Nat Comput (2016) 15:521–532
DOI 10.1007/s11047-016-9570-5

Simulating P systems with membrane dissolution in a chemical calculus

Bogdan Aman[1] · Péter Battyányi[2] · Gabriel Ciobanu[1] · György Vaszil[2]

Published online: 18 July 2016

Abstract We present a transformation of membrane systems, possibly with promoter/inhibitor rules, priority relations, and membrane dissolution, into formulas of the chemical calculus such that terminating computations of membranes correspond to terminating reduction sequences of formulas and vice versa. In the end, the same result can be extracted from the underlying computation of the membrane system as from the reduction sequence of the chemical term.

Keywords Membrane systems · Membrane dissolution · Chemical computing paradigm · Chemical calculus

1 Introduction

In the present paper we continue the investigations of Andrei et al. (2006, 2007) concerning the possibility of defining the semantics of membrane systems with rewriting logic (Agrigoroaiei and Ciobanu 2011; Andrei et al. 2006) in order to obtain a logical description of membrane system computations.

The direct precedent of our work is Battyányi and Vaszil (2014) where a logical description of simple membrane systems was given using the γ-calculus of Banâtre and Le Métayer (1986) (see also Banâtre et al. 2005 for more details), whose aim was to free the expression of algorithms from the sequentiality which is not inherently present in the problem to be solved, that is, the sequentiality which is implied by the structure of the computational model on which the given algorithm is to be performed. They called their calculus chemical calculus, and the underlying computational paradigm the chemical paradigm of computation while the execution model behind them closely resembles the way chemical reactions take place in chemical solutions. A chemical "machine" can be thought of as a symbolic chemical solution where data can be seen as molecules and operations as chemical reactions. If some molecules satisfy a reaction condition, they are replaced by the result of the reaction. If no reaction is possible, the program terminates. Molecules interact freely according to reaction rules which results in an implicitly parallel, non-deterministic, distributed model.

Chemical solutions are represented by multisets of abstract objects, thus, chemical computations can be thought of as a series of multiset transformations which is very similar to the way we can represent the computations of membrane systems, which are abstract computing devices introduced by Păun (2000). A membrane system consists of a structure of hierarchically arranged membranes which delimit regions containing multisets of objects from a certain object alphabet, or they can also contain (sub)regions, delimited by other membranes containing further objects inside. Objects can be changed or moved between the regions based on object evolution and communication rules which are applied to the multisets inside the regions in parallel. The design of these

✉ György Vaszil
vaszil.gyorgy@inf.unideb.hu

Bogdan Aman
baman@iit.tuiasi.ro

Péter Battyányi
battyanyi.peter@inf.unideb.hu

Gabriel Ciobanu
gabriel@info.uaic.ro

[1] Romanian Academy, Institute of Computer Science, Blvd. Carol I no.8, 700505 Iaşi, Romania

[2] Department of Computer Science, Faculty of Informatics, University of Debrecen, Kassai út 26, 4028 Debrecen, Hungary

transformation rules governing the work of the system are based on chemical and biological principles that can be observed inside biological cells, so the resulting distributed and parallel computational system can also be considered to be the realization of the chemical computing paradigm.

This close relation motivated the investigations in Battyányi and Vaszil (2014), where the chemical calculus was used to give a logical description of the computations performed by simple membrane systems. As both models rely on the chemical computing paradigm, using the chemical calculus for such a description is very natural.

In what follows, we continue these investigations by studying the possibility of extending the logical description to more complex types of membrane systems, in particular, to systems where membrane dissolution is also allowed during the computational process. Using a slightly modified variant of the operational semantics of membrane systems presented in Andrei et al. (2007), we show how to transform a membrane system with rules using promoters/inhibitors (see Bottoni et al. 2002), priorities, and also the possibility of membrane dissolution (introduced already in Păun 2000), into formulas of the chemical calculus, such that terminating computations of the membrane system correspond to terminating reduction sequences of formulas and vice versa.

2 Preliminaries

In this section we present the basic notions and notations we are going to use. For a comprehensive treatment of membrane systems ranging from the basic definitions to their computational power, see the monographs (Păun 2002; Păun et al. 2010), for more information on the chemical calculus, we refer to Banâtre et al. (2005, 2006).

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 V. We say that $M_1 \subseteq M_2$ if for all $a \in V$, $M_1(a) \leq M_2(a)$. The union or sum of two multisets over V is defined as $(M_1 + M_2)(a) = M_1(a) + M_2(a)$, the difference is defined for $M_2 \subseteq M_1$ as $(M_1 - M_2)(a) = M_1(a) - M_2(a)$ for all $a \in V$. 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 set of all finite multisets over an alphabet V is denoted by $\mathcal{M}(V)$, the empty multiset is denoted by $\emptyset$ as in the case of the empty set.

2.1 Membrane systems

A membrane system, or P system, has a structure represented by a set of regions (each delimited by a surrounding membrane) arranged in a tree (cell-like Păun 2000) or a graph form (tissue-like Martín-Vide et al. 2003 or neural-like Ionescu et al. 2006). In this paper we use transition P systems (Păun 2000) that have a cell-like structure with each membrane having a label and enclosing a region containing a multiset of objects and possibly other membranes. The unique out-most membrane is called the skin membrane. The membrane structure is denoted by a sequence of matching parentheses having the same labels as the membranes they represent. We assume the membranes are labelled by natural numbers $\{1, \ldots, n\}$, and we use the notation m_i for the membrane with label i. Each membrane m_i, except for the skin membrane, has its parent membrane, which we denote by $\mu(m_i)$. As an abuse of notation μ stands both for the membrane structure and for the function determining the parent membrane of a membrane. To facilitate the presentation we assume that $\mu(m_j) = m_i$ implies $i < j$.

The evolution of the contents of the regions of a P system is described by rules associated with the regions. The system performs a computation by passing from one configuration to another one, applying the rules synchronously in each region. In the variant we consider in this paper, the rules are multiset rewriting rules given in the form of $u \to v$ where u, v are multisets, and they are applied in a maximal parallel manner, that is, as many rules as possible are applied in each region. The end of the computation is defined by the following halting condition: 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 a membrane labelled as output.

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

$$\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \rho_1, \ldots, \rho_n)$$

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 with the regions; they are of the form $u \to v$ where $u \in \mathcal{M}(O)$ and $v \in \mathcal{M}((O \times tar) \cup \{\delta\})$ where $tar = \{here, out\} \cup \{in_j \mid 1 \leq j \leq n\}$ and $\delta \notin O$ is a special symbol,
- $\rho_1, \ldots, \rho_n$ are the priority rules associated with membranes $m_1, \ldots, m_n$.

The evolution rules of the system are applied in a nondeterministic, maximally parallel manner to the n-tuple of multisets of objects constituting the configuration of the system. A configuration is the sequence $C = (\mu_C, w_{k_1}, \ldots, w_{k_j})$ where $w_{k_i} \in O^*$, $1 \leq i \leq j$ represent the

contents of the membranes, and μ_C is the current membrane structure. Let $\mathcal{R} = R_1 \cup R_2 \cup \cdots \cup R_n$, where $R_i = \{r_{i1}, \ldots, r_{ik_i}\}$ is the set of rules corresponding to membrane m_i. 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 rule application in each region takes place in a non-deterministic and maximally parallel manner. This means that the rule application phase finishes, if no rule can be applied anymore in either region. As a result, each region where rule application took place, is possibly supplied with elements of the set $O \times tar$. We call a configuration which is a multiset over $O \cup O \times tar$ an intermediate configuration. If we want to emphasize that $C = (\mu, w_1, \ldots, w_n)$ consists of multisets over O, we say that C is a proper configuration. Rule applications can be preceded by priority check, if priority relations are present. Let $\rho_i \subseteq R_i \times R_i \, 1 \leq i \leq n$ be the (possibly empty) priority relations. Then $r \in R_i$ is applicable only if no $r' \in R_i$ can be applied with $(r', r) \in \rho_i$. We may also denote the relation $(r', r) \in \rho_i$ by $r' > r$.

