Available online at www.sciencedirect.com
ScienceDirect
Theoretical Computer Science 371 (2007) 83–105
Theoretical Computer Science
www.elsevier.com/locate/tcs

Universality results for P systems based on brane calculi operations

Shankara Narayanan Krishna

Department of Computer Science and Engineering, Indian Institute of Technology, Bombay, Powai, Mumbai 400 076, India

Abstract

Operations with membranes are essential both in brane calculi as well as in membrane computing. In this paper, we attempt to express six basic operations of brane calculi, viz., *pino, exo, phago, bud, mate, drip* in terms of the membrane computing formalism. We also investigate the computing power of P systems controlled by phago/exo, pino/exo, bud/mate as well as the mate/drip operations. We give an improvement to a characterization of RE using mate/drip operations given in [L. Cardelli, Gh. Paun, An universality result based on mate/drip operations, International Journal of Foundations of Computer Science (in press)]. We also give a characterization of RE using a new operation, called *selective mate*. We conjecture that it is not possible to obtain Turing completeness using only one of the six operations. We also conjecture that the pairs of operations we have considered for completeness, in this paper, are complete: it is impossible to obtain Turing completeness with any other pair of operations.

Keywords: Brane calculi; Membrane computing; Universality; Matrix grammars

1. Introduction

Membrane computing [12] and brane calculi [3] start from the same reality, viz., the living cell, but they develop in different directions and have different objectives. Membrane computing tries to abstract the computing power of biologically inspired models in the Turing sense, whereas brane calculi work in the framework of process algebra. Various operations on membranes appear in both areas. Quoting [4], the objectives of brane calculi and membrane computing are different: While membrane computing is a branch of natural computing, which tries to abstract computing models, in the Turing sense, from the structure and functioning of the cell, making especially use of automata, languages and complexity theoretic tools, brane calculi pay more attention to fidelity to biological reality, have as primary target systems biology, and use especially the framework of process algebra. Another difference is concerned with the semantics of the two formalisms: whereas brane calculi are equipped with an interleaving, sequential semantics (each computational step consists of execution of a single instruction), the semantics in membrane computing is based on maximal parallelism (each computational step consists of a maximal set of independent interactions). In membrane computing, membranes are supposed to be compartments of a cell, and computation is carried out on the objects in these compartments. Brane calculi, on the other hand, put emphasis on the structure, properties and evolution of membranes. All these similarities and differences have evoked interest in

E-mail address: krishnas@cse.iitb.ac.in.

0304-3975/$ - see front matter
doi:10.1016/j.tcs.2006.10.016

Table 1
Summary of results

Operations	Number of membranes	Weight	RE	Where
Pino, exo	8	4, 3	Yes	Theorem 6.1
Phago, exo	9	5, 2	Yes	Theorem 6.2
Phago, exo	9	4, 3	Yes	Theorem 6.2
Bud, mate	7	5, 3	Yes	Theorem 6.3
Mate, drip	11	5, 5	Yes	[4]
Mate, drip	4	3, 3	Yes	Theorem 6.4
Sel. mate	5	3	Yes	Theorem 7.1
Any other	*,*	*	No?	To be proved

bridging the gap between the two. A first attempt to bring together brane calculi and membrane computing was made in [4]. In [4], the modelling was done precisely in the same way as happened in brane calculi: always working with proteins embedded in the membranes, there were no 'objects' inside any of the membranes. A computation however, was defined in the usual way that happens in the membrane computing area: rules are applied to the embedded proteins in a maximally parallel manner, and halting configurations were used to interpret the results.

Two brane calculi viz., *pino, exo, phago* (PEP) and *mate, drip, bud* (MBD) were considered [1] and it was observed that the first calculi is more expressible than the second (Turing universality in the first case, decidability in the second). In [1], a deterministic encoding of RAMs in PEP (a basic brane calculus with interaction primitives inspired by endocytosis and exocytosis) was investigated and it was shown that the universal as well as existential termination problems for PEP were undecidable. It was further shown that in the case of MBD, universal termination is decidable. The problem of existential termination of MBD was taken up in [2], by providing a non-deterministic encoding of RAMs in MBD, and thereby showing the undecidability of existential termination for MBD. It was further observed in [2] that the computational power of MBDs is increased by incorporating the maximal parallelism semantics of membrane computing. By exploiting the maximal progress, a deterministic encoding of RAMs in MBD was given, obtaining the undecidability of both existential and universal termination for MBD with maximal parallelism. This also confirms the comparison of the computing power of sequential vs. parallel semantics of several variants of P systems.

The power of the MBD (in fact only mate, drip) was investigated in [4] and it was shown that these systems are Turing complete, thereby confirming the increase in power as observed in [2].

[11] is another instance where membrane computing and brane calculi have been brought together. Here, two operations of brane calculi, *endocytosis* and *exocytosis*, were used in a membrane computing framework. As usual, the objects were considered to be inside the compartments, but the operations available were (1) object evolution (2) endocytosis, by which an elementary membrane enters another membrane, and (3) exocytosis, by which an elementary membrane comes out of another membrane. This variant was shown to be computationally complete with 9 membranes. This result was later improved to 4 membranes [9], and later to 3 membranes [10].

In this paper, we take a closer look at various operations of brane calculi from the perspective of membrane computing. Precisely, we consider the pino–exo (PIE), phago–exo (PHE), bud–mate (BUM), mate–drip (MAD) calculi and observe that all of them are Turing complete. In fact, we give a very good improvement to the mate–drip calculi result obtained in [4]. Further, intuitively, it seems that without the exo and mate operations, it is impossible to obtain universality, irrespective of the number of membranes used. We propose a new operation called selectively mate, and prove that we can obtain completeness with this single operation. A summary of the results (existing as well as new ones conjectured) are given in Table 1.

2. The pino/exo/phago/bud/mate/drip calculi

We give an informal introduction to the basic operations of brane calculi, as a prelude to their use later in P systems. A detailed description and formal treatment can be found in [3]. A membrane structure is a collection of nested membranes. Membranes are formed of patches s, where each patch can be a composition of sub patches s_1, s_2. An elementary patch consists of an action a followed by another patch $a.s$. Actions often come in complementary pairs causing interactions between subsystems.

Each specific brane calculus has a specific set of actions with specific operational meanings. Let us first consider the pino action, which has no complementary co-action. The action pino creates an empty bubble within the membrane where the pino action resides. We can imagine that the original membrane buckles towards the inside and pinches off. In the following $\rightarrow$ stands for 'reduces to'.

$$(pino) \quad [\ P]\ (u|pino(s).t) \ \rightarrow \ [\ P[\ ]\ s]\ u|t.$$

The exo action however, comes with a complementary co-action: the operation models the merging of two nested membranes, which starts with the membranes touching at a point.

$$(exo) \quad [\ [\ P]\ u|(exo.t)Q]\ w|(co-exo.v) \ \rightarrow \ P[\ Q]\ u|w|t|v.$$

In the process, the subsystem P gets expelled to the outside, and all residual patches become contiguous.

Next we consider the phago action, which also comes with a complementary co-action: this models a membrane (one with Q) eating another membrane (one with P). It proceeds with the Q membrane wrapping around the P membrane, and joining itself on the other side.

$$(phago) \quad [\ P]\ u|(phago.t)[\ Q]\ w|(co-phago(s).v) \ \rightarrow \ [\ [\ [\ P]\ u|t]\ sQ]\ w|v.$$

We now consider the following three operations

$$(drip) \quad [\ P]\ u|(drip(s).t) \ \rightarrow \ [\ P]\ u|t[\ ]\ s,$$
$$(mate) \quad [\ P]\ u|(mate.t)[\ Q]\ w|(co-mate.v) \ \rightarrow \ [\ PQ]\ u|t|wv,$$
$$(bud) \quad [\ [\ P]\ u|(bud.t)Q]\ w|(co-bud(s).v) \ \rightarrow \ [\ [\ P]\ u|t\ ]\ s[\ Q]\ w|v.$$

Drip produces an empty bubble like pino, but outside the membrane. Mate merges two membranes like exo, but here the membranes are not nested. Bud expels a membrane from inside a membrane, wrapping an additional layer around it.

3. Prerequisites

We refer to [8,14] for all notions of formal language theory. Details of matrix grammars can be found in [5,6].

For an alphabet V, we denote by V^* the set of all strings over V; λ denotes the empty string. V^* is a monoid with λ, the empty string as its unit element. The length of a string $x \in V^*$ is denoted by $|x|$, and $|x|_a$ denotes the number of occurrences of symbol a in x.

Parikh vector: For $V = \{a_1, \ldots, a_n\}$, the Parikh mapping associated with V is $\psi_V : V^* \rightarrow N$ defined by $\psi_V(x) = (|x|_{a_1}, \ldots, |x|_{a_n})$, for all $x \in V^*$. For a language L, the Parikh set of L, $\psi_V(L) = \{\psi_V(x) \mid x \in L\}$ is the set of all Parikh vectors of all words $x \in L$. For a family FL of languages we denote by $PsFL$ the family of sets of vector numbers in FL. Thus, for the four basic families of languages in the Chomsky hierarchy—regular (REG), context-free (CF), context-sensitive (CS) and recursively enumerable (RE), the family of vectors computed are denoted respectively by $PsREG$, $PsCF$, $PsCS$ and $PsRE$.

Left derivative of a language: The left derivative of $L \subseteq V^*$ with respect to a string x is defined as $\partial_x^l(L) = \{w \in T^* \mid xw \in L\}$.

Multisets: A multiset over an alphabet $V = \{a_1, \ldots, a_n\}$ is a map $m : V \rightarrow N$. Since the natural extension of m to strings over V give the Parikh mapping, we can represent the multiset m by any string $w \in V^*$ such that $\psi_V(w) = (m(a_1), \ldots, m(a_n))$.

3.1. Matrix grammars

In our proofs for universality, we characterize recursively enumerable languages by matrix grammars with appearance checking. Such a grammar is a construct $G = (N, T, S, M, F)$, where N, T are disjoint alphabets, $S \in N$, M is a finite set of sequences of the form $(A_1 \rightarrow x_1, \ldots, A_n \rightarrow x_n)$, $n \geq 1$, of context-free rules over $N \cup T$ (with $A_i \in N, x_i \in (N \cup T)^*$, in all cases), and F is a set of occurrences of rules in M (we say that N is the nonterminal alphabet, T is the terminal alphabet, S is the axiom, while the elements of P are called matrices).

For $w, z \in (N \cup T)^*$, we write $w \Rightarrow z$ if there is a matrix $(A_1 \rightarrow x_1, \ldots, A_n \rightarrow x_n)$ in M and the strings $w_i \in (N \cup T)^*, 1 \leq i \leq n+1$, such that $w = w_1, z = w_{n+1}$, and, for all $1 \leq i \leq n$, either

$w_i = w'_i A_i w''_i, w_{i+1} = w'_i x_i w''_i$, for some $w'_i, w''_i \in (N \cup T)^*$, or $w_i = w_{i+1}$, A_i does not appear in w_i, and the rule $A_i \to x_i$ appears in F. The rules of a matrix are applied in order, possibly skipping the rules in F if they cannot be applied; we say that these rules are applied in the *appearance checking mode*. If $F \neq \emptyset$, then the grammar is said to be without appearance checking (and F is no longer mentioned).

We denote by $\Rightarrow^*$ the reflexive and transitive closure of the relation $\Rightarrow$. The language generated by G is defined by $L(G) = \{w \in T^* \mid S \Rightarrow^* w\}$. The family of languages of this form is denoted by MAT_{ac}. When we use only grammars without appearance checking, then the obtained family is denoted by MAT.

As an example of a matrix grammar with appearance checking, consider the grammar $G = (\{X, Y, Z, U, A\}, \{a\}, X, M)$ where M consists of the matrices

$$[Y \to U, A \to U, X \to ZZ],\ [X \to U, Z \to Y],\ [Z \to U, Y \to X],$$
$$[Y \to U, Z \to U, X \to A],\ [X \to U, A \to a].$$

Define the set $F = \{X \to U, Y \to U, Z \to U, A \to U\}$. Then, it can be seen that $L(G) = \{a^{2^n} \mid n \geq 0\}$.

We shall briefly look at matrix grammars in the *strong binary normal form*. Such a grammar is a construct $G = (N, T, S, M, F)$, where $N = N_1 \cup N_2 \cup \{S, \#\}$, with these three sets mutually disjoint, two distinguished symbols $B^{(1)}, B^{(2)} \in N_2$, and the matrices in M of one of the following forms:

(1) $(S \to XA)$, with $X \in N_1, A \in N_2$,
(2) $(X \to Y, A \to x)$, with $X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*$,
(3) $(X \to Y, B^{(j)} \to \#)$, with $X, Y \in N_1, j = 1, 2$,
(4) $(X \to \lambda, A \to x)$, with $X \in N_1, A \in N_2, x \in T^*$.

Moreover, there is only one matrix of type 1 and F consists of all the rules $B^{(j)} \to \#$, $j = 1, 2$, appearing in matrices of type 3; # is a trap-symbol, once introduced it is never removed. (Clearly, a matrix of type 4 is used only once, in the last step of a derivation.)

In [7] it is proved that each recursively enumerable language can be generated by a matrix grammar in the strong binary normal form.

If we ignore the empty string when comparing languages, then the rules of type 4 above can be considered of the form $(X \to a, A \to x)$, for $X \in N_1, a \in T, A \in N_2$, and $x \in T^*$.

Here is a (sketched) proof of this assertion from [11]. For a language $L \in RE$, $L \subseteq T^*$, we write $L = \bigcup_{a \in T} \{a\} \partial_a(L)$, where $\partial_a(L) = \{w \in T^* \mid aw \in L\}$ is the left derivative of L with respect to a. Consider a matrix grammar G_a in the strong binary normal form for each language $\partial_a(L)$, and replace each matrix $(X \to \lambda, A \to x)$ of type 4 from G_a with $(X \to a, A \to x)$. Let $G_a = (N_a, T, S_a, M_a, F_a)$ be the grammar obtained in this way, $L(G_a) = \partial_a(L)$. Let us assume all sets $N_{a,1}, a \in T$, mutually disjoint (we can "colour" each $X \in N_{a,1}$ with a, for instance, using symbols X^a). The sets $N_{a,2}$ are left unchanged—in particular, the corresponding symbols $B^{(1)}, B^{(2)}$ are denoted in the same way in all grammars. We construct the grammar $G' = (N', T, S', M', F')$, with

$$\begin{aligned} N' &= \bigcup_{a \in T} N_a \cup \{S', X_0, A_0\},\ S', X_0, A_0 \text{ are new symbols,} \\ M' &= \{(S' \to X_0 A_0)\} \\ &\quad \cup \{(X_0 \to X_a, A_0 \to A_a) \mid (S_a \to X_a A_a) \in M_a, a \in T\} \\ &\quad \cup \bigcup_{a \in T} (M_a - \{(S_a \to X_a A_a)\}). \end{aligned}$$

Obviously, $L(G') = L(G)$ (because the alphabets $N_{a,1}, a \in T$, are mutually disjoint, we cannot mix matrices from a grammar $G_a, a \in T$, when deriving strings from $L(G_b)$ with $a \neq b$ (remember also that we ignore the empty string). The grammar G' is in the strong binary normal form, and its matrices of type 4 are of the form $(X \to a, A \to x)$, for some $a \in T$. A grammar with these properties is said to be in the *improved strong binary normal form* [11].

