Journal of Membrane Computing
https://doi.org/10.1007/s41965-024-00144-1

RESEARCH PAPER

P systems with reactive membranes

Artiom Alhazov[1] · Rudolf Freund[2] · Sergiu Ivanov[3] · David Orellana-Martín[4,5] · Antonio Ramírez-de-Arellano[4,5] · José-Antonio Rodríguez-Gallego[6]

Received: 13 December 2023 / Accepted: 21 March 2024

Abstract
Membranes are one of the key concepts in P systems and membrane computing, and a lot of research activities focus on their properties and possible extensions: membrane division, membrane dissolution, mobile membranes, etc. In this work, we explore the possibility of using membranes for thinking about the emergence of milieu separations at the origins of life. We propose a new variant of P systems with reactive membranes, in which every symbol is initially surrounded by an elementary membrane, and in which membranes can non-deterministically merge and split, leading to the formation of bigger and more complicated membranes. We show that such non-deterministic splitting and merging does not seem to radically affect the computational power: P systems with reactive membranes and non-cooperative rules generate at least all semilinear languages, and cooperative rules allow for simulating partially blind register machines. We briefly discuss using P systems with reactive membranes for illustrating the emergence of autocatalytic cycles, but actual constructions are left for future work.

Keywords Origins of life · P systems · Self-assembly · Space and topology

1 Introduction

Membrane systems are a multiset rewriting-based theoretical construct for natural computing, introduced by Gheorghe Păun in [25], and extensively studied ever since. The structure of a membrane system—or a P system—mimics that of a living cell: it is a hierarchical family of nested membranes, each carrying a multiset of abstract objects and multiset rewriting rules. The objects can be seen as formal representations of chemical species, and the rewriting rules capture the biochemical interactions these species may have.

Beyond the obvious abstraction arrow between biochemical species and formal objects, membrane computing parallels biological systems in another interesting way. In biology, centralization of functions is quite frequent (e.g., central nervous systems, specialized organs, etc.), but not fundamental. Only as a first example, simple organisms carry out many activities in a decentralized way, weakly orchestrated

✉ Sergiu Ivanov
sergiu.ivanov@universite-paris-saclay.fr

Artiom Alhazov
artiom@math.md

Rudolf Freund
rudi@emcc.at

David Orellana-Martín
dorellana@us.es

Antonio Ramírez-de-Arellano
aramirezdearellano@us.es

José-Antonio Rodríguez-Gallego
jrodriguez14@us.es

[1] Vladimir Andrunachievici Institute of Mathematics and Computer Science, State University of Moldova, Academiei 5, Chişinău, MD 2028, Moldova

[2] Faculty of Informatics, TU Wien, Favoritenstraße 9–11, 1040 Wien, Austria

[3] IBISC, Univ. Évry, Paris-Saclay University, 23, boulevard de France, 91034 Évry, France

[4] Research Group on Natural Computing, Department of Computer Science and Artificial Intelligence, Universidad de Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

[5] SCORE Laboratory, I3US, Universidad de Sevilla, Avda. Reina Mercedes s/n, 41012 Sevilla, Spain

[6] Departmento de Construcciones Arquitectónicas I, Universidad de Sevilla, Avda. Reina Mercedes 2, 41012 Sevilla, Spain

Published online: 23 April 2024

by interference between related processes. Take unicellular organisms: a computer scientist may be tempted to consider the genetic material as the program for the whole cell, but it is now known (e.g., [12]) that the relationship between the genotype and the phenotype—its manifestation—is very far from the clear program–execution duality imbuing computer science. As an abstraction of hierarchically structured biochemistry, P systems inherit this weakly centralized way of functioning, which makes them a good candidate for supporting the thought process about some grand laws of biology.

In this paper, we lay the groundwork for using P systems as a tool for thinking about some aspects of the emergence of life. The particular question we focus on is the emergence of milieu separations, which played an essential role as they allowed to isolate and protect relevant processes from the environment [13]. Since P systems already include membranes as first-class citizens, we will use them as a framework for thinking about the emergence of complex regions from simpler ones.

The approach we take here is to posit that every copy of a symbol a is endowed with some *elementary space*—a membrane which initially only contains the symbol a. Two such symbols can bond by merging their membranes, thereby yielding a more complex membrane containing 2 symbols. Such membranes can further merge, yielding bigger and bigger regions. Dually, membranes containing multiple symbols can split into a pair of simpler membranes, with the contents of the original larger membrane distributed across its children. This is in fact membrane separation (e.g., [9, 23, 24]).

Measuring the complexity of a membrane by the number of symbols it contains is simultaneously simple and appropriate: cooperative evolution rules are allowed, so more symbols means more applicable rules and therefore more interactions. In the setup, we establish in this paper, all membranes share the same common set of evolution rules. The rules can naturally be seen as defining a chemistry, while membrane merging and splitting can on the other hand be seen as some lower-level ground laws governing who may interact with whom, i.e., the topology of the interactions. The resulting abstract structures featuring merging and splitting membranes are therefore systems in which objects interact based on the non-deterministic variations in their neighborhoods. We call such structures P systems with *reactive membranes*.