In the next phase the objects coming from v should be added to the regions as specified by the target indicators associated with 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 $m_i = Skin$, the elements are sent to the environment and cannot be taken into consideration by us any longer. In our membrane systems we assume that the results are formed in a designated membrane of the system. 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 to the contents of region j.

The symbol δ marks a region for dissolution. When it is introduced in the membrane by a rule, after having finished the maximal parallel and communication steps, the actual membrane disappears. Its objects move to the parent membrane and its rules cannot be applied anymore.

To each rule $r = (u \to v) \in \mathcal{R}_i$ we can assign promoter/inhibitor sets, $prom/inhib$. The promoter/inhibitor sets assigned to r are subsets of O. When r is going to be applied they act as follows: r can be applied to the content w_i of membrane m_i only if every element of $prom$ is present in w and no element of $inhib$ can be found in w.

2.2 The chemical calculus

We give a brief summary of the chemical calculus following the presentation in Banâtre et al. (2005, (2006). Chemical programming is the formal equivalent of Gamma programming, which is a higher order multiset manipulating programming language. Like Gamma programming, the chemical calculus is also based on the chemical metaphor: data are represented by γ-terms, which are called molecules, and reactions between them are represented by rewrite rules. We begin with the basic definitions.

The syntactical elements of *molecules, reaction conditions*, and *patterns*, denoted by M, C and P, respectively, are defined as follows.

$$M := true \,|\, false \,|\, x \,|\, (M_1, M_2) \,|\, \langle M \rangle \,|\, \gamma(P)[C] \cdot M$$

where x is a variable standing for any molecule, *true* and *false* are two constants, (M_1, M_2) is a compound molecule built with the commutative and associative "," constructor operator, $\langle M \rangle$ is called a *solution*, and $\gamma(P)[C] \cdot M$ is called a γ-abstraction with pattern P, reaction condition C, result M. The solution $\langle M \rangle$ encapsulates the molecule M which is inside the solution, and thus, insulated from molecules outside the solution. The contents of solutions can only be changed by reactions which occur inside the solution.

The *γ-abstraction* encodes a rewriting rule: when the pattern P is respected and the condition C is met, a substituted variant of M is created as a result. A pattern is

$$P := x \,|\, (P_1, P_2) \,|\, \langle P \rangle,$$

where x matches any molecule, (P_1, P_2) matches a compound molecule, and $\langle P \rangle$ matches an *inert solution*, that is, a solution where no reaction in the outermost level can occur. (The contained solutions can still be active, however.)

Now we define how patterns are matched, which requires the notion of substitution. A *substitution* is a mapping ϕ from the set of variables to the set of molecules. We define the application of a substitution ϕ as follows:

$$\begin{aligned}
&\phi x = \phi(x) \\
&\phi(M_1, M_2) = \phi M_1, \phi M_2 \\
&\phi\langle M \rangle = \langle \phi M \rangle \\
&\phi(\gamma(P)[C] \cdot M) = \gamma(P)[C] \cdot \phi' M,
\end{aligned}$$

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

The result of a match is an assignment of molecules to variables. The first argument of *match* is a pattern, the second one is a molecule, its value is a substitution. If x denotes a variable, P a pattern, and M a molecule, then:

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

where $\oplus$ denotes the composition of substitutions provided $match(P_1, M_1)$ and $match(P_2, M_2)$ agree on their common variables. Otherwise, the result is undefined.

The reaction rule (γ-rule) is defined as

$$\gamma(P)[C] \cdot M, \quad N \rightarrow \phi M,$$

where $match(P, N) = \phi$ assigns values to variables in such a way that $\phi(C)$ reduces to *true*. The abbreviation $\gamma(P) \cdot M$ stands for the term $\gamma(P)[true] \cdot M$.

The commutativity and associativity of pairs are expressed by the following rules (AC-rules):

$$((M_1, M_2), M_3) \rightarrow (M_1, (M_2, M_3)),$$
$$(M_1, M_2) \rightarrow (M_2, M_1).$$

We can also define an operator *replace* (cf. Banâtre et al. 2006) which does not vanish in the course of the reduction:

$$replace\ P\ by\ M\ if\ C \rightleftharpoons let\ rec\ f = \gamma(P)[C] \cdot M, f\ in\ f$$

where *let rec* is an expression which uses the fixed point combinator for implementing recursion, and can be defined similarly as in the λ calculus (see, for example, Mitchell 1996 for more details).

Then the new operator obeys the following reduction rule:

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

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

We would like to remark here, that the authors of Banâtre et al. (2006) define both typed and untyped versions of the γ-calculus. Hitherto we have been using the untyped version of the calculus, although the typed version would make it simpler to talk about Boolean valued arithmetical relations. With some further efforts, we could identify a subset of the untyped γ-calculus (basically equal to the untyped λ-calculus) where truth values, natural numbers, primitive recursive functions, and arithmetical relations could be defined in the usual way. Instead of giving these definitions, we follow another approach, namely, we take the existence of truth values, expressions for logical connectives, and basic arithmetical functions as granted. Strictly speaking, this means, that we use the typed version of the calculus, but we do this only for notational convenience, we use types only as a shorthand to make the presentation easier to follow.

Example 1 Consider the γ-abstraction

$$(\gamma(x, y)[x = y] \cdot x,\ 1, 2, 3, 1, 4)$$

where (x, y) is a pattern, and $x = y$ is a reaction condition. In order for the condition to evaluate to *true*, the pattern needs to be matched as $\phi = match((x, y), (1, 1))$, thus, $\phi(x) = \phi(y) = 1$, and

$$(\gamma(x, y)[x = y] \cdot x,\ 1, 2, 3, 1, 4) \rightarrow (1, 2, 3, 4).$$

If we use the replace operator, we have

$$(replace\ (x, y)\ by\ x\ if\ (x = y),\ 1, 2, 3, 1, 4)$$
$$\rightarrow (replace\ (x, y)\ by\ x\ if\ (x = y),\ 1, 2, 3, 4).$$

3 Describing membrane system configurations and computations

As membrane systems are based on the chemical computational paradigm, it is natural to use the chemical calculus to describe membrane computations. A feature where the resemblance of the two models is manifest is, for example, the commutativity and associativity of the building of pairs (or the construction of multisets, more generally). Usually this kind of commutativity of the chemical calculus helps to make the representation of membrane computations more convenient (as in Battyányi and Vaszil 2014, for example), but in the present paper we take another, although less elegant approach.

The description of membrane system configurations that we are going to present in the following, might seem to contradict the "spirit" of the chemical calculus: it is basically a numerical description based on the multiplicities of the elements in the multisets (similar to the Parikh vector of strings), it does not make use of those features of the γ-calculus that make it appealing for the description of chemical computations in general, or membrane computations in particular. We chose to follow such a somehow "non-chemical-like" approach, because it seemed to be more economical for the handling of rule priorities, promoters/inhibitors, and membrane dissolution, those extended features used in membrane computations which we would like to be able to capture in our logical description. Thus, in some sense, we have abandoned the natural correspondence between membrane configurations and the abstract chemical solutions of the γ-calculus in the favor of including additional membrane system features in the logical descriptions (promoters/inhibitors, rule priorities, and membrane dissolution).

Our approach might also be justified by placing the results of the paper in the more general context of our investigatiosn on the relationship of membrane systems and chemical calculi (started in Battyányi and Vaszil 2014 and carried on in the present paper) which we would like to continue in order to show how variants of the γ-calculus can be used for the purpose of providing programming languages for different variants of membrane systems. For this purpose, it is important to work with chemical calculi, especially if we also intend to give a logical formalism which can be translated into membrane systems and the corresponding computations.