4. Pino/exo/mate/drip/phago/bud as membrane computing operations

We refer to [13,15] for details on membrane computing, we shall describe only the system of interest here. We start by writing the six operations we use below in the formalism of membrane computing. As usual, we represent a membrane by a pair of parentheses, [ ]. As in [4], we associate multisets of proteins with membranes to refer to them.

A membrane having associated a multiset u of proteins is written in the form $[\ ]_u$. We also say that the membrane is marked with the multiset u.

Taking an alphabet V of proteins, we write the six operations as follows, of which 1–6 have been defined in [4]. In this paper, we assume that $u, x \in V^*$, $a, b \in V$, $v \in V^+$ with $uxv \in V^+$.

1. $pino_i : [\ P]_{uav} \to [\ [\ ]_{ux} P\ ]_v$
2. $exo_i : [\ [\ P]_{ua}\ Q]_v \to P[\ Q\ ]_{uxv}$
3. $pino_e : [\ P]_{uav} \to [\ [\ ]_v P\ ]_{ux}$
4. $exo_e : [\ [\ P]_u Q]_{av} \to P[\ Q]_{uxv}$
5. $mate : [\ P]_{ua}\ [\ Q]_v \to [\ PQ]_{uxv}$
6. $drip_1 : [\ P]_{uav} \to [\ ]_{ux}\ [\ P]_v$
7. $drip_2 : [\ P]_{uav} \to [\ P]_{ux}\ [\ ]_v$
8. $phago_i : [\ P]_b\ [\ Q]_{uav} \to [\ [\ [\ P]_b]_{ux} Q]_v$
9. $phago_e : [\ P]_b\ [\ Q]_{uav} \to [\ [\ [\ P]_b]_v Q]_{ux}$
10. $bud_1 : [\ [\ P]_b Q_1 Q_2]_{uav} \to [\ [\ P]_b\ Q_1]_{ux}\ [\ Q_2\ ]_v$
11. $bud_2 : [\ [\ P]_b Q_1 Q_2]_{uav} \to [\ [\ P]_b Q_1]_v\ [\ Q_2\ ]_{ux}$.

Rules 1–11 are applied keeping in mind the following:

(1) P, Q_1, Q_2 represent the contents of the membranes. Thus, they may be objects if the membranes are elementary, or may be membrane structures themselves.
(2) There is no ordering between adjacent membranes.
(3) If $Q = [\ P_1]\ [\ P_2]\ [\ P_3]$, where P_1, P_2, P_3 are membrane sub structures, then Q can be written as $Q_1 Q_2$, where
   - $Q_1 = [\ P_i]\ [\ P_j]$ and $Q_2 = [\ P_k\ ]$ with $i, j, k \in \{1, 2, 3\}$, $i \neq j \neq k$, or
   - $Q_2 = [\ P_i]\ [\ P_j]$ and $Q_1 = [\ P_k\ ]$ with $i, j, k \in \{1, 2, 3\}$, $i \neq j \neq k$, or
   - $Q_1 = Q$ and Q_2 is empty, or $Q_2 = Q$ and Q_1 is empty.
(4) All membranes involved in a rule have non-empty multisets associated with them. In each case, multisets of proteins are transferred from input membranes to output membranes as indicated in the rules, with protein a evolving into the multiset x. Note that the multisets u, v, b and the protein a marking the left hand side membranes correspond to the multisets u, v, b and x in the right hand sides.
(5) The difference between $pino_i, pino_e$ (similarly for others like exo, phago) is that in the first case the main role is played by the internal membrane, whereas in the second it is played by the external membrane. Note that the two versions of *bud* indicate the choice of where the protein x can be placed: either as part of the 'wrap around' membrane or as part of the other one.
(6) A rule is applied to a membrane(s) if it(they) contain the multiset uav which have an applicable rule to them.
(7) All proteins in a membrane which do not form part of a rule are unaffected by the rule. In the case of pino, drip, these proteins are randomly distributed between the two resulting membranes. In the case of phago and bud, the contents of the membrane containing b is unaffected, the contents of the membrane containing uav is distributed between the two membranes containing ux and v.
(8) Mate and Exo always have a unique resultant membrane, merging the contents of two membranes.

We will illustrate the above points by examples. In the following examples, let $w_1 w_2$ be a distribution of w (which means that the symbols of w_1, w_2 put together give the symbols of w). Note that there are several possible ways of doing this, one possibility being $w_1 = \lambda$ or $w_2 = \lambda$. Also, let $x_1, x_2, x_3 \in V^*$. Below, we examine the effect of a few rules on some membrane structures.

(1) $[\ P]_{uavw}$.
   (a) $pino_i$: We get $[\ P\ [\ ]_{w_1 ux}\ ]_{w_2 v}$.
   (b) $pino_e$: We get $[\ P\ [\ ]_{w_1 v}\ ]_{w_2 ux}$
   (c) $drip$: we get either $[\ P]_{w_1 ux}\ [\ ]_{w_2 v}$ or $[\ ]_{w_1 ux}\ [\ P]_{w_2 v}$.
(2) $[\ [\ P]_{bx_1}\ [\ Q]_{x_2}\ [\ R]_{x_3}\ ]_{wuav}$.
   (a) A bud_1 operation $[\ [\ P]_{bx_1}\ ]_{wav}$ can result in

- $[\ [\ P]_{bx_1}\ [\ Q]_{x_2}]_{w_1ux}\quad [\ [\ R]_{x_3}\ ]_{w_2v}$ or
- $[\ [\ P]_{bx_1}\ [\ R]_{x_3}]_{w_1ux}\quad [\ [\ Q]_{x_2}\ ]_{w_2v}$ or
- $[\ [\ P]_{bx_1}\ [\ Q]_{x_2}\ [\ R]_{x_3}\ ]_{w_1ux}\quad [\ ]_{w_2v}$ or
- $[\ [\ P]_{bx_1}\ ]_{w_1ux}\quad [\ [\ Q]_{x_2}\ [\ R]_{x_3}\ ]_{w_2v}$.

(b) The bud_2 operation $[\ [\ P]_{bx_1}\ ]_{wav}$ is similar.

(3) $[\ P]_{x_1b}\ [\ [\ Q]_{x_2}\ [\ R]_{x_3}]_{wuav}$.

(a) The $phago_i$ operation $[\ P]_{x_1b}\ [\ ]_{wuav}$ gives $[\ [\ [\ P]_{x_1b}\ ]_{w_1ux}\ [\ Q]_{x_2}\ [\ R]_{x_3}\ ]_{w_2v}$.

We shall henceforth refer only to the generic names of pino, exo, bud etc. instead of specifying $pino_i$, exo_e, bud_1 etc.

Note: In the operations of pino, drip, phago and bud, there is a non-deterministic distribution of the objects which do not evolve. If we had restricted the definition so that all objects which are not part of a rule remain in their respective membranes, unaffected, then it would have been easy to predict where a particular object can be found during a computation. In all the results in this paper, we do not find any extra expressiveness due to the non-deterministic distribution of objects, and in fact, write extra rules to keep track of the possible locations of objects, due to the non-deterministic distribution. We would have had a lesser set of rules if we had not allowed the random distribution.

Anyway, the power of random distribution needs to be investigated more systematically.

5. P systems based on pino/exo/phago/mate/drip/bud operations

Using the rules defined above, we can define a P system as follows:

$$\Pi = (V, \mu, w_1, \ldots, w_m, R)$$

where

- V is the basic finite alphabet of proteins;
- μ is a membrane structure with $m \geq 2$ membranes;
- $w_1, \ldots, w_m$ are multisets of proteins bound to the m membranes of μ at the beginning of the computation (we assume that the membranes have a precise identification by means of their labels); the skin membrane is labeled 1 and is marked with the multiset λ;
- R is a finite set of pino/exo/phago/mate/drip/bud rules of the form specified above, with the proteins from the alphabet V.

Note that here the skin membrane has no protein associated with it. It actually plays no role in the computation other than forming a boundary to all membranes within it. Note that since none of our rules can be used with λ as the sole content of any membrane, no rule can be applied to the skin membrane. There are no objects in any of the compartments, they can contain other membranes inside.

The following notions are important while applying rules:

- The membrane(s) in the left hand side of a rule are said to be "involved", and the membranes on the right hand side are "produced".
- The protein a specified in the left hand side of a rule is said to be consumed and is replaced by x. Note that the other proteins which mark the membranes remain unchanged, and are reproduced in the new membranes produced.
- In the case of the exo and mate rules, all proteins from both the membranes are inherited by the new membrane. In the case of pino, drip, bud, phago etc. (as described in the previous section), the proteins of the membrane which are not involved are non-deterministically distributed to the new membranes.

5.1. Evolution of the system

The evolution of the system is through transitions among configurations, based on *non-deterministic maximal parallel* use of rules. A configuration consists of the membrane structure as well as all multisets marking the membranes. The initial configuration is $\mu, w_1, \ldots, w_m$. In each step, a membrane may be part of at most one rule, with the restriction that the choice of the rules must be maximal (after choosing some rules to apply, no further rule can be applied to the membranes not involved in the chosen rules). A membrane remains unchanged if not evolving

by a rule. Note that evolution is parallel at the level of membranes, but sequential at the level of multisets marking each membrane: at most one protein a evolves by a rule, and at most one rule is applied to each membrane.

A sequence of transitions forms a *computation*. A computation which starts from the initial configuration is *successful* if it halts, i.e, reaches a configuration wherein no rule can be applied. The result of a successful computation is given by the multiset which marks the membrane just below the skin membrane in a halting computation. If there are several membranes just below the skin membrane, then we take the multiset obtained by putting together the multisets of all such membranes as the output. We consider the vector describing the multiplicity of proteins in this multiset to be computed by Π. Note that unlike [4], the acceptance condition given here does not bother about the number of membranes and the structure of μ at the end of a halting configuration, and hence is more general.

Since rules are applied non-deterministically, we can get several computations starting from the initial configuration, of which many may be successful. The set of all vectors computed this way by Π is denoted $Ps(\Pi)$.
Convention: When comparing the power of two systems Π, Π', we consider $Ps(\Pi) = Ps(\Pi')$ if and only if $Ps(\Pi) - \{(0, 0, \ldots, 0)\} = Ps(\Pi') - \{(0, 0, \ldots, 0)\}$, that is, the null vector is ignored. This corresponds to the fact that, in many cases in language theory, when comparing two grammars or automata the empty string is ignored—and this assumption holds good here since we have assumed uxv to be nonempty.

The length of the string uxv is called the *weight* of a rule.

In what follows, we shall investigate the power of P systems using pino/exo, phago/exo, mate/drip and bud/mate operations. The family of all sets of vectors $Ps(\Pi)$ computed by P systems Π using at any moment during a halting computation at most m membranes, and any of the rules $r_1, r_2 \in \{$ *mate, drip, exo, bud, pino, phago* $\}$ of weight at most r, s, respectively, is denoted by $PsOP_m(r_1(r), r_2(s))$. When one of the parameters is not bounded, we replace it with a *.

5.2. Examples

We consider the following pairs of rules: pino/exo, phago/exo, bud/mate, drip/mate. We will consider some intuitive examples of the maximal parallelism possible while using these pairs of rules:

(1) Consider a membrane structure $[\ [\ [\ ]_a\ ]_b\ [\ ]_{ef}\ ]_c$ and rules
(i) $[\ ]_{ef} \rightarrow_{pino} [\ [\ ]_e\ ]_f$ (ii) $[\ [\ ]_b\ ]_c \rightarrow_{exo} [\ ]_{bd}$.
Applying (i) and (ii) in parallel, we obtain $[\ ]_a\ [\ [\ ]_e\ ]_f\ ]_{bd}$.
(2) Consider the structure $[\ [\ [\ [\ [\ ]_a\ ]_b\ ]_c\ ]_d\ ]_e$, and rules
(i) $[\ [\ ]_c\ ]_d \rightarrow_{exo} [\ ]_{cd}$ (ii) $[\ [\ ]_a\ ]_b \rightarrow_{exo} [\ ]_{ab}$. We obtain $[\ [\ ]_{ab}\ [\ ]_{cd}\ ]_e$.
(3) Consider $[\ [\ ]_a\ [\ ]_{ef}\ [\ [\ ]_b\ ]_c\ ]_d$, and rules
(i) $[\ ]_a\ [\ ]_{ef} \rightarrow_{phago} [\ [\ [\ ]_a\ ]_{e'}\ ]_f$ (ii) $[\ [\ ]_c\ ]_d \rightarrow_{exo} [\ ]_d$. Then we obtain $[\ [\ [\ [\ ]_a\ ]_{e'}\ ]_f\ [\ ]_b\ ]_d$.

Lemma 5.1. • $PsOP_m(r_1(i), r_2(j)) \subseteq PsOP_{m'}(r_1(i'), r_2(j'))$, *for all* $m \leq m', i \leq i', j \leq j'$.
• $PsOP_*(r_1(*), r_2(*)) \subseteq PsRE$.

The above result can be easily proved based on the following observation: Given a system Π with m membranes, and rules of type r_1, r_2, with weights i, j, it is easy to construct another system Π' with $m' \geq m$ membranes, and having rules of type r_1, r_2 with weights $i' \geq i, j' \geq j$ such that $Ps\Pi \subseteq Ps\Pi'$ as follows: keep the initial contents of the m membranes of Π unaltered as it is in Π', and let the initial contents of the extra $m' - m$ membranes be empty. Do not involve these membranes in any rules. Now, coming to the rules of Π', add all rules of Π and any extra rules of a greater weight, as required. Clearly, Π' can generate all vectors that Π can generate.

