Salomaa, A. (eds): Handbook of Formal Languages, Springer Berlin (1997)
10. Salomaa, A.: Formal Languages, Academic Press, New York (1973)
11. M. Fésüs, Gy. Vaszil, Chemical programming and membrane systems. In: *Proc. 14th International Conference on Membrane Computing*, Institute of Mathematics and Computer Science, Academy of Moldova, 2013, 313–316.