To introduce molecules for the description of membrane system configurations, we first need molecules being able to represent ordered sequences. Since we use only variables and basic values in keeping track of the computation

sequences of a P system, we can either choose the ordered pair construct $(A_1 : A_2)$ of the typed calculus, which is related to atoms of the calculus, or we can formulate our ordered pair. We choose the latter approach, since this construction equally works for the untyped calculus, as well.

Notation 1 *Let* $[x, y] = (\langle x \rangle, y)$, *and* $[x_1, \ldots, x_n, x_{n+1}] = [[x_1, \ldots, x_n], x_{n+1}]$.

Claim Let $P = [x_1, x_2, \ldots, x_k]$ be a pattern and $M = [n_1, n_2, \ldots, n_k]$, where $n_1, \ldots, n_k \in \mathbb{N}$. Then

$$match(P, N) = \{x_i \mapsto n_i\}_{i=1}^{k}.$$

If we use a, b as variables for elements of O and r as a rule variable, respectively, then we say that a rule $r = u \to v \in R_i$ is valid with respect to the proper configuration $(\mu, w_1, \ldots, w_n)$ if the following conditions hold:

1. membrane structure μ contains membrane m_i,
2. $(\forall a \in prom_r)\,(w_i(a) \geq 1)$,
3. $(\forall a \in inhib_r)\,(w_i(a) = 0)$, and
4. $(\forall a \in O)\,(\forall\ 1 \leq j \leq n)\ \ (v(a, in_j) \geq 1)$ implies that μ contains the membrane m_j (m_j is not dissolved) and $\mu(m_j) = m_i$, namely m_i is the parent membrane of m_j

where $prom_r \subseteq O$ and $inhib_r \subseteq O$ denotes the set of promoters and inhibitors associated with rule r, respectively. A rule r is valid in an intermediate configuration C', if there exists a proper configuration C such that $C \to^*_{mpr} C'' \to^*_{msg} C''' \to^*_{\delta} C'$ and r is valid in C. Moreover, if, for $C' = (\mu', w'_1, \ldots, w'_n)$ and $r = u \to v$, $u(a_j) \leq w'_i(a_j)$ holds as well for every element $a_j \in O$, then r is not only valid but also applicable.

Assume we have a set of priority rules $\rho_i \subseteq R_i \times R_i$ for each membrane m_i. Let C be a proper configuration. Then $r \in R_i$ is ρ-valid in C, if the above hold for the validity of r together with the stipulation that no r' such that $(r', r) \in \rho_i$ is applicable in C. A rule $r = u \to v$ is ρ-valid in an intermediate configuration C', if there exists a proper configuration C such that $C \to^*_{mpr} C'' \to^*_{msg} C''' \to^*_{\delta} C'$ and r is ρ-valid in C. Moreover, if we have $u(a_i) \leq w'(a_i)$, then r is called ρ-applicable or, simply, applicable. In the future, we do not intend to explicitly distinguish validity and ρ-validity in our terminology, if no confusion occurs.

A *description* of a membrane system configuration as above is a molecule of the form

$$\begin{aligned} Descr = [&c_{11}, \ldots, c_{1k}, \ldots, c_{n1}, \ldots, c_{nk},\\ &\overline{c}_{11}, \ldots, \overline{c}_{1k}, \ldots, \overline{c}_{n1}, \ldots, \overline{c}_{nk},\\ &d_1, \ldots, d_n,\\ &p_{11}, \ldots, p_{1k_1}, \ldots, p_{n1}, \ldots, p_{nk_n}], \end{aligned}$$

where c_{ij} and $\overline{c}_{ij}$ are natural numbers $(1 \leq i \leq n, 1 \leq j \leq k)$, $d_i \in \{0, 1\}$ $(1 \leq i \leq n)$ and $p_{ik_j} \in \{0, 1\}$ $(1 \leq i, j \leq n)$. If N is a description we denote by c_{ij}, $\overline{c}_{ij}$, etc. the respective parts of N. In what follows we explain the meaning of these parts of a description.

Let $C = (\mu, w_1, \ldots, w_n)$ be an (intermediate) configuration. A description corresponding to C is a description, where $c_{ij} = w_i(a_j)$ and $\overline{c}_{ij} = w_i(a_j, here) + \sum_{p \neq i, \mu(m_i) = m_p} w_p(a_j, in_i) + \sum_{\mu(m_p) = m_i} w_p(a_j, out)$ with $(1 \leq i, p \leq n)$ and $(1 \leq j \leq k)$. Here $\mu(m_p)$ denotes the parent membrane of m_p, and recall that $w(a)$ denotes the number of elements a in the multiset w. Intuitively, c_{ij} stands for the number of occurrences of a_j in m_i, and $\overline{c}_{ij}$ denotes the number of occurrences of a_j to be added in m_i according to the targeted elements of O. Moreover, $d_i = 1$ iff m_i is dissolved or under dissolution. The values $p_{ik_j} \in \{0, 1\}$ describe the validity of rules: a description for a proper configuration is called proper if, for every rule r_{ik_j}, the applicability and validity coincide, that is, r_{ik_j} is valid iff $p_{ik_j} = 1$ hold and every valid rule is applicable in the same time. Observe that if C is a proper configuration then $\overline{c}_{ij} = 0$ for every possible i and j. The construction will be such that when a configuration is proper, $d_i = 1$ will imply $w_i = 0$. Since p_{ik_j} describe validities which, in case of intermediate configurations, depend on proper configurations preceding them in the reduction sequence, the same intermediate configuration can have various descriptions depending on the rules we actually choose for the maximal parallel steps in advance. Hence, the descriptions corresponding to configurations C form a set which we denote by $Descr(C) - p_{ik_j}$ taking either 0 or 1 as values.

A pattern for a description is a tuple of the form

$$\begin{aligned} S = [&x_{m_1a_1}, \ldots, x_{m_1a_k}, \ldots, x_{m_na_1}, \ldots, x_{m_na_k},\\ &\overline{x}_{m_1a_1}, \ldots, \overline{x}_{m_1a_k}, \ldots, \overline{x}_{m_na_1}, \ldots, \overline{x}_{m_na_k},\\ &x_{d_1}, \ldots, x_{d_n},\\ &x_{r_{1k_1}}, \ldots, x_{r_{nk_n}}]. \end{aligned} \tag{1}$$

Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \rho_1, \ldots, \rho_n)$ be a P system, and let $C' = (\mu', w'_{k_1}, \ldots, w'_{k_j})$ be a proper configuration obtained from the initial configuration in a finite number of computational steps, where $1 \leq k_1 < \ldots < k_j \leq n$. Then a description of C' relative to μ is the description obtained from an element of $Descr(C')$ when we set $d_i = 1$ for $i \notin \{k_1, \ldots, k_j\}$ and $c_{ij} = 0\,(1 \leq j \leq k)$ and $p_{il} = 0$ for every rule $r_{il} \in R_i$. That is, we treat the missing membranes as empty membranes. We denote the set of descriptions of a configuration C' relative to μ by $Descr_{\mu}(C')$.

Example 2 Consider the P system $\Pi = (\{a, b, c, d\}, [\,[\,]_2\,]_1, \emptyset, abd, R_1, R_2, \rho_1, \rho_2)$ with

$$R_1 = \{r_{11} : c \to (d, here)\},$$
$$R_2 = \{r_{21} : ba \to (b, here)(c, here)(c, here)|_d > r_{22} : b \to \delta\},$$

where the sign $>$ indicates the priority between the rules of R_2, and $d \in O$ is a promoter: rule r_{21} can only be applied if there is a d object present in the second region (in our notation: $prom_{r_{21}} = \{d\}$).

The initial configuration is $C_0 = (\emptyset, abd, [\,[\,]_2\,]_1)$, which has a corresponding description

$$[0,0,0,0,\ 1,1,0,1,\ 0,0,0,0,\ 0,0,0,0,\ 0,0,\ 0,0,0] \quad \in \quad Descr(C_0),$$

if we assume an alphabetical ordering of the object alphabet.