6. Universality results

In this section, we explore the power of P systems in the mate/drip, pino/exo, phago/exo and bud/mate cases. The power of mate/drip was already investigated in [4], we give an improvement here.
Note 1: In all results in the following section, we shall denote by $a \rightarrow_{ph} b, a \rightarrow_{pin} b, a \rightarrow_{exo} b, a \rightarrow_{bud} b$, and $a \rightarrow_{mate} b$, the evolution of a into b using phago ($phago_i$ or $phago_e$), pino ($pino_i$ or $pino_e$), exo (exo_i or exo_e), or bud, mate rules. Here a, b are the corresponding membrane structures.
Note 2: While specifying a P system Π, we shall write the multisets associated to membranes in an initial configuration as bound to the membranes, instead of writing them separately.

Note 3: In all the proofs below, we use the notation $C_1 \rightarrow^{r_1,\ldots,r_n} C_2$ to denote that configuration C_1 has evolved into configuration C_2 by application of rules $r_1, \ldots, r_n$ in parallel. Similar are the cases with C_1 and $C_2 \leftarrow^{r_1,\ldots,r_n} C_1$.
$\downarrow^{r_1,\ldots,r_n}$
C_2

Note 4: In all the universality proofs (Sections 6 and 7), we consider a matrix grammar $G = (N, T, S, M, F)$ in the improved strong binary normal form (hence with $N = N_1 \cup N_2 \cup \{S, \#\}$), having n_1 matrices of types 2,4 (that is, not used in appearance checking mode), and n_2 matrices of type 3 (with appearance checking rules). Let $B^{(1)}$ and $B^{(2)}$ be the two objects in N_2 for which we have rules $B^{(j)} \rightarrow \#$ in matrices of M. The matrices of the form $(X \rightarrow Y, B^{(j)} \rightarrow \#)$ are labeled by m'_i, with $i \in lab_j$, for $j \in \{1, 2\}$, such that lab_1, lab_2 and $lab_0 = \{1, 2, \ldots, n_1\}$ are mutually disjoint sets.

6.1. The PIE calculus

In this section, we look at the computing power of pino–exo operations, and prove their universality.

Theorem 6.1. *$PsRE = PsOP_m(pino(r), exo(s))$, for all $m \geq 8, r \geq 4, s \geq 3$.*

Proof. In view of Lemma 5.1, we only prove the inclusion $PsRE \subseteq PsOP_8(pino(4), exo(3))$, using the equality $PsRE = PsMAT_{ac}$, by considering a matrix grammar with appearance checking in the *improved strong binary normal form* [11].

We construct a P system

$$\Pi = (V, [\ [\ [\ ]_{EA}\ [\ ]_{D_1Q}\ [\ ]_{D_2Q}\ [\ ]_L\ ]_X\ ]_\lambda, R)$$

with the alphabet

$$\begin{aligned} V\ &= \{X, X_l, X_l^{(i)}, X'_l \mid X \in N_1, 1 \leq l \leq n_1 + n_2, 1 \leq i \leq 2\} \\ &\cup\ \{\alpha, \alpha', \mid \alpha \in N_2 \cup T\} \cup \{Z_a \mid a \in T\} \\ &\cup\ \{f, g, D, D', D_1, D_2, E, Q, Q', Q_1, Q_2, Q_3, Q_4, Q_5, H, H', H'', H''', L\}. \end{aligned}$$

Any computation starts from the initial configuration $[\ [\ [\ ]_{EA}\ [\ ]_{D_1Q}\ [\ ]_{D_2Q}\ [\ ]_L\ ]_X\ ]_\lambda$ where $S \rightarrow XA$ is the initial matrix of G.

Proof idea: We give an overview of the proof before going into the details. The membranes labeled EA and X are used in the simulation of a non-terminal type 2 matrix. All symbols of $N_1 \cup N_2 \cup T$ will be distributed between these two membranes. The membranes labeled D_1Q, D_2Q are used for checking if the symbols $B^{(1)}$, $B^{(2)}$ appear during simulation of a type 3 matrix. The membrane labeled L is used for testing if all symbols $A \in N_2$ are replaced by terminals after simulation of the terminal matrix. If any $A \in N_2$ is found, an infinite computation is induced by L, giving no result.

The set R of rules is constructed as follows:

(1) Simulation of a non-terminal matrix $m_l : (X \rightarrow Y, A \rightarrow x),\ X, Y \in N_1, A \in N_2,\ x \in (N_2 \cup T)^*,\ 1 \leq l \leq n_1$.

1. $[\ [\ ]_{AE}\ ]_X \rightarrow_{exo} [\ ]_{X_lAE}$,
   $[\ [\ ]_E\ ]_{XA} \rightarrow_{exo} [\ ]_{X_lEA}$,
2. $[\ ]_{X_lAE} \rightarrow_{pin} [\ [\ ]_{x'E}\ ]_{X_l}$, if $x \neq \lambda$,
   (If $m_l : (X \rightarrow Y, A \rightarrow \alpha_1\alpha_2)$, then, $x' = \alpha'_1\alpha_2$ or $\alpha_1\alpha'_2$,
   and if $m_l : (X \rightarrow Y, A \rightarrow \alpha_1)$, then $x' = \alpha'_1$.)
   $[\ ]_{X_lAE} \rightarrow_{pin} [\ [\ ]_{fE}\ ]_{X_l}$, if $x = \lambda$,
3. $[\ [\ ]_{\alpha'E}\ ]_{X_l} \rightarrow_{exo} [\ ]_{\alpha'EX'_l}$, $\alpha \in N_2 \cup T$
   $[\ [\ ]_{fE}\ ]_{X_l} \rightarrow_{exo} [\ ]_{fEX'_l}$,
4. $[\ ]_{\alpha'EX'_l} \rightarrow_{pin} [\ [\ ]_{\alpha g}\ ]_{X'_lE}$, $\alpha \in N_2 \cup T$
   $[\ ]_{fEX'_l} \rightarrow_{pin} [\ [\ ]_g\ ]_{X'_lE}$,
5. $[\ [\ ]_g\ ]_{X'_lE} \rightarrow_{exo} [\ ]_{YgE}$,

6. $[\ ]_{EgY} \rightarrow_{pin} [\ [\ ]_E\ ]_Y$,
7. $[\ [\ ]_E\ ]_X \rightarrow_{exo} [\ ]_{\#E}$,
   (X does not correspond to a type 3 matrix)
8. $[\ ]_{\#\beta} \rightarrow_{pin} [\ [\ ]_{\#}\ ]_{\#}$, $\beta \in V \cup \{\#\}$,
9. $[\ [\ ]_{\#}\ ]_{\#} \rightarrow_{exo} [\ ]_{\#\#}$,
10. $[\ [\ ]_{D_j Q} \rightarrow_{pin} [\ [\ ]_{D_j}\ ]_{Q_1}$,
11. $[\ [\ ]_{D_j}\ ]_{Q_1} \rightarrow_{exo} [\ ]_{D_j Q_2}$,
12. $[\ ]_{D_j Q_2} \rightarrow_{pin} [\ [\ ]_{D_j}\ ]_{Q_3}$,
13. $[\ [\ ]_{D_j}\ ]_{Q_3} \rightarrow_{exo} [\ ]_{D_j Q_4}$,
14. $[\ ]_{D_j Q_4} \rightarrow_{pin} [\ [\ ]_{D_j}\ ]_{Q_5}$,
15. $[\ [\ ]_{D_j}\ ]_{Q_5} \rightarrow_{exo} [\ ]_{D_j Q}$.

We shall now examine the simulation. We start with rule 1. Initially, the symbol $A \in N_2$ corresponding to the initial matrix will be part of the inner membrane. In later steps, however, since we use pino rules, the corresponding A may be found along with X as well. In any case, we replace X by X_l marking the beginning of the simulation. This is followed by rule 2, wherein the symbols X_l, A are used to replace A by either x' or f (in case $x = \lambda$). Next, we apply rule 3, replacing X_l by X'_l; this prevents replacing any more A's. Now, using X'_l, we replace α' by α (or erase f), as the case may be, simultaneously introducing a new symbol g. The new symbol g is used to replace X'_l by Y. The last step involves removal of g while coming back to the original configuration (having membrane labeled E inside the membrane labeled Y). Note that while we apply rules 1 to 6, we also use in parallel, rules 10 through 15. These rules play an important role in the simulation of type 3 matrices. But anyway, when we end the simulation with rule 6, we also use rule 15 in parallel, thereby preserving the initial membrane structure. Note that rule 7 introduces a trap symbol if the corresponding $A \in N_2$ is not present. We now illustrate the evolution of the configurations during one simulation of a type 2 matrix.

$$
\begin{array}{l}
[\ [\ [\ ]_{EA}\ [\ ]_{D_1 Q}\ [\ ]_{D_2 Q}\ [\ ]_L\ ]_X\ ]_\lambda \rightarrow^{1,10} [\ [\ [\ [\ ]_{D_1}\ ]_{Q_1}\ [\ [\ ]_{D_2}\ ]_{Q_1}\ [\ ]_L\ ]_{X_l EA}\ ]_\lambda \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow^{2,11} \\
[\ [\ [\ [\ ]_{D_1}\ ]_{Q_3}\ [\ [\ ]_{D_2}\ ]_{Q_3}\ [\ ]_L\ ]_{X'_l E\alpha'}\ ]_\lambda \leftarrow^{3,12} [\ [\ [\ ]_{Ex'}\ [\ ]_{D_1 Q_2}\ [\ ]_{D_2 Q_2}\ [\ ]_L\ ]_{X_l}\ ]_\lambda \\
\qquad\qquad\quad \downarrow^{4,13} \\
[\ [\ [\ [\ ]_{\alpha g}\ [\ ]_{D_1 Q_4}\ [\ ]_{D_2 Q_4}\ [\ ]_L\ ]_{X'_l E}\ ]_\lambda \rightarrow^{5,14} [\ [\ [\ [\ ]_{D_1}\ ]_{Q_5}\ [\ [\ ]_{D_2}\ ]_{Q_5}\ [\ ]_L\ ]_{Y g E}\ ]_\lambda \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow^{6,15} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad [\ [\ [\ ]_E\ [\ ]_{D_1 Q}\ [\ ]_{D_2 Q}\ [\ ]_L\ ]_Y\ ]_\lambda .
\end{array}
$$

(2) Simulation of a type 3 matrix $m'_l : (X \rightarrow Y, B^{(j)} \rightarrow \#)$, $n_1 + 1 \leq l \leq n_2$
We start once again from $[\ [\ [\ ]_E\ [\ ]_{D_1 Q}\ [\ ]_{D_2 Q}\ [\ ]_L\ ]_X\ ]_\lambda$.

16. $[\ [\ ]_E\ ]_X \rightarrow_{exo} [\ ]_{X_l^{(j)} E}$,
17. $[\ ]_{X_l^{(j)} E} \rightarrow_{pin} [\ [\ ]_E\ ]_{X_l^{(j)}}$,
18. $[\ [\ ]_{B^{(j)}}\ ]_{X_l^{(j)}} \rightarrow_{exo} [\ ]_{\#B^{(j)}}$,
19. $[\ [\ ]_E\ ]_{B^{(j)} X_l^{(j)}} \rightarrow_{exo} [\ ]_{\#B^{(j)} E}$,
20. $[\ [\ ]_{D_j Q_4}\ ]_{X_l^{(j)}} \rightarrow_{exo} [\ ]_{D_j Q' X_l^{(j)}}$,
21. $[\ ]_{D_j Q' X_l^{(j)}} \rightarrow_{pin} [\ [\ ]_{D_j Q'}\ ]_Y$,
22. $[\ ]_{D_j Q'} \rightarrow_{pin} [\ [\ ]_{D_j}\ ]_{Q_1}$.

Rule 16 starts the simulation replacing X by $X_l^{(j)}$, thereby remembering the index l of the matrix. Rule 10 will be applied in parallel. This is followed by using rules 11,17. Now, we need to check if the corresponding symbol

$B^{(j)} \in N_2$ is present. The symbol $B^{(j)}$, if present, will be either part of the membrane labeled $X_l^{(j)}$ or part of the membrane labeled E. It cannot be present in the membrane labeled D_j, Q_i since these membranes have never interacted with any other membrane so far. If $B^{(j)}$ is present, rules 18 or 19 will be applied (in parallel we use rule 12). In the next step, if the symbol $X_l^{(j)}$ is still there, it means that the corresponding $B^{(j)}$ was not present in the previous step. We now apply rule 13, introducing the symbol Q_4. We can now either use rule 14 or rule 20. If we use rule 14, Q_4 will be replaced by Q_5, and subsequently by Q. This will be followed by the whole chain of rules 10–13 again, with $X_l^{(j)}$ and other symbols remaining the same. The simulation can be progressed only by using rule 20. To ensure progress, let us assume we use rule 20. This renames the Q_4 in $D_j Q_4$ into $D_j Q'$. In the next step, rule 21 (and in parallel rule 15 for the other $D_i Q$) is used. This replaces $X_l^{(j)}$ by Y and sends $[\ ]_{D_j Q'}$ inside the membrane labeled Y, adjacent to the membrane labeled $D_i Q$. We will now illustrate the simulation, assuming $j = 1$.

$$
\begin{array}{l}
[\,[\,[\,]_{EA}\,[\,]_{D_1 Q}\,[\,]_{D_2 Q}\,[\,]_L]_X]_\lambda \rightarrow^{16,10} [\,[\,[\,[\,]_{D_1}]_{Q_1}\,[\,[\,]_{D_2}]_{Q_1}\,[\,]_L]_{X_l^{(1)} E}]_\lambda \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow^{17,11} \\
[\,[\,[\,]_E\,[\,]_{D_1}]_{Q_3}\,[\,[\,]_{D_2}]_{Q_3}\,[\,]_L]_{X_l^{(1)}}]_\lambda \leftarrow^{12} [\,[\,[\,]_E\,[\,]_{D_1 Q_2}\,[\,]_{D_2 Q_2}\,[\,]_L]_{X_l^{(1)}}]_\lambda \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow^{13} \\
[\,[\,[\,]_E\,[\,]_{D_1 Q_4}\,[\,]_{D_2 Q_4}\,[\,]_L]_{X_l^{(1)}}]_\lambda \rightarrow^{14,20} [\,[\,[\,]_E\,[\,[\,]_{D_2}\,]_{Q_5}\,[\,]_L]_{D_1 Q' X_l^{(1)}}]_\lambda \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow^{21} \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad [\,[\,[\,]_E\,[\,]_{D_1 Q'}\,[\,]_{D_2 Q}\,[\,]_L]_Y]_\lambda .
\end{array}
$$

Note the use of the exo rule 20, wherein we unify the contents of the two membranes. Certainly, $B^{(j)}$ is not part of this membrane. When we segregate $D_j Q_4$ using the pino rule 21, non-deterministically some symbols of $N_2 \cup T$ other than $(B^{(j)})$ may get mixed up with it. Note that this does not create any future problem for appearance checking, since rule 20 can be applied only in the absence of $B^{(j)}$, but we might have a scenario wherein a symbol $A \in N_2$ becomes part of this inner membrane. If this happens, we may be unable to use the occurrence of this $A \in N_2$, since when we simulate matrices of type 2, we use rule 1, wherein we always look for E to occur with $A \in N_2$. Clearly, E will not be part of the membranes labeled $D_j Q$, and hence, we will not be able to use rule 1. However, in such a case, rule 7 will be used leading to an infinite computation.