Because a description should also contain information about the structure of the original P system itself, we append a representation of the function μ at the end of each description. Let Π be a P system of degree n as before. Then a tuple $[p_2, \ldots, p_n]$ of length $n-1$ is appended to every description in the simulation with the following meaning: if membrane m_j has membrane m_i as its parent, then $p_j = i$. Since the *Skin* has no parent membrane, numbering begins with 2. Likewise, a description pattern is expanded with the tuple $[x_{p_2}, \ldots, x_{p_n}]$. Since the structure of the original P system remains the same in the course of the simulation process, we do not indicate the appended values for μ, they are implicitly understood to be there.

With this in hand we are able to define the molecule in charge of deciding rule validity. Let $r = u \to v \in R_i$, and S be a description pattern. Then let

$$\begin{aligned} Val(r) = (x_{d_i} = 0 \wedge & \\ & \bigwedge_{1 \le j \le k} (a_j \in prom_r \supset x_{m_i a_j} \ge 1) \wedge \\ & \bigwedge_{1 \le j \le k} (a_j \in inhib_r \supset x_{m_i a_j} = 0) \wedge \\ & \bigwedge_{1 \le l \le k} \bigwedge_{1 \le j \le n} \left(v(a_l, in_j) \ge 1 \supset x_{d_j} = 0\right) \wedge \\ & \left(\bigvee_{l_0 = i > l_1 > \ldots > l_{s-1} > j = l_s} \left(\bigwedge_{1 \le t \le s} x_{p_{l_t}} = l_{t-1} \wedge \bigwedge_{1 \le q \le s-1} x_{d_{l_q}} = 1 \right) \right) \Bigg), \end{aligned} \tag{2}$$

where "$\supset$" denotes logical implication. The first three rows express the presence of promoters, the absence of inhibitors, and the availability of the objects required by the left hand side of the rule, respectively. The next row expresses the property that all target membranes exist, while the last row expresses the fact that either m_i is the parent of m_j, or m_i is an ancestor of m_j and all the intermediate parent membranes have been dissolved in the construction.

Now rule applicability can be expressed as

$$Val(r) \wedge \bigwedge_{1 \le j \le k} \left(u(a_j) \le x_{m_i, a_j}\right) \tag{3}$$

and the operator in charge for rule applicability is

$$RuleApp(r) = replace\ [S, 0]\ by\ [S[x_r/1], 0]\ if \left(Val(r) \wedge \bigwedge_{1 \le j \le k} (u(a_j) \le x_{m_i, a_j}) \right)$$

where the value 0 plays a role of synchronization to be specified later on. We remark that if a rule r is determined to be valid in this phase of the simulation, then r remains valid in the course of the simulation of a maximal parallel step.

We can also incorporate in the simulation of a membrane system the priority rules, if present. Let $(\rho_1, \ldots, \rho_n)$ be the tuple prescribing the priority relations in the membranes of a given P system. We define molecules determining the applicability of rules. For $r \in R_i$, we distinguish two cases:

- There does not exist $r' \in R_i$ such that $r' > r$. Then $RuleApp_\rho(r)$ is defined as $RuleApp(r)$ above.
- There are rules $r_1, \ldots, r_j \in R_i$ such that $r_l > r\ (1 \le l \le j)$. Let S be a description pattern and let $Val(r)$ defined in Eq. (2). Then

$$\begin{aligned} RuleApp_\rho(r) = (& replace\ [S, 0]\ by\ [S[x_r/1], 0] \\ & if \left(\left(Val(r) \wedge \bigwedge_{1 \le j \le k} (u(a_j) \le x_{m_i, a_j}) \right) \right) \\ & \wedge x_r = 0 \wedge \bigwedge_{1 \le l \le j} x_{r_l} = 0), \\ & replace\ [S, 0]\ by\ [S[x_r/0], 0] \\ & if \left(x_r = 1 \wedge \left(\bigvee_{1 \le l \le j} x_{r_l} = 1 \right) \right). \end{aligned}$$

Example 3 Considering the P system of the previous example, the applicabilty of the rules is checked as follows.

$$\begin{aligned} & (RuleApp_{\rho_1}(r_{11}), RuleApp_{\rho_2}(r_{21}), RuleApp_{\rho_2}(r_{22}), \\ & \quad [[0,0,0,0,\ 1,1,0,1,\ 0,0,0,0,\ 0,0,0,0,\ 0,0,\ 0,0,0],0]) \to \\ & (RuleApp_{\rho_1}(r_{11}), RuleApp_{\rho_2}(r_{21}), RuleApp_{\rho_2}(r_{22}), \\ & \quad [[0,0,0,0,\ 1,1,0,1,\ 0,0,0,0,\ 0,0,0,0,\ 0,0,\ 0,1,0],0]) \end{aligned}$$

Now we can turn to the main part of the simulation. Keep in mind that a rule is applicable if it is valid and there are enough elements in the membrane to be used by the left hand side of the rule.

Definition 1 Let $r = u \to v \in R_i$, and let S be a description pattern. Then the molecule describing the effect of an execution of r is defined as

$$App(r) = replace\,[S,1]\,by\,[\,apply(S,r),1]\,if \left(x_r = 1 \wedge \bigwedge_{1\le j\le k} \left(u(a_j) \le x_{m_i,a_j}\right)\right),$$

where

$$apply(S,r)(x_{m_s a_t}) = \begin{cases} x_{m_s a_t} - u(a_t) & \text{if s} = \text{i}, \\ x_{m_s a_t} & \text{otherwise}, \end{cases}$$

$$apply(S,r)(\overline{x}_{m_s a_t}) = \begin{cases} \overline{x}_{m_s a_t} + v(a_t, here) & \text{if s} = \text{i}, \\ \overline{x}_{m_s a_t} + v(a_t, in_j) & \text{if s} = \text{j} \neq \text{i}, \\ \overline{x}_{m_s a_t} + v(a_t, out) & \text{if } \text{m}_\text{s} = \mu(\text{m}_\text{i}), \end{cases}$$

$$apply(S,r)(x_{d_j}) = \begin{cases} 1 & \text{if v}(\delta) = 1, \\ x_{d_j} & \text{otherwise}, \end{cases}$$

$$apply(S,r)(x_r) = x_r.$$

Here we made use of the implicit stipulation that S is of the form as in Eq. (1), which is indeed the case if we ignore variable renaming.

Example 4 Considering the P system of our previous examples, we have:

$$\begin{aligned}&(App(r_{11}), App(r_{21}), App(r_{22}),\\ &\quad [[0,0,0,0,\ 1,1,0,1,\ 0,0,0,0,0,\ 0,0,0,0,0,\ 0,0,\ 0,1,0],1]) \rightarrow\\ &(App(r_{11}), App(r_{21}), App(r_{22}),\\ &\quad [[0,0,0,0,\ 0,0,0,1,\ 0,0,0,0,0,\ 0,1,2,0,\ 0,0,\ 0,1,0],1])\end{aligned}$$

The next group of rules is the set of communication rules. In what follows, we define the chemical calculus equivalents of communication steps.

Definition 2

$$Msg = replace\,[S,2]\,by\,[msg(S),2]\,if \left(\bigvee_{1\le i\le n}\ \bigvee_{1\le j\le k} \overline{x}_{m_i a_j} \ge 1\right),$$

where

$$msg(S)(x_{m_i a_j}) = x_{m_i a_j} + \overline{x}_{m_i a_j}, \text{ for } 1 \le i \le n \text{ and } 1 \le j \le k$$

and

$$msg(S)(\overline{x}_{m_i a_j}) = 0, \quad \text{for } 1 \le i \le n \quad \text{and} \quad 1 \le j \le k.$$

Continuing our example, we have the following.

Example 5 Considering the P system of our previous examples, we have:

$$\begin{aligned}&(Msg, [[0,0,0,0,\ 0,0,0,1,\ 0,0,0,0,0,\ 0,1,2,0,\ 0,0,\ 0,1,0],2])\\ &\quad \rightarrow (Msg, [[0,0,0,0,\ 0,1,2,1,\ 0,0,0,0,0,\ 0,0,0,0,0,\\ &\quad 0,0,\ 0,1,0],2])\end{aligned}$$

At this point, we simulate the effects of membrane dissolving. We have to make sure that only membranes not dissolved and not under dissolution contain elements. To this end, we define the next molecule, which moves every element being in a membrane dissolved or under dissolution to its parent membrane.

Definition 3

$$Dis_i = replace\,[S,3]\,by\,[dis_i(S),3]\,if \left(x_{d_i} = 1 \wedge \left(\bigvee_{1\le j\le k} x_{m_i a_j} \ge 1\right)\right),$$

where

$$dis_i(S)(x_{m_j a_l}) = \begin{cases} x_{m_j a_l} + x_{m_i a_l} & \text{if } m_j = \mu(m_i), \\ 0 & \text{if } j = i, \\ x_{m_j a_l} & \text{otherwise.} \end{cases}$$

We also need some auxiliary molecules to set the values indicating the applicability of rules to zero, in order to start a new maximal parallel step. Thus:

Definition 4

$$RemRule(r) = replace\,[S,4]\,by\,[S[x_r/0],4]\,if\,x_r = 1.$$

Example 6 Considering the P system of our previous examples, we have:

$$\begin{aligned}&(RemRule(r_{11}), RemRule(r_{21}), RemRule(r_{22}),\\ &\quad [[0,0,0,0,\ 0,1,2,1,\ 0,0,0,0,0,\ 0,0,0,0,0,\ 0,0,\ 0,1,0],4]) \rightarrow\\ &(RemRule(r_{11}), RemRule(r_{21}), RemRule(r_{22}),\\ &\quad [[0,0,0,0,\ 0,1,2,1,\ 0,0,0,0,0,\ 0,0,0,0,0,\ 0,0,\ 0,0,0],4]).\end{aligned}$$

After removing the rule applicabilities, the simulation of another rewriting step can begin. The configuration corresponding to the above description is $(\emptyset, bccd, [\,[\,]_2\,]_1)$. Now rule r_{22} becomes applicable, leading to the dissolution of the second membrane.

$$\begin{aligned}&(RuleApp_{\rho_1}(r_{11}), RuleApp_{\rho_2}(r_{21}), RuleApp_{\rho_2}(r_{22}),\\ &\quad [[0,0,0,0,\ 0,1,2,1,\ 0,0,0,0,0,\ 0,0,0,0,0,\ 0,0,\ 0,0,0],0]) \rightarrow\\ &(RuleApp_{\rho_1}(r_{11}), RuleApp_{\rho_2}(r_{21}), RuleApp_{\rho_2}(r_{22}),\\ &\quad [[0,0,0,0,\ 0,1,2,1,\ 0,0,0,0,0,\ 0,0,0,0,0,\ 0,0,\ 0,0,1],0])\end{aligned}$$

Now rule r_{22} is applied as

$$\begin{aligned}&(App(r_{11}), App(r_{21}), App(r_{22}),\\ &\quad [[0,0,0,0,\ 0,1,2,1,\ 0,0,0,0,0,\ 0,0,0,0,0,\ 0,0,\ 0,0,1],1]) \rightarrow\\ &(App(r_{11}),\ App(r_{21}), App(r_{22}),\\ &\quad [[0,0,0,0,\ 0,0,2,1,\ 0,0,0,0,0,\ 0,0,0,0,0,\ 0,1,\ 0,0,1],1]),\end{aligned}$$

changing the coordinate indicating the dissolution of the second membrane from 0 to 1.

Since there are no objects on the right side of the applied rule, the Msg molecule has no effect on this description, so we continue with the simulation of the dissolution of the second membrane (see Definition 3).

$$(Dis_1, Dis_2, [[0,0,0,0,\ 0,0,2,1,\ 0,0,0,0,\ 0,0,0,0,\ 0,1,\ 0,0,1],3]) \rightarrow$$
$$(Dis_1, Dis_2, [[0,0,2,1,\ 0,0,0,0,\ 0,0,0,0,\ 0,0,0,0,\ 0,1,\ 0,0,1],3]).$$

Similarly as above, $RemRule(r)$ removes the indicators for the applicability of rules, leaving us with the description

$$[0,0,2,1,\ 0,0,0,0,\ 0,0,0,0,\ 0,0,0,0,\ 0,1,\ 0,0,0].$$

The whole translation is coordinated by the *Sync* molecule defined below. As shorthand, we introduce the notation $\cup(M_i \mid i \in I) = (M_{i_1}, \ldots, M_{i_k})$, where $I = \{i_1, \ldots, i_k\}$.

$$RuleApp_\rho = \bigcup(RuleApp_\rho(r) \mid r \in \mathcal{R}),$$
$$App = \bigcup(App(r) \mid r \in \mathcal{R}),$$
$$Dis = \bigcup(Dis_i \mid i \in \{1,\ldots,n\}),$$
$$RemRule = \bigcup(RemRule(r) \mid r \in \mathcal{R}),$$

$$Sync = replace\ \langle [S, x_{sync}], Val_\rho, App, Msg, Dis, RemRule\rangle\ by$$
$$\langle [S, x_{sync} + 1\ mod\ 5], Val_\rho, App, Msg, Dis, RemRule\rangle\ if$$
$$\bigvee_{1 \le i \le n} x_{r_i} = 1,$$

where S is a description pattern.

Notation 2 *Let N be a molecule and let*

$$M(N) = (\langle N, RuleApp_\rho, App, Msg, Dis, RemRule\rangle, Sync).$$

As an abuse of notation, we denote by $M(C, i)$ the term $M([D, i])$ with a certain $D \in Descr_\mu(C)$, where the exact value of D is not interesting for us. Thus a relation like $M(C_1, i) \rightarrow M(C_2, j)$ should be interpreted as there exists $D_i \in Descr_\mu(C_i)$ $(1 \le i \le 2)$ such that $M([D_1, i]) \rightarrow M([D_2, j])$. Moreover, let τ be a transformation on descriptions, that is, a function assigning a description to a description. Then, if $M(C, i) = M([D, i])$ for some $D \in Descr_\mu(C)$, then $M(\tau(C), i)$ denotes the term $M([\tau(D), i])$.

The terms of the chemical calculus, and also the configurations of membrane systems can be considered as rewriting systems. A rewriting system, as used in this paper, 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. It is supposed that a reduction relation $\rightarrow_i$ is compatible with the term formation rules. Moreover, if $\rightarrow_i$ is a reduction relation, we denote by $\rightarrow_i^*$ its reflexive, transitive closure. We may use the notation $\rightarrow = \cup_{i \in I}(\rightarrow_i)$, too. In the following, the set Σ is the set of configurations of a P system or, in the case of the chemical formalism, the set of γ-terms, and $\rightarrow_i$ are the binary relations rendering configurations to configurations or terms to terms, respectively. We say that $m \in \Sigma$ is in *normal form*, if there is no $n \in \Sigma$, such that $m \rightarrow n$. Moreover, an $m \in \Sigma$ is *strongly normalizable*, if every reduction sequence starting from m is finite, or *weakly normalizable*, if there exists a finite reduction sequence starting from m. We say that a molecule or a membrane M is $\rightarrow_i$-irreducible, if there is no M' such that $M \rightarrow_i M'$. In what follows, to conform to the usual membrane system notation, we use $\Rightarrow$ to denote $\rightarrow$ when we speak of a rewriting step in a membrane computation.