(3) For each terminal matrix $m_l : (X \rightarrow a, A \rightarrow x), X, Y \in N_1, A \in N_2, x \in T^*, a \in T$, with $1 \leq l \leq n_1$, we consider the rules:
First use rules 1 and 2, obtaining $[\,[\,[\,]_{\alpha' E}\,[\,]_{D_1 Q_2}\,[\,]_{D_2 Q_2}\,[\,]_L\,]_{X_l}\,]_\lambda$ or $[\,[\,[\,]_{f E}\,[\,]_{D_1 Q_2}\,[\,]_{D_2 Q_2}\,]_{X_l}\,[\,]_L\,]_\lambda$. Now consider the following rules:

23. $[\,[\,]_{\alpha' E}\,]_{X_l} \rightarrow_{exo} [\,]_{\alpha' X_l H}$, $\alpha \in N_2 \cup T$,
    $[\,[\,]_{f E}\,]_{X_l} \rightarrow_{exo} [\,]_{f X_l H}$,
24. $[\,]_{\alpha' X_l H} \rightarrow_{pin} [\,[\,]_{\alpha' H}\,]_{Z_a}$,
    $[\,]_{f X_l H} \rightarrow_{pin} [\,[\,]_{f H}\,]_{Z_a}$,
25. $[\,[\,]_{\alpha' H}\,]_{Z_a} \rightarrow_{exo} [\,]_{Z_a \alpha H}$,
    $[\,[\,]_{f H}\,]_{Z_a} \rightarrow_{exo} [\,]_{Z_a H}$,
26. $[\,]_{Z_a H} \rightarrow_{pin} [\,[\,]_H\,]_{a H'}$,
27. $[\,[\,]_{D_1 Q}\,]_{H'} \rightarrow_{exo} [\,]_{H' D_1}$,
28. $[\,[\,]_H\,]_{H' D_1} \rightarrow_{exo} [\,]_{H' D_1}$,
29. $[\,[\,]_{D_2 Q_2}\,]_{H' D_1} \rightarrow_{exo} [\,]_{H' D_1 D_2}$,
30. $[\,]_{H' D_1 D_2} \rightarrow_{pin} [\,[\,]_{D'}\,]_{H' D_2}$,
31. $[\,[\,]_{D'}\,]_{H' D_2} \rightarrow_{exo} [\,]_{H' D'}$,
32. $[\,]_{a H' D'} \rightarrow_{pin} [\,[\,]_{a D'}]_{H''}$,
33. $[\,[\,]_{a D'}]_{H''} \rightarrow_{exo} [\,]_{a H''}$,

34. $[\ ]_{aH''} \rightarrow_{pin} [\ [\ ]_a\ ]_{H'''}$,
35. $[\ [\ ]_a\ ]_{H'''} \rightarrow_{exo} [\ ]_a$,
36. $[\ [\ ]_L\ ]_{aA} \rightarrow_{exo} [\ ]_{a\#L},\ A \in N_2$,
37. $[\ [\ ]_{\#}\ ]_{\beta} \rightarrow_{exo} [\ ]_{\#\#}, \beta \in V \cup \{\#\}$.

We need to simulate the terminal matrix as well as remove all the auxiliary symbols at the end of a halting configuration. We start with rules 1 and 2. This is followed by rules 23–25, wherein we replace X_l by Z_a and α' by α. The symbol Z_a indicates that the terminal matrix has been simulated. Rules 27–35 assist in the removal of the remaining auxiliary symbols like D_1, D_2, Q etc. giving $[\ [\ [\ ]_L\ ]_w\ ]_\lambda$. This will be clear from the following sequence of configurations:

$$
\begin{array}{l}
[\ [\ [\ ]_{EA}\ [\ ]_{D_1Q}\ [\ ]_{D_2Q}\ [\ ]_L]_X]_\lambda \rightarrow^{1,10} [\ [\ [\ [\ ]_{D_1}]_{Q_1}\ [\ [\ ]_{D_2}]_{Q_1}\ [\ ]_L]_{X_lEA}]_\lambda \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow^{2,11} \\
[\ [\ [\ [\ ]_{D_1}]_{Q_3}\ [\ [\ ]_{D_2}]_{Q_3}\ [\ ]_L]_{X_lH\alpha'}]_\lambda \leftarrow^{23,12} [\ [\ [\ ]_{Ex'}\ [\ ]_{D_1Q_2}\ [\ ]_{D_2Q_2}\ [\ ]_L]_{X_l}]_\lambda \\
\qquad\quad \downarrow^{24,13} \\
[\ [\ [\ ]_{\alpha'H}\ [\ ]_{D_1Q_4}\ [\ ]_{D_2Q_4}\ [\ ]_L]_{Z_a}]_\lambda \rightarrow^{25,14} [\ [\ [\ [\ ]_{D_1}]_{Q_5}\ [\ [\ ]_{D_2}]_{Q_5}\ [\ ]_L]_{Z_a\alpha H}]_\lambda \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow^{26,15} \\
[\ [\ [\ ]_H[\ [\ ]_{D_2}]_{Q_1}[\ ]_L]_{aH'D_1}]_\lambda \leftarrow^{27,10} [\ [\ [\ ]_H[\ ]_{D_1Q}[\ ]_{D_2Q}\ [\ ]_L]_{aH'}]_\lambda \\
\qquad\qquad\quad \downarrow^{28,11} \\
[\ [\ [\ ]_{D_2Q_2}[\ ]_L]_{aH'D_1}]_\lambda \rightarrow^{29,12} [\ [\ [\ ]_L]_{aH'D_1D_2}]_\lambda \rightarrow^{30} [\ [\ [\ ]_{D'}[\ ]_L]_{H'D_2}]_\lambda \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow^{31} \\
[\ [\ [\ ]_L]_{aH'''}]_\lambda \leftarrow^{33} [\ [\ [\ ]_{aD'}[\ ]_L]_{H''}]_\lambda \leftarrow^{32} [\ [\ [\ ]_L]_{aH'D'}]_\lambda \\
\qquad\quad \downarrow^{34} \\
[\ [\ [\ ]_L\ [\ ]_a\ ]_{H'''}]_\lambda \rightarrow^{35} [\ [\ [\ ]_L\ ]_a\ ]_\lambda.
\end{array}
$$

Rules 36, 37 should be used if any $A \in N_2$ exists after simulation of a terminal matrix, leading to an infinite computation. It is enough if we check the applicability of rule 36, after applying all other rules (after rule 35). There may be cases where rule 36 is applicable before, but we can avoid the use of rule 36 till the very end, thereby getting a correct result, if one exists, by assuming that all symbols of $N_2 \cup T$ are stored in the inner membrane containing E (this assumption is fair since the pino rules distribute symbols non-deterministically). Note also the fact that when E (and all other symbols part of the inner membrane are unified with the outer membrane, then in all scenarios previously, there are other rules applicable, like rules 2, 4, 6, 17), and hence we can say that rule 36 need be applied only after rule 35. □

6.2. The PHE calculus

Now we examine the effect of the phago–exo operations.

Theorem 6.2. *$PsRE = PsOP_m(phago(r), exo(s))$, for all $m \geq 9, r \geq 5, s \geq 2$, and $PsRE = PsOP_m(phago(r), exo(s))$, for all $m \geq 9, r \geq 4, s \geq 3$.*

Proof. As in Theorem 6.1, we shall only prove the inclusion $PsRE \subseteq PsOP_9(phago(5), exo(2))$, using a matrix grammar G in the strong binary normal form. The second characterization of RE using $phago(4), exo(3)$ can be obtained by a slight variation of rules, which we will mention towards the end of the proof.

Construct a P system

$$\Pi = (V, [\ [\ ]_{XF}\ [\ ]_{AE}\ [\ [\ ]_L\ [\ ]_I\ ]_{KM}\ [\ ]_N\ ]_\lambda, R)$$

with alphabet

$$V = \{X, X_l, X_l^{(i)}, X_l^{(i)'}, \langle X \rangle \mid X \in N_1, 1 \leq l \leq n_1 + n_2, 1 \leq i \leq 2\}$$

$\cup\ \{\alpha, \alpha', \alpha'' \mid \alpha \in N_2 \cup T\}$

$\cup\ \{E, E', E'', F, F', F'', I, K, K', L, M, N, f, f'\} \cup \{E_l \mid 1 \leq l \leq n_1\}$.

The initial configuration is $[\ [\ ]_{XF}\ [\ ]_{AE}\ [\ [\ ]_L\ [\ ]_I\ ]_{KM}\ [\ ]_N\ ]_\lambda$, where $X \in N_1$, $A \in N_2$ correspond to the initial matrix $S \to XA$.

Proof idea: The membranes labeled XF, AE are used in the simulation of a type 2 matrix. The membrane labeled KM is used (i) to induce an infinite computation if a symbol $A \in N_2$ does not occur during a type-2 matrix simulation, and (ii) to check for the appearance of symbols $B^{(j)}$ during simulation of a type-3 matrix. The membranes labeled L, I, N are used at the end, after simulation of a terminal matrix, to remove all auxiliary symbols, and to check if any more symbols $A \in N_2$ remain.

The rules R are as follows:

(1) For each non-terminal matrix $m_l : (X \to Y, A \to x)$, $X, Y \in N_1$, $A \in N_2$, $x \in (N_2 \cup T)^*$, with $1 \leq l \leq n_1$, and $x \neq \lambda$, we consider the rules:

1. $[\ ]_E\ [\ ]_{XF} \to_{ph} [\ [\ [\ ]_E\ ]_{X_l}\ ]_F$,
2. $[\ [\ ]_{X_l}\ ]_F \to_{exo} [\ ]_{X_lF}$,
3. $[\ ]_{X_l}\ [\ ]_{AE} \to_{ph} [\ [\ [\ ]_{X_l}\ ]_{x'}\ ]_E$,
   (If $m_l : (X \to Y, A \to \alpha_1\alpha_2)$, then, $x' = \alpha_1'\alpha_2$ or $\alpha_1\alpha_2'$,
   and if $m_l : (X \to Y, A \to \alpha_1)$, then $x' = \alpha_1'$.)
4. $[\ [\ ]_{\alpha'}\ ]_E \to_{exo} [\ ]_{\alpha'E'}$, $\alpha \in N_2 \cup T$,
5. $[\ ]_{E'}\ [\ ]_{X_lF} \to_{ph} [\ [\ [\ ]_{E'}\ ]_Y\ ]_F$,
6. $[\ [\ ]_Y\ ]_F \to_{exo} [\ ]_{YF}$,
7. $[\ ]_Y\ [\ ]_{\alpha'E'} \to_{ph} [\ [\ [\ ]_Y\ ]_\alpha\ ]_{E'}$,
8. $[\ [\ ]_\alpha\ ]_{E'} \to_{exo} [\ ]_{\alpha E}$,
9. $[\ ]_K\ [\ ]_{X_lF} \to_{ph} [\ [\ [\ ]_K\ ]_\#\ ]_F$,
10. $[\ [\ ]_\#\ ]_\beta \to_{exo} [\ ]_{\#\#}$, $\beta \in V \cup \{\#\}$,
11. $[\ ]_\beta\ [\ ]_{\#\alpha} \to_{ph} [\ [\ [\ ]_\beta\ ]_\#\ ]_\#$, $\beta, \alpha \in V \cup \{\#\}$.

(2) If $x = \lambda$, i.e, if $m_l : (X \to Y, A \to \lambda)$ then we have the following rules: Apply 1, 2 as above. Then consider the following rules:

12. $[\ ]_{X_l}\ [\ ]_{AE} \to_{ph} [\ [\ [\ ]_{X_l}\ ]_A\ ]_{E''E_l}$,
13. $[\ [\ ]_A\ ]_{E_l} \to_{exo} [\ ]_{E_l}$,
14. $[\ ]_{E_l}\ [\ ]_{X_lF} \to_{ph} [\ [\ [\ ]_{E_l}\ ]_Y\ ]_F$,
16. $[\ ]_{YF}\ [\ ]_{E_lE''} \to_{ph} [\ [\ [\ ]_{YF}\ ]_{E_l}\ ]_E$,
17. $[\ [\ ]_{E_l}\ ]_E \to_{exo} [\ ]_E$.