Theorem 1

1. *Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \rho_1, \ldots, \rho_n)$ be a P system of degree n with membrane dissolution, promoter/inhibitor sets for rules and priority relations. Assume*

   $$C_0 = (\mu, w_1, \ldots, w_n) \Rightarrow^* C_1 = \left(\mu', w'_{n_1}, \ldots, w'_{n_i}\right),$$

   where $1 \le n_1 \le \ldots \le n_i \le n$. Then

   $$M(C_0, 0) \rightarrow^* M(C_1, 0).$$

   If the computation starting from C_0 contains at least one step, then the reduction sequence starting from $M(C_0, 0)$ is also non-empty.
2. *Let Π be a P system as above. Assume*

   $$M(C_0, 0) \rightarrow^* M([N, 0]), \quad \text{and}$$

   assume that $\overline{c}_{ij} = 0$ for $(1 \le i \le n)$ and $(1 \le j \le k)$ in N and $[N, 0]$ is $RuleApp_\rho$ irreducible. Then there exists a configuration $C_1 = (\mu', w'_{n_1}, \ldots, w'_{n_i})$ of Π such that $M([N, 0]) = M(C_1, 0)$ and

   $$C_0 \Rightarrow^* C_1.$$

Moreover, if the length of $M(C_0, 0) \rightarrow^* M([N, 0])$ is at least one, then the length of the computation starting from C_0 is non-zero.

In order to prove the theorem we need to state several auxiliary lemmas.

As formulated in Andrei et al. (2006), a computational step starting from a configuration C_0 of Π consists of a maximal parallel step (mpr), a step for removing the directions from the targeted elements (tar) and a step for accomplishing membrane dissolution (δ). In notation, if C_0

is a configuration of Π and $C_0 \Rightarrow C_1$, then there are C_0' and, if δ is present, C_0'' such that

$$C_0 \Rightarrow^*_{mpr} C_0' \Rightarrow_{tar} C_0'' \Rightarrow_\delta C_1.$$

In the present paper, instead of $\Rightarrow_{tar}$, we choose a sequential relation (msg) defined in Definition 5 for removing messages instead of parallel communication rules, which equally suffices for our purposes. In what follows, if $C \Rightarrow_s C'$ by an intermediate step, we denote by $s \in mpr$ $(s \in msg, s \in \delta)$ the fact whether s is a maximal parallel, message removing, or membrane dissolving step, respectively.

We verify the lemmas simultaneously by induction on the number of intermediate steps in a computational step of the P system and on the number of reductions in the chemical calculus.

Notation 3 *Let C' be an (intermediate) configuration, where $C \Rightarrow^* C'$. Let $D' \in Descr_\mu(C')$ be a description of C' relative to μ. Then we use the notation below to extract the corresponding values from $M(C', l)$:*

$$\begin{aligned}
&\lfloor M(C',l)\rfloor_{c_{ij}} = D'_{i\cdot k+j} && (0 \le i \le n-1,\ 1 \le j \le k),\\
&\lfloor M(C',l)\rfloor_{\overline{c}_{ij}} = D'_{(n+i)\cdot k+j} && (1 \le i \le n,\ 1 \le j \le k),\\
&\lfloor M(C',l)\rfloor_{d_i} = D'_{2n\cdot k+i} && (1 \le i \le n),\\
&\lfloor M(C',l)\rfloor_{r_{ij}} = D'_{(2k+1)\cdot n+k_1+\ldots+k_{i-1}+j} && (1 \le j \le k_i,\ 1 \le i \le n).
\end{aligned}$$

The next claims can be easily verified. Below, let D and D' denote descriptions.

Claim Let $M([D,0]) \rightarrow^*_{RuleApp} M'$. Then $M' = M([D',0])$.

Claim Let $M([D,1]) \rightarrow^*_{App} M'$. Then $M' = M([D',1])$.

Claim Let $M([D,2]) \rightarrow^*_{Msg} M'$. Then $M' = M([D',2])$.

Claim Let $M([D,3]) \rightarrow^*_{Dis} M'$. Then $M' = M([D',3])$.

In the following, we assume that every possible configuration is the result of some computational sequence starting from a fixed configuration C of the P system $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \rho_1, \ldots, \rho_n)$ of degree n with membrane dissolution, promoter/inhibitor sets for rules, and priority relations.

We prove the two parts of the theorem by simultaneous induction on the number of reduction steps in the chemical calculus and computational steps in the P system, respectively.

Lemma 1

1. *Let $C' \Rightarrow^*_{mpr} C''$. Then $M(C',1) \rightarrow^*_{App} M(C'',1)$, and conversely,*
2. *if we assume $M(C',1) \rightarrow^*_{App} M''$, then there is C'' such that $C' \Rightarrow^*_{mpr} C''$ and $M'' = M(C'',1)$.*

Proof We prove the lemma by simultaneous induction on the lengths of the reduction sequences. Assume we know the result for reduction sequences of lengths at most s.

1. Let $C \Rightarrow^*_{mpr} C'$, with $C' = (\mu', w_1', \ldots, w_n')$. Assume $C' \Rightarrow^s_{mpr} C''' \Rightarrow_r C''$, $C'' = (\mu'', w_1'', \ldots, w_n'')$, $C''' = (\mu''', w_1''', \ldots, w_n''')$ and $r = u \rightarrow v \in \mathcal{R}_i$. Since r is applicable to C''', we have $\lfloor M(C''',1)\rfloor_r = 1$ and $u(a_j) \le w_i(a_j)$, where $M(C''',1)$ stand for $M([D''',1])$ with some $D''' \in Descr_\mu(C''')$. This means $u(a_j) \le \lfloor M(C''',1)\rfloor_{c_{ij}}$. These together imply that $App(r)$ can be applied to $M(C''',1)$ yielding $M([apply(D''',r),1])$ with $D''' \in Descr_\mu(C''')$.
   - Let $\lfloor M([apply(D''',r),1])\rfloor_{c_{lj}} = s_{lj}$. Then $s_{lj} = \lfloor M(C''',1)\rfloor_{c_{lj}} - u(a_j) = w_i'''(a_j) - u(a_j)$, if $l = i$, and $s_{lj} = \lfloor M(C''',1)\rfloor_{c_{lj}} = w_i'''(a_j)$ otherwise.
   - Let $\lfloor M([apply(D''',r),1])\rfloor_{\overline{c}_{lj}} = t_{lj}$. Then $t_{lj} = \lfloor M(C''',1)\rfloor_{\overline{c}_{lj}} + v(a_j, here)$, if $l = i$, $t_{lj} = \lfloor M(C''',1)\rfloor_{\overline{c}_{lj}} + v(a_j, in_h)$, if $l = h \ne i$ and $\mu'''(m_h) = m_i$, and $t_{lj} = \lfloor M(C''',1)\rfloor_{\overline{c}_{lj}} + v(a_j, out)$, if $l = \mu'''(m_i)$. Taking these into account, $t_{lj} = w_l''(a_j, here) + \sum_{p \ne l, \mu''(m_l) = m_p} w_p''(a_j, in_l + \sum_{\mu''(m_p) = m_l} w_p''(a_j, out)$ remains valid. Here, we made use of the fact that the membrane structure remains unchanged during a maximal parallel step, that is, $\mu' = \mu'' = \mu'''$.
   - If $v(\delta) = 1$, then $\lfloor M(C'',i)\rfloor_{d_i}$ is set to 1.
2. Let $M(C',1) \rightarrow^*_{App} M''$, assume $D' \in Descr_\mu(C')$ is the underlying description. It is enough to prove the result for the case $M(C',1) \rightarrow_{App(r)} M''$, where $r = u \rightarrow v \in \mathcal{R}_i$ for some i. By Claim 3, $M'' = M([D'',1])$. Since $App(r)$ is applicable to $M(C',1)$, we have $\lfloor M(C',1)\rfloor_r = 1$. It is easy to check that this implies the validity of r with respect to C'. Moreover, $u(a_j) \le \lfloor M(C',1)\rfloor_{c_{ij}} = w_i'(a_j)$, for every $1 \le j \le k$, which makes r applicable to C' yielding some intermediate configuration C''. From this point on, we can show by a reasoning similar to that of the previous point that D'' is a description for C'', where $D'' = (apply(S,r))\Phi$, where $\Phi = match(S, D')$. We omit the details. □

Instead of parallel communication as defined in Andrei et al. (2006) we choose a simpler way which is equally suitable to our present purposes and we define $\Rightarrow_{msg}$ as the following set of sequential multiset transformations.