We begin the simulation with rule 1, replacing X by X_l. The membrane labeled AE enters the membrane labeled X_l and rule 2 brings it out. This is followed by replacing A by x' using rule 3. Now, to prevent any more A's from being replaced, rule 4 replaces E by E'. This is followed by replacing x' by x and X_l by Y to finish the simulation correctly. This is done by first replacing X_l by Y using rule 5, followed by rules 6,7 which replace α' by α, and finally rule 8 replaces E' by E. We depict a simulation by the following sequence of configurations:

$$[\ [\ ]_{XF}\ [\ ]_{AE}\ [\ [\ ]_L\ [\ ]_I]_{MK}\ [\ ]_N]_\lambda \to^1 [\ [\ [\ [\ ]_{AE}]_{X_l}]_F\ [\ [\ ]_L\ [\ ]_I]_{MK}\ [\ ]_N]_\lambda$$
$$\downarrow^2$$
$$[\ [\ [\ [\ ]_{X_l}]_{x'}]_E\ [\ [\ ]_L\ [\ ]_I]_{MK}\ [\ ]_N]_\lambda \leftarrow^3 [\ [\ ]_{X_lF}\ [\ ]_{AE}\ [\ [\ ]_L\ [\ ]_I]_{MK}\ [\ ]_N]_\lambda$$
$$\downarrow^4$$
$$[\ [\ ]_{X_lF}\ [\ ]_{\alpha'E'}\ [\ [\ ]_L\ [\ ]_I\ ]_{MK}\ [\ ]_N]_\lambda \to^5 [\ [\ [\ [\ ]_{E'}]_Y]_F\ [\ [\ ]_L\ [\ ]_I]_{MK}\ [\ ]_N]_\lambda$$
$$\downarrow^6$$

$$[\,[\,[\,[\,]_{YF}]_{\alpha}]_{E'}\,[\,[\,]_L\,[\,]_I]_{MK}\,[\,]_N]_{\lambda} \leftarrow^{7} [\,[\,]_{YF}\,[\,]_{\alpha' E'}\,[\,[\,]_I]_{MK}\,[\,]_N]_{\lambda}$$
$$\downarrow^{8}$$
$$[\,[\,]_{YF}\,[\,]_{\alpha E}\,[\,[\,]_L\,[\,]_I]_{MK}\,[\,]_N]_{\lambda}.$$

Note that rule 9 is an exception, which can be used in a scenario when the corresponding $A \in N_2$ does not occur. This leads to an infinite computation using rules 10, 11.

(3) For each type 3 matrix $m'_l : (X \to Y, B^{(j)} \to \#)$, $n_1 + 1 \le l \le n_2$, we have the following rules:

18. $[\,]_E\,[\,]_{XF} \to_{ph} [\,[\,[\,]_E\,]_{X_l^{(j)}}\,]_F$,
19. $[\,[\,]_{X_l^{(j)}}\,]_F \to_{exo} [\,]_{X_l^{(j)} F'}$,
20. $[\,]_E\,[\,]_{X_l^{(j)} F'} \to_{ph} [\,[\,[\,]_E\,]_{X_l^{(j)'}}\,]_{F'}$,
21. $[\,[\,]_{B^{(j)}}\,]_{X_l^{(j)'}} \to_{exo} [\,]_{\#B^{(j)}}$,
22. $[\,[\,]_{X_l^{(j)'}}\,]_{F'} \to_{exo} [\,]_{X_l^{(j)''} F'}$,
23. $[\,]_E\,[\,]_{X_l^{(j)''} F'} \to_{ph} [\,[\,[\,]_E\,]_{X_l^{(j)''}}\,]_{F''}$,
24. $[\,[\,]_{B^{(j)}}\,]_{X_l^{(j)''}} \to_{exo} [\,]_{\#B^{(j)}}$,
25. $[\,]_{F''}\,[\,]_{KM} \to_{ph} [\,[\,[\,]_{F''}\,]_{K'}\,]_M$,
26. $[\,[\,]_{F''}\,]_{K'} \to_{exo} [\,]_{F''J}$,
27. $[\,]_{X_l^{(j)''}}\,[\,]_{F''J} \to_{ph} [\,[\,[\,]_{X_l^{(j)''}}\,]_F\,]_{F''}$,
28. $[\,[\,]_{F''}\,]_M \to_{exo} [\,]_{KM}$,
29. $[\,[\,]_{X_l^{(j)''}}\,]_F \to_{exo} [\,]_{FY}$,
30. $[\,[\,]_{X_l^{(j)''}}\,]_{MJ} \to_{ph} [\,[\,[\,]_{X_l^{(j)''}}\,]_{\#}\,]_M$.

The simulation is done as follows: We start with rule 18 replacing X by $X_l^{(j)}$. This is followed by the rule 19, wherein F is replaced by F'. Rule 20 replaces $X_l^{(j)}$ by $X_l^{(j)'}$ and starts the appearance checking process. Rule 21 can be applied now if the corresponding $B^{(j)}$ is present. However, we can non-deterministically choose rule 22 and continue. By this, we replace $X_l^{(j)'}$ by $X_l^{(j)''}$. In rule 23, we replace F' by F''.

The next few steps are crucial and will ensure that if $B^{(j)}$ actually occurs, then the trap symbol will be introduced. For these next few steps, we preserve the structure $[\,[\,]_E]_{X_l^{(j)''}}$ so that rule 24, if applicable, will certainly be applied. In the meantime, we do the following: We use rule 25 so that the entire structure $[\,[\,[\,]_E]_{X_l^{(j)''}}\,]_{F''}$ enters $[\,]_{KM}$ by a phago rule. This is followed by rule 26, which makes $[\,[\,]_E]_{X_l^{(j)''}}$ to be adjacent to $[\,]_{F''J}$, both structures being inside the membrane labeled M. Rule 27 is applicable now, by which we replace J by F and obtain the structure $[\,[\,[\,[\,[\,]_E\,]_{X_l^{(j)''}}\,]_F\,]_{F''}\,]_M\,]_{\lambda}$. Note that rule 28 is also an applicable rule. However, if we use rule 28 over rule 27, we would end up using rule 30, giving an infinite computation.

Following rules 26, 27, we apply the two exo rules 28, 29 in parallel, collapsing the hierarchical membrane structure and getting back to the original structure we started with. It is obvious that during the time we used rules 25–28, rule 24 if applicable, would have been used. We now give the steps involved in a type-3 matrix simulation below:

(a) $[\,[\,]_E\,[\,]_{XF}\,[\,[\,]_L\,[\,]_I\,]_{KM}\,[\,]_N\,]_{\lambda} \downarrow^{18}$
(b) $[\,[\,[\,[\,]_E]_{X_l^{(j)}}\,]_F\,[\,[\,]_L\,[\,]_I\,]_{KM}\,[\,]_N\,]_{\lambda} \downarrow^{19}$
(c) $[\,[\,]_E\,[\,]_{X_l^{(j)} F'}\,[\,[\,]_L\,[\,]_I\,]_{KM}\,[\,]_N\,]_{\lambda} \downarrow^{20}$
(d) $[\,[\,[\,[\,]_E]_{X_l^{(j)'}}\,]_{F'}\,[\,[\,]_L\,[\,]_I\,]_{KM}\,[\,]_N\,]_{\lambda} \downarrow^{22}$
(e) $[\,[\,]_E\,[\,]_{X_l^{(j)''} F'}\,[\,[\,]_L\,[\,]_I\,]_{KM}\,[\,]_N\,]_{\lambda} \downarrow^{23}$
(f) $[\,[\,[\,[\,]_E]_{X_l^{(j)''}}\,]_{F''}\,[\,[\,]_L\,[\,]_I\,]_{KM}\,[\,]_N\,]_{\lambda} \downarrow^{25}$

(g) $[\ [\ [\ [\ [\ [\ ]_E\ ]_{X_l^{(j)''}}\ ]_{F''}\ ]_{K'}\ [\ ]_L\ [\ ]_I\ ]_M\ [\ ]_N\ ]_\lambda \ \downarrow^{26}$

(h) $[\ [\ [\ [\ ]_E\ ]_{X_l^{(j)''}}\ [\ ]_{F''J}\ [\ ]_L\ [\ ]_I\ ]_M\ [\ ]_N\ ]_\lambda \ \downarrow^{27}$

(i) $[\ [\ [\ [\ [\ ]_E\ ]_{X_l^{(j)''}}\ ]_F\ ]_{F''}\ [\ ]_L\ [\ ]_I\ ]_M\ [\ ]_N\ ]_\lambda \ \downarrow^{28,29}$

(j) $[\ [\ ]_E\ [\ ]_{YF}\ [\ [\ ]_L\ [\ ]_I\ ]_{KM}\ [\ ]_N\ ]_\lambda$.

(4) For each terminal matrix $m_l : (X \to a, A \to x), 1 \le l \le n_1, X \in N_1, A \in N_2, a \in T, x \in T^*$, we consider the following rules: (We denote by x'' either $a_1''a_2$ or a_1a_2'' when $x = a_1a_2$, and $x'' = a_1''$ when $x = a_1$, for $A \to x \in m_l, A \in N_2$)

31. $[\ ]_{KM}\ [\ ]_{XF} \to_{ph} [\ [\ [\ ]_{KM}\ ]_F\ ]_a$,
32. $[\ [\ ]_{KM}\ ]_F \to_{exo} [\ ]_{MF}$,
33. $[\ [\ ]_{MF}\ ]_a \to_{exo} [\ ]_{aM}$,
34. $[\ ]_{aM}\ [\ ]_{AE} \to_{ph} [\ [\ [\ ]_{aM}\ ]_E\ ]_{x''}$, if $x \neq \lambda$, or
    $[\ ]_{aM}\ [\ ]_{AE} \to_{ph} [\ [\ [\ ]_{aM}\ ]_E\ ]_{f'}$ if $x = \lambda$.
35. $[\ [\ ]_I\ ]_{aM} \to_{exo} [\ ]_{aI}$,
36. $[\ [\ ]_{AE}\ ]_{\alpha''} \to_{exo} [\ ]_{\#A}$, $\alpha \in T$,
    $[\ [\ ]_{AE}\ ]_{f'} \to_{exo} [\ ]_{\#A}$,
    $[\ [\ ]_E\ ]_A \to_{exo} [\ ]_{\#A}$,
37. $[\ [\ ]_{aI}\ ]_E \to_{exo} [\ ]_{aI}$,
38. $[\ [\ ]_L\ ]_{\alpha''} \to_{exo} [\ ]_{L\alpha}$, $\alpha \in N_2 \cup T$
    $[\ [\ ]_L\ ]_{f'} \to_{exo} [\ ]_L$,
39. $[\ [\ ]_{aI}\ ]_L \to_{exo} [\ ]_{aL}$,
40. $[\ ]_N\ [\ ]_{La} \to_{ph} [\ [\ [\ ]_N\ ]_L\ ]_a$,
41. $[\ [\ ]_N\ ]_L \to_{exo} [\ ]_N$,
42. $[\ [\ ]_N\ ]_a \to_{exo} [\ ]_a$,
43. $[\ ]_E\ [\ ]_{aI} \to_{ph} [\ [\ [\ ]_E\ ]_\#\ ]_a$,
44. $[\ ]_E\ [\ ]_{aM} \to_{ph} [\ [\ [\ ]_E\ ]_\#\ ]_a$.

Termination: Here, we start by making use of the membranes $[\ ]_{XF}$ and $[\ ]_{KM}$. Rule 31 is used, by which we replace X by a. This is followed by rules 32, 33, wherein K, F are eliminated respectively. Rule 32 ejects out the membranes $[\ ]_L, [\ ]_I$ making them adjacent to $[\ ]_{MF}$. Rule 34 now replaces the $A \in N_2$ corresponding to the terminal matrix. Note that rule 35 is also applicable in this step, since both involve the membrane $[\ ]_{aM}$. However, if we use rule 35, we obtain $[\ ]_{aI}\ [\ ]_E$ which leads to an infinite computation (rule 43). After application of rule 34, rules 35, 36 should be applied in parallel, checking if any more $A \in N_2$ are present. If none are present, then we obtain $[\ [\ [\ ]_{aI}\ ]_E\ ]_{x''}$. Next, rule 37 is used to eliminate E, and this makes rule 38 applicable in the next step. This is followed by rules 39–42, removing all auxiliary symbols. Note that while simulating the terminal matrix, if $A \in N_2$ does not occur, we use rule 44. Below, we illustrate the sequence of configurations during the simulation of a terminal matrix.

$$[\ [\ ]_{AE}\ [\ ]_{XF}\ [\ [\ ]_L\ [\ ]_I\ ]_{KM}\ [\ ]_N]_\lambda \to^{31} [\ [\ ]_{AE}\ [\ [\ [\ [\ ]_L\ [\ ]_I]_{KM}]_F]_a\ [\ ]_N]_\lambda$$
$$\downarrow^{32}$$
$$[\ [\ ]_{AE}\ [\ [\ ]_L\ [\ ]_I]_{aM}\ [\ ]_N]_\lambda \leftarrow^{33} [\ [\ ]_{AE}\ [\ [\ ]_{MF}\ [\ ]_L\ [\ ]_I]_a\ [\ ]_N]_\lambda$$
$$\downarrow^{34}$$
$$[\ [\ [\ [\ [\ ]_L\ [\ ]_I]_{aM}]_E]_{x''}\ [\ ]_N]_\lambda \to^{35,-} [\ [\ [\ [\ [\ ]_L]_{aI}]_E]_{x''}\ [\ ]_N]_\lambda$$
$$\downarrow^{37}$$
$$[\ [\ ]_{aL}\ [\ ]_N\ ]_\lambda \leftarrow^{39} [\ [\ [\ ]_{aI}\ ]_{Lx}\ [\ ]_N\ ]_\lambda \leftarrow^{38} [\ [\ [\ ]_{aI}\ [\ ]_L\ ]_{x''}\ [\ ]_N\ ]_\lambda$$
$$\downarrow^{40}$$
$$[\ [\ [\ [\ ]_N\ ]_L\ ]_a\ ]_\lambda \to^{41} [\ [\ [\ ]_N\ ]_a\ ]_\lambda \to^{42} [\ [\ ]_a\ ]_\lambda.$$

Note: To obtain the characterization $PsRE = PsOP_9(phago_4, exo_3)$, observe that in the above, rule 34 is the only phago rule having weight 5. We can replace this by a rule $[\ ]_{aM}\ [\ ]_{AE} \rightarrow_{ph} [\ [\ [\ ]_{aM}\ ]_E\ ]_{\langle x\rangle}$, where $\langle x\rangle \in V$, for all $A \rightarrow x$. Now this phago rule has weight 4. Then, correspondingly, instead of rule 38 replacing α'', we will have to replace $\langle x\rangle$ by a rule $[\ [\ ]_L\ ]_{\langle x\rangle} \rightarrow_{exo} [\ ]_{Lx}$ of weight 3. □

6.3. The BUM calculus

The bud–mate interplay is examined below.

Theorem 6.3. $PsRE = PsOP_m(bud(r), mate(s))$, $\forall m \geq 7, r \geq 5, s \geq 3$.

Proof. As before, we shall only prove the inclusion $PsRE \subseteq PsOP_7(bud(5), mate(3))$, using a matrix grammar G in strong binary normal form. Construct a P system

$$\Pi = (V, [\ [\ [\ [\ ]_Q\ ]_{DR}\ [\ ]_P\ ]_{XFA}\ ]_\lambda, R)$$

with alphabet

$$V = \{X, X_l, X_l^{(j)}, \langle X\rangle \mid X \in N_1, 1 \leq l \leq n_2, j = 1, 2\}$$
$$\cup\ \{\alpha, \alpha' \mid \alpha \in N_2 \cup T\} \cup \{f, D, D_1, D_2, D_3, F, Q, R, P\}.$$

The initial configuration is $[\ [\ [\ [\ ]_Q\ ]_{DR}\ [\ ]_P\ ]_{XFA}\ ]_\lambda$ where $X \in N_1$, $A \in N_2$ and $(S \rightarrow XA)$ is the initial matrix of G.

Proof idea: The membrane labeled XFA in conjunction with the membrane labeled P simulate a type-2 matrix. The membranes labeled DR and XFA are used to begin simulation of a type-3 matrix. The membrane labeled P tests for the presence of the symbol $B^{(j)}$ during simulation. Finally, the membranes labeled DR and P, mate, eliminating the symbol R, during simulation of the terminal matrix. This new membrane containing DP is then used to check for the presence of any $A \in N_2$ in the upper membrane. If none exists, and the terminal matrix is simulated successfully, we stop with $[\ [\ [\ [\ ]_Q]_{DP}]_w]_\lambda$, where $Ps\ w$ is the output.

The rules are as follows:

(1) For each non-terminal matrix $m_l : (X \rightarrow Y, A \rightarrow x)$, $1 \leq l \leq n_1$, $X, Y \in N_1$, $A \in N_2$, $x \in (N_2 \cup T)^*$, consider the following rules:

1. $[\ [\ ]_P\ ]_{XAF} \rightarrow_{bud} [\ [\ ]_P\ ]_{X_l}\ [\ ]_{AF}$,
1′. $[\ [\ ]_P\ ]_{XF} \rightarrow_{bud} [\ [\ ]_P\ ]_{\#}\ [\ ]_F$,
2. $[\ ]_{X_l}\ [\ ]_A \rightarrow_{mate} [\ ]_{X_l f}$, if $x = \lambda$,
   $[\ ]_{X_l}\ [\ ]_A \rightarrow_{mate} [\ ]_{X_l x'}$, if $x \neq \lambda$,
   (If $m_l : (X \rightarrow Y, A \rightarrow \alpha_1\alpha_2)$, then, $x' = \alpha_1'\alpha_2$ or $\alpha_1\alpha_2'$,
   and if $m_l : (X \rightarrow Y, A \rightarrow \alpha_1)$, then $x' = \alpha_1'$.)
3. $[\ [\ ]_P\ ]_{X_l\alpha' F} \rightarrow_{bud} [\ [\ ]_P\ ]_Y\ [\ ]_{\alpha' F}$, $\alpha \in N_2 \cup T$,
   $[\ [\ ]_P\ ]_{X_l f F} \rightarrow_{bud} [\ [\ ]_P\ ]_Y\ [\ ]_{fF}$,
4. $[\ ]_Y\ [\ ]_{\alpha'} \rightarrow_{mate} [\ ]_{Y\alpha}$, $\alpha \in N_2 \cup T$,
   $[\ ]_Y\ [\ ]_f \rightarrow_{mate} [\ ]_Y$.

Consider the following rules as well, which happen as part of type 2/3/4 matrix simulation:

5. $[\ [\ ]_Q\ ]_{DR} \rightarrow_{bud} [\ [\ ]_Q\ ]_{D_1}\ [\ ]_R$,
6. $[\ ]_{D_1}\ [\ ]_R \rightarrow_{mate} [\ ]_{D_2R}$,
7. $[\ [\ ]_Q\ ]_{D_2R} \rightarrow_{bud} [\ [\ ]_Q\ ]_{D_3}\ [\ ]_R$,
8. $[\ ]_{D_3}\ [\ ]_R \rightarrow_{mate} [\ ]_{DR}$.

We have the rules 1 through 4 mimicking the simulation as well as rules 5 through 8 that act in parallel. Rules 5 through 8 play a major role while simulating type 3 matrices, so at the moment, it is enough to concentrate on

rules 1–4. The simulation is straightforward, replacing X by X_l, followed by replacing the corresponding symbol $A \in N_2$ by x' or f, followed by finally replacing X_l and x' or f, as the case may be. Note that in the case the symbol $A \in N_2$ is not present during simulation, the rule $1'$ is used, inducing an infinite computation. The sequence of configurations below give a better idea of the simulation:

$$[\,[\,[\,[\,]_Q\,]_{DR}\,[\,]_P\,]_{XFA}\,]_\lambda \rightarrow^{1,5} [\,[\,[\,]_P]_{X_l}\,[\,[\,[\,]_Q]_{D_1}\,[\,]_R\,]_{AF}]_\lambda$$
$$\downarrow^{2,6}$$
$$[\,[\,[\,]_P]_Y\,[\,[\,[\,]_Q]_{D_3}\,[\,]_R]_{\alpha' F}]_\lambda \leftarrow^{3,7} [\,[\,[\,[\,]_Q]_{D_2 R}\,[\,]_P]_{X_l F x'}]_\lambda$$
$$\downarrow^{4,8}$$
$$[\,[\,[\,[\,]_Q]_{DR}\,[\,]_P]_{YFx}]_\lambda.$$

(2) For each matrix $m_l : (X \rightarrow Y, B^{(j)} \rightarrow \#)$, $n_1 + 1 \leq l \leq n_2$, we have the following rules:

9. $[\,[\,]_D\,]_{XF} \rightarrow_{bud} [\,[\,]_D\,]_{X_l^{(j)}}\,[\,]_F$,
10. $[\,]_{X_l^{(j)}}\,[\,]_F \rightarrow_{mate} [\,]_{X_l^{(j)} F}$,
11. $[\,[\,]_P\,]_{X_l^{(j)} B^{(j)}} \rightarrow_{bud} [\,[\,]_P\,]_\#\,[\,]_{B^{(j)}}$,
12. $[\,]_\#\,[\,]_\beta \rightarrow_{mate} [\,]_{\#\#}$, $\beta \in V \cup \{\#\}$,
13. $[\,[\,]_P\,]_{\#\beta} \rightarrow_{bud} [\,[\,]_P\,]_\#\,[\,]_\#$, $\beta \in V \cup \{\#\}$
14. $[\,[\,]_{D_3}\,]_{X_l^{(j)} F} \rightarrow_{bud} [\,[\,]_{D_3}\,]_{\langle Y\rangle}\,[\,]_F$,
15. $[\,]_{\langle Y\rangle}\,[\,]_F \rightarrow_{mate} [\,]_{YF}$.

To start the simulation, we need the membranes labeled D as well as $X \in N_1$. So, rules 5, 9 are applicable. If rule 5 is chosen, then we need to wait till rule 8 is completed, in order to apply 9. Let us assume that we start with rule 9.

$$[\,[\,[\,[\,]_Q]_{DR}\,[\,]_P]_{XF}]_\lambda \rightarrow^{9} [\,[\,[\,[\,]_Q]_{DR}]_{X_l^{(j)}}\,[\,[\,]_P]_F]_\lambda$$
$$\downarrow^{5,10}$$
$$[\,[\,[\,[\,]_Q]_{D_2 R}\,[\,]_P]_{X_l^{(j)} F}]_\lambda \leftarrow^{6} [\,[\,[\,[\,]_Q]_{D_1}[\,]_R\,[\,]_P]_{X_l^{(j)} F}]_\lambda$$
$$\downarrow^{7}$$
$$[\,[\,[\,[\,]_Q]_{D_3}[\,]_R\,[\,]_P]_{X_l^{(j)} F}]_\lambda \rightarrow^{14} [\,[\,[\,[\,]_Q]_{D_3}]_{\langle Y\rangle}\,[\,[\,]_P\,[\,]_R]_F]_\lambda$$
$$\downarrow^{15}$$
$$[\,[\,[\,[\,]_Q\,]_{D_3}\,[\,]_R\,[\,]_P\,]_{YF}\,]_\lambda.$$

Rules 10, 5 can be applied in parallel: one mates the outer membranes whereas the other one is a bud with respect to D, Q. In the next step, rule 11 if applicable, will be applied, since there are no other rules applicable to $X_l^{(j)}$, P. In parallel, we apply rule 6, which mates the two membranes labeled D_1, R. In the next step, again, we use rule 7, replacing D_2 by D_3. With D_3, we can apply rule 14, provided $X_l^{(j)}$ is present (i.e, if $B^{(j)}$ is absent). We also have a choice of using rule 8 for D_3, but this will mean that we apply rules 5–7 again before applying rule 14. Clearly, the computation does not progress with respect to $X_l^{(j)}$ unless rule 14 is used. Rule 15 completes the simulation, replacing $\langle Y\rangle$ by Y.

Note that in the above configurations, we have non-deterministically chosen to direct the inner membranes into different components while using bud rules. It is interesting to note that while this does not affect the results, in the last two configurations above, had we chosen to put both the membranes labeled R and D_3 inside the membrane labeled $\langle Y\rangle$, we could have used rules 8 and 15, in parallel. The illustration given above should make the computation clear while simulating a type-3 matrix.

(3) For each terminal matrix $m_l : (X \rightarrow a, A \rightarrow x)$, $X \in N_1$, $A \in N_2$, $a \in T$, $x \in T^*$, $1 \leq l \leq n_1$,
First apply rules 1–4, replacing Y by Z_a. Then consider the following:

16. $[\,]_{DR}\,[\,]_P \rightarrow_{mate} [\,]_{DP}$,

17. $[\,[\,]_{DP}\,]_{Z_aF} \rightarrow_{bud} [\,[\,]_{DP}\,]_{Z_a}\,[\,]_{aF'}$,
18. $[\,]_{Z_a}\,[\,]_{F'} \rightarrow_{mate} [\,]_{F'}$,
19. $[\,[\,]_{DP}\,]_{aF'} \rightarrow_{bud} [\,[\,]_{DP}\,]_a\,[\,]_{F''}$,
20. $[\,]_a\,[\,]_{F''} \rightarrow_{mate} [\,]_a$,
21. $[\,[\,]_{DP}\,]_{\beta F} \rightarrow_{bud} [\,[\,]_{DP}\,]_{\#}\,[\,]_F,\ \beta \neq Z_a$,
22. $[\,[\,]_{DP}\,]_{a\beta} \rightarrow_{bud} [\,[\,]_{DP}\,]_{\#}\,[\,]_\beta, \beta \in N_2$.

Termination: Use rule 16 which mates the membranes labeled DR and P, some time after using rules 1–4. This prevents any further application of rules 5 through 8.

$$
\begin{array}{ccc}
[\,[\,[\,[\,]_Q]_{DR}\,[\,]_P]_{XF}]_\lambda & \rightarrow^{1\cdots4} & [\,[\,[\,[\,]_Q]_{DR}\,[\,]_P]_{Z_aF}]_\lambda \\
 & & \downarrow^{16} \\
[\,[\,[\,[\,]_Q]_{DP}]_{Z_a}\,[\,]_{aF'}]_\lambda & \leftarrow^{17} & [\,[\,[\,[\,]_Q]_{DP}]_{Z_aF}]_\lambda \\
\downarrow^{18} & & \\
[\,[\,[\,[\,]_Q]_{DP}]_{aF'}]_\lambda & \rightarrow^{19} & [\,[\,[\,[\,]_Q]_{DP}]_a\,[\,]_{F''}]_\lambda \\
 & & \downarrow^{20} \\
 & & [\,[\,[\,[\,]_Q]_{DP}]_a]_\lambda .
\end{array}
$$

Next use rule 17, replacing F by aF' (we had Z_aF earlier, and now F is replaced by aF'). Rule 18 follows, wherein F' helps to eliminate Z_a. Now we need to eliminate F'. Rule 19 first replaces F' by F''. This is followed by rule 20, wherein F'' is eliminated as well. Thus, we obtain a configuration $[\,[\,[\,[\,]_Q\,]_{DP}\,]_w\,]_\lambda$.

Note that in the above sequence of configurations, we have chosen to use rule 16 after rules 1–4 (5–8), but in general, it does no harm to use this anytime during the last simulation when both membranes $[\,]_{DR}$, $[\,]_P$ are free. However, using this any time before the terminal matrix is simulated will lead to no output (Note the use of D_3 in rule 14 to complete a type-3 matrix simulation. In the case R is removed, there will be no D_3, and in such a scenario, rule 21 will be used, leading to an infinite computation). □

6.4. The MAD calculus

Finally, we improve the universality result of mate–drip operations, as given in [4].

Theorem 6.4. *$PsRE = PsOP_m(mate(r), drip(s))$, for all $m \geq 4, r \geq 3, s \geq 3$.*

Proof. We prove only $PsRE \subseteq PsOP_4(mate(3), drip(3))$. Let G be a matrix grammar with appearance checking in the strong binary normal form. Construct the P system

$$\Pi = (V, [\,[\,]_{XAE}\,[\,]_d\,[\,]_e\,]_\lambda, R)$$

with alphabet

$$
\begin{aligned}
V = {} & \{X, X', X'', \langle X\rangle, \langle\langle X\rangle\rangle, X_l, X'_l \mid X \in N_1, 1 \leq l \leq n_1\} \\
\cup {} & \{X_l^{(j)} \mid X \in N_1, n_1 + l \leq n_2\} \cup \{\alpha, \alpha' \mid \alpha \in N_2 \cup T\} \\
\cup {} & \{d, d_1, d_2, d_3, d_4, e, e', e'', e''', \langle e\rangle, f, f', g\} \cup \{Z_a, Z'_a, Z''_a \mid a \in T\}.
\end{aligned}
$$

The initial configuration is $[\,[\,]_{XAE}\,[\,]_d\,[\,]_e\,]_\lambda$ where $X \in N_1$, $A \in N_2$, and $(S \rightarrow XA)$ is the initial matrix of G.
Proof idea: The membrane labeled XAE contains all the symbols of $N_1 \cup N_2 \cup T$ during a simulation. The two membranes labeled d, e assist the membrane labeled XAE in simulation. For simulation of a type-2 matrix, the membrane XAE drips initially, replacing X and then mates, by replacing a single $A \in N_2$. The d, e membranes help in completing the simulation successfully. Similarly, a type-3 matrix simulation is also initiated by the membrane labeled XE, and the membranes d, e help in the appearance checking process. Finally, to terminate, a couple of mate rules are used, eliminating all auxiliary symbols d, e, E etc. in the process, and finally obtaining $[\,[\,]_w]_\lambda$.