Definition 5 Let $C = (\mu, w_1, \ldots, w_n)$ and $C' = (\mu', w_1', \ldots, w_n')$.

Then $C \Rightarrow^*_{msg} C'$ holds iff one of the following cases is valid.

1. Assume that $w_i(a_j, here) > 0$. Then $w'_i(a_j) = w_i(a_j) + w_i(a_j, here)$ and $w'_i(a_j, here) = 0$. All the other values remain unchanged.
2. Assume $w_i(a_j, in_l) > 0$. Then $w'_l(a_j) = w_l(a_j) + w_i(a_j, in_l)$ and $w'_i(a_j, in_l) = 0$. All the other values remain unchanged.
3. Assume $w_i(a_j, out) > 0$ and $m_l = \mu(m_i)$ is defined. Then $w'_l(a_j) = w_l(a_j) + w_i(a_j, out)$ and $w'_i(a_j, out) = 0$. If $i = Skin$, then $w'_i(a_j, out) = 0$. All the other values remain unchanged.

Lemma 2

1. *Let $C' \Rightarrow^*_{msg} C''$, and assume that C'' is msg-irreducible. Then $M(C', 2) \rightarrow^*_{Msg} M(C'', 2)$.*
2. *Conversely, assume $M(C', 2) \rightarrow^*_{Msg} M''$. Then there is C'' such that $C' \Rightarrow^*_{msg} C''$, and $M'' = M(C'', 1)$.*

Moreover, in both cases, if one of the transition sequences is not empty, then its corresponding transition sequence contains at least one step, also.

Proof

1. We prove by induction on the number of steps in $C \Rightarrow^*_{msg} C'$ that, if $D \in Descr_\mu(C)$ and $D' \in Descr_\mu(C')$, then, for every $1 \le i \le n$ and $1 \le j \le k$,

$$D_{c_{ij}} + D_{\overline{c}_{ij}} = D'_{c_{ij}} + D'_{\overline{c}_{ij}}. \tag{4}$$

To this end, we show that, if $C \Rightarrow_{msg} C'$ and $C = (\mu, w_1, \ldots, w_n)$ and $C = (\mu', w'_1, \ldots, w'_n)$, then

$$\begin{aligned} &w_i(a_j) + w_i(a_j, here) + \sum_{p \neq i} w_p(a_j, in_i) + \sum_{\mu(m_p) = m_i} w_p(a_j, out) \\ &\quad = w'_i(a_j) + w'_i(a_j, here) + \sum_{p \neq i} w'_p(a_j, in_i) + \sum_{\mu'(m_p) = m_i} w'_p(a_j, out). \end{aligned} \tag{5}$$

First, observe that $\mu = \mu'$, since a *msg* transition does not change the underlying membrane structure. We treat Point 2 of Definition 5, the remaining cases can be handled similarly. Let $C \Rightarrow_{msg} C'$ by Point 2 of Definition 5. Assume $w_i(a_j, in_l) > 0$. Let us consider only the case $i = l$ in Eq. 5, since for all the other cases the equation trivially holds. But in this case the left hand side contains $w_l(a_j) + w_i(a_j, in_l)$, and the right hand side contains the corresponding $w'_l(a_j) + w'_i(a_j, in_l)$, which, by definition, are equal.

Let $C \Rightarrow^*_{msg} C'$, assume that C' is msg-irreducible. A msg-irreducible P system with the *Skin* membrane as the outermost membrane contains no messages, thus, by Eq. 4, $M(C, 2) \rightarrow^*_{Msg} M(C', 2)$.

2. Let $M(C, 2) \rightarrow^*_{Msg} N'$, we may assume $M(C, 2) \rightarrow_{Msg} N'$. Then, by Claim 3, $N' = M([D', 2])$ for some description D'. We can prove by induction on the number of summands constituting the defining expression of $\overline{c}_{ij}$ that there exists C' such that $C \Rightarrow^*_{msg} C'$ and $D' \in Descr_\mu(C')$. □

Now, following Agrigoroaiei and Ciobanu (2011), we define the skeleton of a configuration $(\mu, u_1, \ldots, u_n)$ as $U' = (u'_1, \ldots, u'_n)$, where $u'_i = *$, if membrane i is dissolved or under dissolution (that is, $u_i(\delta) = 1$ and $i \neq Skin$) and $u'_i = 0$ otherwise. Let

$$\begin{aligned} &\mu^0(m_i) = m_i, \\ &\mu^j(m_i) = \mu(\mu^{j-1}(m_i)) \quad \text{for } j > 0. \end{aligned}$$

Let $\mu_{U'}(m_i) = m_j$, where $j = min\{p \mid \mu^k(m_i) = m_p \wedge u'_p \neq * \wedge u'(\mu^l(m_i)) = * \text{ for } 0 \le l \le k-1\}$. That is, $\mu_{U'}(m_i)$ is the smallest membrane containing membrane m_i which exists or does not disappear. Now we are in a position to define the δ reduction on configurations.

Definition 6 Let $C' \Rightarrow_\delta C''$, assume $w'_l(\delta) = 1$ for at least one membrane m_l. We define the effect of the dissolution rule as follows: $(\mu', w'_1, \ldots, w'_n) \Rightarrow_\delta (\mu'', w''_1, \ldots, w''_n)$, where $w''_i = *$ provided $u'_i = *$, and $w''_i(a_j) = w'_i(a_j) + \sum\{w'_l(a_j) \mid \mu_{U'}(m_l) = m_i, w'_l(\delta) = 1\}$, if $u'(i) = 0$.

Lemma 3

1. *If $C' \Rightarrow_\delta C''$, then $M(C', 3) \rightarrow^*_{Dis} M(C'', 3)$ and $M(C'', 3)$ is Dis-irreducible.*
2. *Conversely, assume that $M(C', 3) \rightarrow^*_{Dis} M''$, and M'' is Dis-irreducible. Then there exists C'' with $C' \Rightarrow_\delta C''$ and $M'' = M(C'', 3)$.*

Proof

1. Let $C' = (\mu, w'_1, \ldots, w'_n) \Rightarrow_\delta C''$, and assume that $w'_i(\delta) = 1$. Then $\lfloor M(C', 3) \rfloor_{d_i} = 1$. Let $\lfloor M(C', 3) \rfloor_{c_{ij}} > 1$ for some $1 \le j \le k$. Then $M(C', 3) \rightarrow_{Dis_i} M([dis_i(C'), 3])$. Let $p = \mu_{U'}(m_i)$, where U' is the skeleton of C'. Let $D' = Descr_\mu(C')$ and $D'' = dis_i(Descr_\mu(C'))$. Let us denote by $D'_{c_{ij}}$ and $D''_{c_{ij}}$ the values of the descriptions pertaining to the coordinates (i, j). It follows immediately, by Definition 3, that

$$D'_{c_{pj}} + \sum \left\{ D'_{c_{lj}} \mid \mu_{U'}(l) = p, D'_{d_l} = 1 \right\} = D''_{c_{pj}} + \sum \left\{ D''_{c_{lj}} \mid \mu_{U'}(l) = p, D''_{d_l} = 1 \right\}. \quad (6)$$

In other words, for every $1 \leq j \leq k$, the sums of the occurrences of elements a_j in the dissolved or to be dissolved descendants of membrane m_p plus the multiplicity of a_j in m_p remain the same at a dissolution step in the chemical calculus. Let $M(C', 3) \rightarrow_{Dis} M([D, 3])$ such that $M([D, 3])$ is irreducible with respect to *Dis*. Then $D_{d_i} = 1$ implies $D_{c_{ij}} = 0$, which, by Eq. 6, involves that $D \in Descr_\mu(C'')$.