The rules R are as follows:

(1) For each non-terminal matrix $m_l : (X \to Y, A \to x)$, $1 \leq l \leq n_1$, consider the following rules:

1. $[\ ]_{XA} \to_{drip} [\ ]_{X_l} [\ ]_A$,
2. $[\ ]_A [\ ]_{X_l} \to_{mate} [\ ]_{fX_l}$, if $x = \lambda$,
   $[\ ]_A [\ ]_{X_l} \to_{mate} [\ ]_{x'X_l}$, if $x \neq \lambda$,
   (If $m_l : (X \to Y, A \to \alpha_1\alpha_2)$, then, $x' = \alpha_1'\alpha_2$ or $\alpha_1\alpha_2'$,
   and if $m_l : (X \to Y, A \to \alpha_1)$, then $x' = \alpha_1'$.)
3. $[\ ]_{X_l\alpha'} \to_{drip} [\ ]_{X_l'} [\ ]_{\alpha'}$, $\alpha \in N_2 \cup T$,
   $[\ ]_{X_l f} \to_{drip} [\ ]_{X_l'} [\ ]_f$,
4. $[\ ]_{\alpha'} [\ ]_{X_l'} \to_{mate} [\ ]_{\alpha g X_l'}$,
   $[\ ]_f [\ ]_{X_l'} \to_{mate} [\ ]_{gX_l'}$,
5. $[\ ]_{X_l'g} [\ ]_{e'} \to_{mate} [\ ]_{Y'ge'}$,
6. $[\ ]_{Y'ge'} \to_{drip} [\ ]_{Y'} [\ ]_g$,
7. $[\ ]_{Y'} [\ ]_g \to_{mate} [\ ]_{Y''g}$,
8. $[\ ]_{Y''gE} \to_{drip} [\ ]_{Y''} [\ ]_E$,
9. $[\ ]_{Y''} [\ ]_E \to_{mate} [\ ]_{YE}$,
10. $[\ ]_d [\ ]_e \to_{mate} [\ ]_{d_1e}$,
11. $[\ ]_{d_1e} \to_{drip} [\ ]_{d_1} [\ ]_e$,
12. $[\ ]_{d_1} [\ ]_e \to_{mate} [\ ]_{d_1e''}$,
13. $[\ ]_{d_1e''} \to_{drip} [\ ]_{d_1e'''} [\ ]_{e'}$,
14. $[\ ]_{d_1e'''} \to_{drip} [\ ]_{d_2} [\ ]_{e'''}$,
15. $[\ ]_{d_2} [\ ]_{e'''} \to_{mate} [\ ]_{d_2e}$,
16. $[\ ]_{d_2e} \to_{drip} [\ ]_{d_3} [\ ]_e$,
17. $[\ ]_{d_3} [\ ]_e \to_{mate} [\ ]_{d_4e}$,
18. $[\ ]_{d_4e} \to_{drip} [\ ]_d [\ ]_e$.

Starting with $[\ [\ ]_{XAE} [\ ]_d [\ ]_e\ ]_\lambda$, we first apply rules 1, 8 in parallel. The computation follows deterministically: Whenever we have an $X \in N_1$ and its corresponding $A \in N_2$ together, we replace X by X_l through a drip operation. This is followed by a mate, wherein the A is rewritten by x'. In the next step, we need to rename X_l to prevent any more A's being replaced by x' as in rule 2. This is done by rule 3, and rule 4 changes α' back to α. The new symbol g introduced helps to replace X_l' by Y', which is in subsequent steps replaced by Y.

$$
\begin{array}{l}
[\ [\ ]_{XAE} [\ ]_d [\ ]_e\ ]_\lambda \ \to^{1,10}\ [\ [\ ]_{X_l} [\ ]_A [\ ]_{d_1e}\ ]_\lambda \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow^{2,11} \\
[\ [\ ]_{X_l'} [\ ]_{x'} [\ ]_{d_1e''}\ ]_\lambda \ \leftarrow^{3,12}\ [\ [\ ]_{x'X_lE} [\ ]_{d_1} [\ ]_e\ ]_\lambda \\
\quad \downarrow^{4,13} \\
[\ [\ ]_{xgX_l'E} [\ ]_{d_1e'''} [\ ]_{e'}\ ]_\lambda \ \to^{5,14}\ [\ [\ ]_{xY'gEe'} [\ ]_{d_2} [\ ]_{e'''}\ ]_\lambda \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow^{6,15} \\
[\ [\ ]_{Y''gE} [\ ]_{d_3} [\ ]_e\ ]_\lambda \ \leftarrow^{7,16}\ [\ [\ ]_{Y'} [\ ]_g [\ ]_{d_2e}\ ]_\lambda \\
\quad \downarrow^{8,17} \\
[\ [\ ]_{Y''} [\ ]_E [\ ]_{d_4e}\ ]_\lambda \ \to^{9,18}\ [\ [\ ]_{YE} [\ ]_d [\ ]_e\ ]_\lambda .
\end{array}
$$

The membranes labeled with d and e in the initial configuration, evolve in parallel during the simulation and help in the process as can be seen above from the sequence of configurations.

(2) For every matrix $m_l : (X \to Y, B^{(j)} \to \#)$, $n_1 + 1 \leq l \leq n_2$, consider the following rules:

19. $[\ ]_{XE} \ \to_{drip} \ [\ ]_{X_l^{(j)}}[\ ]_E$,
20. $[\ ]_{X_l^{(j)}}[\ ]_E \ \to_{mate} \ [\ ]_{X_l^{(j)}E_j}$,
21. $[\ ]_{X_l^{(j)}B^{(j)}} \ \to_{drip} \ [\ ]_{\#\#}[\ ]_{B^{(j)}}$,
22. $[\ ]_{X_l^{(j)}E_j}[\ ]_{e'} \ \to_{mate} \ [\ ]_{X_l^{(j)}Ee'}$,
23. $[\ ]_{X_l^{(j)}e'} \ \to_{drip} \ [\ ]_{\langle Y\rangle}[\ ]_{e'}$,
24. $[\ ]_{\langle Y\rangle}[\ ]_{e'} \ \to_{mate} \ [\ ]_{\langle\langle Y\rangle\rangle e'}$,
25. $[\ ]_{Ee'\langle\langle Y\rangle\rangle} \ \to_{drip} \ [\ ]_E[\ ]_{\langle\langle Y\rangle\rangle}$,
26. $[\ ]_{\langle\langle Y\rangle\rangle}[\ ]_E \ \to_{mate} \ [\ ]_{YE}$.

We start with a configuration $[\ [\ ]_{XE}[\ ]_d[\ ]_e\ ]_\lambda$ and first apply rules 19, 10. As in the above case, the evolution of the membranes labeled d, e happen hand in hand with the simulation of the type 3 matrix. We use a drip operation in 19 replacing X by $X_l^{(j)}$. This is followed by 20 or 21. Rule 21 will be applied only if the corresponding symbol $B^{(j)}$ is present, otherwise, rule 20 is applied. The next applicable rule is rule 22, for which we need the symbol e' will is generated at the 4th step (starting from rule 10).

$$
\begin{array}{rl}
[\ [\ ]_{XE}[\ ]_d[\ ]_e\ ]_\lambda & \to^{19,10} \ [\ [\ ]_{X_l^{(j)}}[\ ]_E[\ ]_{d_1e}\ ]_\lambda \\
 & \qquad\qquad\qquad\qquad\qquad\qquad \downarrow^{20,11} \\
[\ [\ ]_{X_l^{(j)}E_j}[\ ]_{d_1e''}\ ]_\lambda & \leftarrow^{12} \ [\ [\ ]_{X_l^{(j)}E_j}[\ ]_{d_1}[\ ]_e\ ]_\lambda \\
\quad \downarrow^{13} & \\
[\ [\ ]_{X_l^{(j)}E_j}[\ ]_{d_1e'''}[\ ]_{e'}\ ]_\lambda & \to^{22,14} \ [\ [\ ]_{X_l^{(j)}Ee'}[\ ]_{d_2}[\ ]_{e'''}\ ]_\lambda \\
 & \qquad\qquad\qquad\qquad\qquad\qquad \downarrow^{23,15} \\
[\ [\ ]_{\langle\langle Y\rangle\rangle e'E}[\ ]_{d_3}[\ ]_e\ ]_\lambda & \leftarrow^{24,16} \ [\ [\ ]_{\langle Y\rangle}[\ ]_{e'}[\ ]_{d_2e}\ ]_\lambda \\
\quad \downarrow^{25,17} & \\
[\ [\ ]_{\langle\langle Y\rangle\rangle}[\ ]_E[\ ]_{d_4e}\ ]_\lambda & \to^{26,18} \ [\ [\ ]_{YE}[\ ]_d[\ ]_e\ ]_\lambda .
\end{array}
$$

That means, the symbol $X_l^{(j)}$ has at least one idle step waiting for e'. This waiting time ensures that if $B_l^{(j)}$ was present, rule 21 would have been certainly used. Thus, rule 22 would be applicable only if $B_l^{(j)}$ was not present. Now, we replace $X_l^{(j)}$ by $\langle Y\rangle$ and in subsequent steps, obtain Y, completing the simulation. Note that, if rule 21 were applied, we would have got a never ending computation due to rule 37.

(3) For each terminal matrix $m_l : (X \to a, A \to x)$, $1 \leq l \leq n_1$, consider the rules: Use rules 1 through 9, replacing Y', Y'' by Z'_a, Z''_a and Y by Z_a.

27. $[\ ]_{Z_aA}[\ ]_{e'} \ \to_{mate} \ [\ ]_{Ae'\#}$, $A \in N_2$,
28. $[\ ]_{Z_aE}[\ ]_{d_3} \ \to_{mate} \ [\ ]_{Z'_aEd_3}$,
29. $[\ ]_{Z'_aE}[\ ]_{e'} \ \to_{mate} \ [\ ]_{Z'_ae'}$,
30. $[\ ]_{Z'_ae'd_3} \ \to_{drip} \ [\ ]_{Z''_a}[\ ]_{e'd_3}$,
31. $[\ ]_{e'd_3}[\ ]_e \ \to_{mate} \ [\ ]_{d_3e}$,
32. $[\ ]_{Z''_a}[\ ]_{d_3e} \to_{mate} \ [\ ]_{Z''_ae}$,
33. $[\ ]_{Z''_ae} \ \to_{drip} \ [\ ]_{Z''_a}[\ ]_{\langle e\rangle}$,
34. $[\ ]_{Z''_a}[\ ]_{\langle e\rangle} \ \to_{mate} \ [\ ]_{af'\langle e\rangle}$,
35. $[\ ]_{af'\langle e\rangle} \ \to_{drip} \ [\ ]_a[\ ]_{f'}$,
36. $[\ ]_a[\ ]_{f'} \ \to_{mate} \ [\ ]_a$.

Termination: Now we examine the simulation of a terminal matrix and the subsequent removal of all auxiliary symbols used, $d, e, E \ldots$ so that an output over T^* is obtained. The simulation of a terminal matrix is same as case 1, the only change being that when we stop, we replace Y', Y'' by Z'_a, Z''_a and Y by Z_a. Now, using Z_a, we simplify the membrane structure as well as remove all intermediate symbols.

The configuration to begin with, after obtaining Z_a, is $[\ [\ ]_{Z_a wE}\ [\ ]_d\ [\ ]_e\ ]_\lambda$. Now we continue with rules 10–13 until e' is obtained. Once e' is obtained, we check if there is any $A \in N_2$ along with Z_a, and if yes, rule 27 is used. Otherwise, we wait till d_3 is obtained at the end of rule 15. Using d_3, we rename Z_a to Z'_a. The first thing we do using Z'_a is to mate with e', removing E, getting $Z'_a d_3 e'$. Now, we rename Z'_a into Z''_a, using rule 30. Note that rule 31 is also applicable in this step since $d_3 e'$ is available. If we use rule 31 before 30, we obtain $[\ ]_{Z'_a d_3 e}$. However, this leads to an infinite computation due to rule 39. Note also that by using rule 17 to $[\ ]_{Z'_a d_3}, [\ ]_e$, we can obtain $Z'_a e' d_4 e$ eventually, which also leads to an infinite computation by rule 39 (see below). Assuming that we use rule 30, followed by 31, we obtain $[\ ]_{Z''_a}\ [\ ]_{d_3 e}$ to which we can apply rule 32, eliminating d_3. Now, rules 33–36 follow, eliminating e and replacing Z''_a to af' and finally to a.

(4) Exceptions:

The following are some cases when we get an infinite computation.

1. While simulating $(X \to Y, A \to x)$, A is absent, and there is no type 3 matrix corresponding to X. (Rule 37)
2. There is an $A \in N_2$ remaining after simulation of the terminal matrix.

37. $[\ ]_{XE} \to_{drip} [\ ]_{\#\#}[\ ]_E$,
38. $[\ ]_{\beta\#} \to_{drip} [\ ]_{\#\#}[\ ]_\#$, $\beta \in V \cup \{\#\}$.
39. $[\ ]_{Z'_a e d_i} \to_{drip} [\ ]_\#[\ ]_{e' d_i}, i = 3, 4.$ □

In all the results so far, we saw that we need at least 2 operations to obtain completeness. In this section, we try to figure out if we can obtain completeness using just one of the six operations. Intuitively, this is not possible, since all of the six operations, when used iteratively, change the initial membrane structure (viz., the hierarchy of the membranes with respect to each other). Hence, we need some complementary operations to restore the membrane structure in future steps. Further, we feel that it is impossible to obtain universality without using one of the operations *mate*, *exo*.

In the next section, we look at a new 'stand alone' operation which gives completeness.

7. Universality with a single operation

In this section, we introduce a new operation *'selective mate'*, which is similar to the mate operation, but mates a selected multiset of proteins in the two membranes. We shall explain this operation in more detail below.

The selective mate operation: Let $a \in V, u, x, w, w' \in V^*, v \in V^+$. Consider two membranes $[\ ]_{w'ua}$ and $[\ ]_{wv}$. The *selective mate* operation on $[\ ]_{w'ua}$ and $[\ ]_{wv}$ is defined as $[\ ]_{w'ua}\ [\ ]_{wv} \to [\ ]_{w'}\ [\ ]_{wuxv}$. The selective mate operation selects multisets ua, v in two membranes for mating. The multiset ua evolves into ux and mates with v in the second membrane. The multisets of both the membranes which are not part of the selective mating remain unchanged in their original membranes. Note that unlike mating, selective mating does not merge the contents of the two participating membranes, and hence does not change the structure of the membranes.

The weight of the operation, as before, is the length of uxv. We shall, from now on, write the operation only as $[\ ]_{ua}\ [\ ]_v \to [\ ]\ [\ ]_{uxv}$, since the multisets w, w' are not involved. Since we do not change the structure of membranes, we can always refer to a particular membrane as the output membrane at the end of a halting configuration. Thus, a P system using the selective mate operation is a tuple $\Pi = (V, \mu, w_1, \ldots, w_m, R, [\ ]_{w_j})$ where $[\ ]_{w_j}$ denotes that the membrane with initial contents w_j will be the output membrane. If there are two or more membranes with the same initial contents w_j, then any one of them is chosen as the output membrane.

The family of all sets of vectors $Ps(\Pi)$ computed by P systems Π using at any moment during a halting computation at most m membranes, and and using selective mate rules of weight at most r, is denoted by $PsOP_m(selmate(r))$. When one of the parameters m, r is not bounded, we replace it with a *.

7.1. Universality with selective mate

In this section, we give a universality result obtained by using the selective mate operation.

Theorem 7.1. *$PsRE = PsOP_m(selmate(r))$, for all $m \geq 5, r \geq 3$.*

Proof. Construct a P system

$$\Pi = (V, [\ [\ ]_{XF}\ [\ ]_{AP}\ [\ ]_{LN}\ [\ ]_M\ ]_\lambda, R, [\ ]_{AP})$$

where $X \in N_1, A \in N_2$ correspond to the initial matrix $(S \to XA)$. The output membrane is the one containing the multiset AP in the initial configuration. Note that the AP may not be present in this membrane in a halting configuration. The alphabet consists of

$$\begin{aligned} V &= \{X, X', X'', X^0, X^1, \langle X\rangle, X_l^{(j)}, X_l^{(j)'}, X_l^{(j)''} \mid X \in N_1, n_1 + 1 \leq l \leq n_2\} \\ &\cup \{\langle\alpha\rangle \mid \alpha \in (N_2 \cup T)^*, |\alpha| \leq 2\} \\ &\cup \{f, \langle f\rangle, P, F, L, N, M, K\}. \end{aligned}$$

Proof idea: The membranes containing the multisets XF and AP are used to simulate type-2 and type-4 matrices. The X is replaced by Y'' initially and it becomes part of the membrane containing A. The A is then replaced with x in the presence of Y'', and using the membrane labeled LN. The Y'' then becomes part of the membrane labeled LN, so that no more A's are replaced. Y'' is replaced finally by Y, using membranes labeled LN and M. The simulation of a type 3 matrix again starts using the membranes labeled X and P. Appearance checking is done using the membranes labeled L and F. Finally, termination is achieved by introducing a special symbol Z_a and checking if there are any $A \in N_2$ in the membrane labeled AP, and if not, the symbol P is transferred to the membrane labeled M, thereby retaining only terminals.

The rules R as follows:

(1) For each non-terminal matrix $m_l : (X \to Y, A \to x), 1 \leq l \leq n_1, X, Y \in N_1, A \in N_2, x \in (N_2 \cup T)^*$, we have

1. $[\ ]_X\ [\ ]_{AP} \to [\ ]\ [\ ]_{Y''AP}$,
2. $[\ ]_{Y''A}\ [\ ]_L \to [\ ]\ [\ ]_{Y''\langle x\rangle L}$,
3. $[\ ]_{\langle x\rangle}\ [\ ]_P \to [\ ]\ [\ ]_{xP}$,
4. $[\ ]_{Y''}\ [\ ]_P \to [\ ]\ [\ ]_{Y'P}$,
5. $[\ ]_{Y'}\ [\ ]_F \to [\ ]\ [\ ]_{Y^0F}$,
6. $[\ ]_{Y^0}\ [\ ]_M \to [\ ]\ [\ ]_{Y^1M}$,
7. $[\ ]_{Y^1}\ [\ ]_F \to [\ ]\ [\ ]_{YF}$.

We first examine how a non-terminal type 2 matrix is simulated. Start with rule 1 rewriting $X \in N_1$, and obtaining a configuration $[\ [\ ]_F\ [\ ]_{Y''AP}\ [\ ]_{LN}\ [\ ]_M]_\lambda$. This is followed by the following sequence of configurations obtained by using rules 1–7 in order:

$$\begin{array}{ccc} [\ [\ ]_{XF}\ [\ ]_{AP}\ [\ ]_{LN}\ [\ ]_M]_\lambda & \to & [\ [\ ]_F\ [\ ]_{Y''AP}\ [\ ]_{LN}\ [\ ]_M]_\lambda \\ & & \downarrow \\ [\ [\ ]_F\ [\ ]_{xP}\ [\ ]_{Y''LN}\ [\ ]_M]_\lambda & \leftarrow & [\ [\ ]_F\ [\ ]_P\ [\ ]_{Y''\langle x\rangle LN}\ [\ ]_M]_\lambda \\ \downarrow & & \\ [\ [\ ]_F\ [\ ]_{Y'xP}\ [\ ]_{LN}\ [\ ]_M]_\lambda & \to & [\ [\ ]_{Y^0F}\ [\ ]_{xP}\ [\ ]_{LN}\ [\ ]_M]_\lambda \\ & & \downarrow \\ [\ [\ ]_{YF}\ [\ ]_{xP}\ [\ ]_{LN}\ [\ ]_M]_\lambda & \leftarrow & [\ [\ ]_F\ [\ ]_{xP}\ [\ ]_{LN}\ [\ ]_{Y^1M}]_\lambda. \end{array}$$

Note that we may obtain the same end result by another sequence of configurations, viz., when rule 4 is used before rule 3.

(2) For each matrix $m_l : (X \to Y, B^{(j)} \to \#), n_1 + 1 \leq l \leq n_2$, consider the following rules:

8. $[\ ]_X\ [\ ]_P \to [\ ]\ [\ ]_{X_l^{(j)}P}$,
9. $[\ ]_{X_l^{(j)}}\ [\ ]_F \to [\ ]\ [\ ]_{X_l^{(j)'}QF}$,

10. $[\ ]_{X_l^{(j)\prime}}[\ ]_P \rightarrow [\ ][\ ]_{X_l^{(j)\prime\prime}P}$,
11. $[\ ]_{X_l^{(j)\prime\prime}B^{(j)}}[\ ]_M \rightarrow [\ ]_{\#B^{(j)}M}$,
12. $[\ ]_Q[\ ]_L \rightarrow [\ ][\ ]_{Q'L}$,
13. $[\ ]_{Q'L}[\ ]_M \rightarrow [\ ][\ ]_{Q''LM}$,
14. $[\ ]_{Q''L}[\ ]_N \rightarrow [\ ][\ ]_{Q'''LN}$,
15. $[\ ]_{Q'''}[\ ]_F \rightarrow [\ ][\ ]_{SF}$,
16. $[\ ]_{X_l^{(j)\prime\prime}}[\ ]_S \rightarrow [\ ][\ ]_{\langle Y\rangle S}$,
17. $[\ ]_{\langle Y\rangle S}[\ ]_L \rightarrow [\ ][\ ]_{\langle Y\rangle L}$,
18. $[\ ]_{\langle Y\rangle L}[\ ]_F \rightarrow [\ ][\ ]_{YF}$.

We start with rule 8, replacing $X \in N_1$ by a symbol $X_l^{(j)}$ so that we remember the matrix index l as well as lab_l. It needs to be checked if the corresponding $B^{(j)}$ exists in the membrane containing P, and if so, a trap symbol is introduced. Otherwise, we just need to replace the $X_l^{(j)}$ by Y and ensure that Y is part of the membrane containing F. The following is a sequence of possible configurations:

$$
\begin{array}{l}
[\ [\ ]_{XF}[\ ]_P[\ ]_{LN}[\ ]_M]_\lambda \rightarrow^{8} [\ [\ ]_F[\ ]_{X_l^{(j)}P}[\ ]_{LN}[\ ]_M]_\lambda \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow^{9} \\
[\ [\ ]_{X_l^{(j)\prime}F}[\ ]_P[\ ]_{Q'LN}[\ ]_M]_\lambda \leftarrow^{12} [\ [\ ]_{X_l^{(j)\prime}QF}[\ ]_P[\ ]_{LN}[\ ]_M]_\lambda \\
\qquad\qquad \downarrow^{13,10} \\
[\ [\ ]_F[\ ]_{X_l^{(j)\prime\prime}P}[\ ]_N[\ ]_{Q''LM}]_\lambda \rightarrow^{14} [\ [\ ]_F[\ ]_{X_l^{(j)\prime\prime}P}[\ ]_{Q'''LN}[\ ]_M]_\lambda \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow^{15} \\
[\ [\ ]_{SF}[\ ]_{X_l^{(j)\prime\prime}P}[\ ]_{LN}[\ ]_M]_\lambda \leftarrow^{16} [\ [\ ]_{\langle Y\rangle SF}[\ ]_P[\ ]_{LN}[\ ]_M]_\lambda \\
\qquad\qquad\qquad \downarrow^{17} \\
[\ [\ ]_F[\ ]_P[\ ]_{\langle Y\rangle LN}[\ ]_M]_\lambda \rightarrow^{18} [\ [\ ]_{YF}[\ ]_P[\ ]_{LN}[\ ]_M]_\lambda.
\end{array}
$$

In the above, note that along with rule 15, 11 would have been applicable if the corresponding $B^{(j)}$ were present, leading to an infinite computation. The reader can check that any other order of using the rules viz., choosing 10 over 12, would definitely give rise to the same scenario.

(3) For each terminal matrix $m_l : (X \rightarrow a, A \rightarrow x), 1 \leq l \leq n_1, X \in N_1, A \in N_2, a \in T, x \in T^*$, do the following: First use rules 1 through 6, followed by 7 replacing Y by Z_a. This gives the configuration $[\ [\ ]_{Z_aF}[\ ]_{wP}[\ ]_{LN}[\ ]_M\ ]_\lambda$, $w \in (N_2 \cup T)^*$. Now consider the rules:

19. $[\ ]_{Z_a}[\ ]_P \rightarrow [\ ][\ ]_{aKP}$,
20. $[\ ]_{KP}[\ ]_M \rightarrow [\ ][\ ]_{KPM}$,
21. $[\ ]_{aA}[\ ]_{KP} \rightarrow [\ ][\ ]_{a\#KP}$,
22. $[\ ]_\#[\ ]_\beta \rightarrow [\ ][\ ]_{\#\#\beta}$, $\beta \in V \cup \{\#\}$.

To terminate, we have to simulate a terminal matrix $m_l : (X \rightarrow a, A \rightarrow x), 1 \leq l \leq n_1$. This simulation is exactly same as what we saw in (1), with the exception that we now need to remove all non-terminal symbols from the membrane containing P so that we get a meaningful output. Note that, at the end of every type-2 matrix simulation, the string over $(N_2 \cup T)^*$ is directed towards the membrane containing P, so that at the end, this contains the multiset representing the output. This is illustrated below:

$$
\begin{array}{l}
[\ [\ ]_{XF}[\ ]_{AP}[\ ]_{LN}[\ ]_M]_\lambda \rightarrow^{1\cdots 7} [\ [\ ]_{Z_aF}[\ ]_{xP}[\ ]_{LN}[\ ]_M]_\lambda \\
\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \downarrow^{19} \\
[\ [\ ]_F[\ ]_{xa}[\ ]_{LN}[\ ]_{KPM}]_\lambda \leftarrow^{20} [\ [\ ]_F[\ ]_{xaKP}[\ ]_{LN}[\ ]_M]_\lambda. \quad \square
\end{array}
$$

8. Conclusion and future work

In this paper, we have investigated the power of different brane calculi operations in the context of membranes. We have obtained different characterizations of RE using various combinations of operations from brane calculi. All operations involved changing the structure of membranes, and hence, a 'reverse' operation is required to retain the membrane structure. Based on this observation, we conjecture that none of the basic six operations can give completeness by themselves and that without using either *mate* or *exo*, it is impossible to obtain completeness, while working in the framework of P systems. In this regard, we have introduced a new 'stand alone' operation, which can give completeness.

It is an open problem whether the universality results given above can be improved, and whether there exist new combinations of operations giving universality. The power of the basic operations, and whether there are restrictions on operations which can characterize language classes which are not RE, is an interesting open problem. Again, the power of these systems when operated in a 'minimally parallel' way, as compared to the maximal parallel way considered here, remains to be seen.

References

[1] N. Busi, R. Gorrieri, On the computational power of brane calculi, in: Third Workshop on Computational Methods in Systems Biology, Edinburgh, 2005.
[2] N. Busi, On the computational power of mate, bud, drip brane calculus: Interleaving vs. maximal parallelism, in: Pre-Proceedings of WMC6, Vienna, 2005, pp. 235–252.
[3] L. Cardelli, Brane calculi, interactions of biological membranes, in: Proceedings of Computational Methods in Systems Biology, 2004.
[4] L. Cardelli, Gh. Paun, An universality result based on mate/drip operations, International Journal of Foundations of Computer Science (in press).
[5] J. Dassow, Gh. Paun, Regulated Rewriting in Formal Language Theory, Springer, 1989.
[6] J. Dassow, Gh. Paun, A. Salomaa, Grammars with Controlled Derivations, in: Handbook of Formal Languages, vol. 2, Springer, 1997 (Chapter 3).
[7] R. Freund, Gh. Păun, On the number of non-terminals in graph-controlled, programmed, and matrix grammars, in: Conf. on Universal Machines and Computations, in: LNCS, vol. 2055, Springer-Verlag, Chişinău, 2001, pp. 214–225.
[8] M. Harrison, Introduction to Formal Language Theory, Addison-Wesley, Reading, MA, 1978.
[9] S.N. Krishna, The power of mobility: Four membranes suffice, in: Proceedings of CiE 2005, in: LNCS, vol. 3526, 2005, pp. 242–251.
[10] S.N. Krishna, Upper and Lower Bounds for the Computational Power of P Systems with Mobile Membranes, 2006 (submitted for publication).
[11] S.N. Krishna, Gh. Paun, P systems with mobile membranes, Natural Computing 4 (3) (2005) 255–274.
[12] Gh. Paun, Computing with membranes, Journal of Computer and System Sciences 61 (1) (2000) 108–143.
[13] Gh. Paun, Membrane Computing. An Introduction, Springer, 2002.
[14] A. Salomaa, Formal Languages, Academic Press, 1973.
[15] The membrane computing web page. http://psystems.disco.unimib.it.