2. Let $M(C', 3) \rightarrow^*_{Dis} M''$. By Claim 3, there exists a description D'' such that $M'' = M([D'', 3])$. Moreover, $M(C', 3)$ being *Dis* reducible involves that there exists $1 \leq l \leq n$ such that $w'_l(\delta) > 0$. Let $C' \Rightarrow^*_\delta C''$. Since M'' is *Dis* irreducible, $D''_{d_i} = 1$ implies $D''_{c_{ij}} = 0$. Then Eq. 6 simplifies to

$$D'_{c_{pj}} + \sum \left\{ D'_{c_{lj}} \mid \mu_{U'}(l) = p, D'_{d_l} = 1 \right\} = D''_{c_{pj}},$$

when $D''_{d_p} = 0$, and $D''_{c_{pj}} = 0$ otherwise. Then, by Definition 6, this amounts to $D'' \in Descr_\mu(C'')$. □

Now we make sure that the process can be iterated after simulating one maximal parallel step in the membrane computation.

Lemma 4 *Let D' be a description, with $D'_{c_{ij}} = 0$ and $D'_{r_{jk_j}} = 0\,(1 \leq i \leq n, \quad 1 \leq j \leq k)$ and, for every $1 \leq i \leq n, 1 \leq j \leq k$, $D'_{d_i} = 1$ implies $D'_{c_{ij}} = 0$. Let $C' = (\mu', w'_1, \ldots, w'_n)$ be the configuration such that $D' \in Descr_\mu(C')$. Assume $M([D', 0]) \rightarrow^*_{RuleApp_\rho} M([D'', 0])$. Then $M([D'', 0])$ is $RuleApp_\rho$-irreducible iff $D'' \in Descr_\mu(C')$ is proper.*

Proof Let D', D'' and C' be as above. Let $r = u \rightarrow v \in R_i$. Let us introduce $Val^+_\rho(r) = Val_\rho(r) \wedge \bigwedge_{1 \leq j \leq k} u(a_j) \leq w_i(a_j)$. In what follows we omit the subscripts ρ. Making use of the correspondence between configurations and descriptions we have, for every rule r, r is ρ-applicable iff $Val^+(r)$ and, for every $r' > r$, $\neg Val^+(r')$. Thus we have to show $M([D'', 0])$ is *RuleApp* irreducible iff $D''_r = 1$ is equivalent to $Val^+(r)$ and, for every $r' > r$, $\neg Val^+(r')$. Let us denote this last equivalence by $(*)$.

1. Let $M([D', 0]) \rightarrow^s_{RuleApp} M([D'', 0])$. Assume $M([D'', 0])$ is *RuleApp*-irreducible. By induction on s we have

$$D''_r = 1 \text{ implies } Val^+(r). \quad (7)$$

Suppose $Val^+(r)$ and, for every $r' > r$, $\neg Val^+(r')$ holds and $D''_r = 0$. By Eq. 7 we can conclude, for every $r' > r$, $D''_{r'} = 0$. But this would make $RuleApp(r)$ applicable to D'', which contradicts the *RuleApp* irreducibility of D''. For the other direction, assume $D''_r = 1$ and, furthermore, there exists $r' > r$ such that $Val(r')$. We may suppose r' is maximal with respect to priority. Then, for every $r'' > r'$, $\neg Val(r'')$. By the above reasoning, we have $D''_{r'} = 1$, which would imply the reducibility of D'' with respect to $RuleApp(r)$, which is not possible.

2. Assume $M([D'', 0])$ is $RuleApp(r)$-reducible and $(*)$ is true. Let $Val^+(r)$, $D''_r = 0$ and, for every $r' > r$, $D''_{r'} = 0$. Let $r' > r$ be maximal such that $Val^+(r')$. By $(*)$ we have $D''_{r'} = 1$, a contradiction. Assume furthermore that $D''_r = 1$ and $D''_{r'} = 1$ for some r and $r' > r$. Then, by $(*)$, $(\forall r^* > r)\neg Val^+(r^*)$. But 7 implies $(\forall r^* > r)(D''_{r^*} = 0)$, in particular, $D''_{r'} = 0$, which is not possible. □

Proof of Theorem 1.

($\Rightarrow$) Let $C_0 \Rightarrow^t C_1$. Assume $M(C_0, 1)$ is such that the description of C_0 in $M(C_0, 0)$ is proper. We obtain the result, by induction on t, making use of Lemmas 1, 2, 3 and 4 together with the observation that $M(D, i) \rightarrow_{Sync} M(D, i + 1 \, mod \, 5)$ whenever $M(D, i)$ is irreducible for a reduction corresponding to a membrane computation step.

($\Leftarrow$) Follows in a way similar to the above reasoning, this time making use of the other directions of the lemmas.

Corollary 1 *Let $\Pi = (O, \mu, w_1, \ldots, w_n, R_1, \ldots, R_n, \rho_1, \ldots, \rho_n)$ and let $C = (\mu, w_1, \ldots, w_n)$. Then Π is strongly (resp. weakly) normalizing iff $M(C, 0)$ is strongly (resp. weakly) normalizing. Moreover, the halting computations starting from C provide the same results as those supplied by the terminating reduction sequences of $M(C, 0)$.*

4 Conclusion

We have shown how to transform computations of membrane systems with promoters/inhibitors, priorities, and membrane dissolution into reduction sequences of certain molecules of the chemical calculus. The transformation is "faithful" in the sense that terminating membrane computations correspond to terminating reduction sequences of the chemical calculus, and the same results are obtained in both computational models. This connection will allow to use the tools and techniques developed for chemical calculi to reason about membrane systems and their computations.

Acknowledgments The work of the authors from Iasi was supported by the Romanian National Authority for Scientific Research, project number PN-II-ID-PCE-2011-3-0919.

References

Agrigoroaiei O, Ciobanu G (2011) Flattening the transition P systems with dissolution. In: Gheorghe M, Hinze T, Păun G, Rozenberg G, Salomaa A (eds) Membrane Computing, 11th International Conference, CMC 2010, Jena, Germany, 2010. Revised Selected Papers. Volume 6501 of Lecture Notes in Computer Science. Springer, Berlin, pp 53–64

Andrei O, Ciobanu G, Lucanu D (2006) Structural operational semantics of P systems. In: Freund R, Păun G, Rozenberg G, Salomaa A (eds) Membrane Computing, 6th International Workshop, WMC 2005, Vienna, Austria. Revised Selected and Invited Papers. Volume 3850 of Lecture Notes in Computer Science. Springer, Berlin, pp 32–49

Andrei O, Ciobanu G, Lucanu D (2007) A rewriting logic framework for operational semantics of membrane systems. Theor Comput Sci 373:163–181

Banâtre JP, Fradet P, Radenac Y (2005) Principles of chemical programming. Electron Notes Theor Comput Sci 124(1): 133–147

Banâtre JP, Fradet P, Radenac Y (2006) Generalized multisets for chemical programming. Math Struct Comput Sci 16:557–580

Banâtre JP, Le Métayer D (1986) A new computational model and its discipline of programming. Technical Report RR0566, INRIA

Battyányi P, Vaszil Gy (2014) Describing membrane computations with a chemical calculus. Fundam Inform 134:39–50

Bottoni P, Martín-Vide C, Păun G, Rozenberg G (2002) Membrane systems with promoters/inhibitors. Acta Inform 38:695–720

Ionescu M, Păun G, Yokomori T (2006) Spiking neural P systems. Fundam Inform 71:279–308

Martín-Vide C, Păun G, Pazos J, Rodríguez-Patón A (2003) Tissue P Systems. Theor Comput Sci 296:295–326

Mitchell JC (1996) Foundations of Programming Languages. MIT Press, Cambridge

Păun G (2000) Computing with membranes. J Comput Syst Sci 61(1):108–143

Păun G (2002) Membrane Computing. An Introduction. Springer, Berlin

Păun G, Rozenberg G, Salomaa A (eds) (2010) The Oxford Handbook of Membrane Computing. Oxford University Press, Oxford