Before using P systems with reactive membranes as a formal tool, a number of important details have to be sorted out. In particular, we show that the definition of membrane splitting and merging turns out to be rather nontrivial. Choosing when to recover and how to interpret the result impacts the form of the computations of a P system with reactive membranes, as well as what kind of results one can expect. Finally, this P system variant as informally introduced above and defined in Sect. 3 is very basic and may be extended in many ways, as we briefly show in Sect. 5.

Note that we do not pretend to faithfully model in any way the processes which happened at the origins of life. Rather, we acknowledge the exceptional complexity of these processes, as well as the impossibility to experimentally verify any of the related hypotheses (e.g., [19]). The intended role of P systems with reactive membranes is to serve as a formal vehicle for an otherwise abstract thought process, to help verify the latter in a basic way, and to help the researcher to deal with complex questions. This approach is similar in spirit to the works [28, 29], in which sign Boolean networks are used with a similar purpose.

P systems with reactive membranes are naturally part of the lineage of P systems with active membranes, and feature similarities with other variants in this family. Among closely related variants are P systems with mobile membranes, in which membranes are allowed to move across the membrane structure, and thereby change their immediate neighbors [10, 11, 22]. Other variants are P systems with vesicles of multisets, in which multisets are contained in vesicles, which are contained in membranes, implying that entire multisets of symbols can travel between different membranes, thereby activating different sets of rules [7, 17]. A key specificity of P systems with reactive membranes setting them apart from the other variants is that membrane splitting and merging is global, compulsory, and independent of the contents of the membranes or of the rules. This feature introduces a basic form of space, through which the entities travel and in which they interact in their immediate neighborhood. On the other hand, compulsory splitting and merging modulates the computational power in interesting ways.

This paper is an improved and extended version of the works [3, 4], and is structured as follows: In Sect. 2, we recall some basic concepts from formal languages and P systems. In Sect. 3, we introduce P systems with reactive membranes, and define the precise semantics of splitting and merging of membranes. In Sect. 4, we present some first results concerning the computational power of P systems with reactive membranes, with non-cooperative and cooperative rules. In Sect. 5, we give some examples of possible extensions to the new variant. Finally, in Sect. 6, we discuss the potential of reactive membranes for illustrating some processes which happened at the origins of life, as well as some aspects of their computational power.

2 Preliminaries

For two natural numbers $a, b \in \mathbb{N}$, $a \leq b$, we use the notation $[a..b]$ to refer to the interval of natural numbers between a and b, both included: $[a..b] = \{a, a+1, \ldots, b\}$.

For an alphabet V, a finite non-empty set of abstract symbols, the free monoid generated by V under the operation of concatenation, i.e., the set containing all possible strings over V, is denoted by V^*. The empty string is denoted by λ, and $V^* \setminus \{\lambda\}$ is denoted by V^+. For a string $w \in V^*$ and a symbol $a \in V$, by $|w|$ we denote the length of the string and by $|w|_a$ the number of copies of a appearing in w.

A multiset over V is a function $w : V \to \mathbb{N}$, assigning the number of times an element of V appears in w. The infinite set of all multisets over V is denoted by V°. The family of finite sets of finite multisets over V is denoted by $\mathcal{P}_f in(V^\circ)$.

Given two multisets w_1 and w_2 over V, their multiset union $w_1 \cup w_2$ is defined as $(w_1 \cup w_2)(a) = w_1(a) + w_2(a)$, for all $a \in V$. As their multiset intersection $w_1 \cap w_2$, we define $(w_1 \cap w_2)(a) = \min\{w_1(a), w_2(a)\}$. A restriction of the multiset $w : V \to \mathbb{N}$ to the subset $B \subseteq V$ is the multiset $w|_B : V \to \mathbb{N}$ with the property that $w|_B(a) = w(a)$ if $a \in B$ and $w|_B(a) = 0$ otherwise.

To spell out a multiset w, we will generally write any string containing exactly the same symbols with the same multiplicities. For example, the strings aab, aba, ba^2 will be used to refer to the same multiset w with the property $w(a) = 2$, $w(b) = 1$, and $w(c) = 0$ for all $c \in A \setminus \{a, b\}$. We denote the empty multiset by Λ, i.e., $\forall a \in A : \Lambda(a) = 0$, and its string representation is λ, the empty string. We will use the notation $|w|$ to refer to the size of the multiset: $|w| = \sum_{a \in A} w(a)$.

Given an alphabet V, we define a pair of functions connecting the finite strings in V^* and all finite multisets in V°: $\mathsf{supp}^* : V^* \to V^\circ$ and $\mathsf{str} : V^\circ \to \mathcal{P}(V^*)$. Given a string $w \in V^*$, $w' = \mathsf{supp}^*(w)$ is the multiset containing exactly the same symbols as w in the same number of copies: $w'(a) = |w|_a$, for all $a \in V$. The function str is the inverse of supp^*, and associates to every multiset the set of strings containing the same symbols in the same number of copies: $\mathsf{str}(w') = \{w \in V^* \mid \mathsf{supp}^*(w) = w'\} = (\mathsf{supp}^*)^{-1}(w')$.

The family of regular, context-free, and recursively enumerable string languages is denoted by $\mathcal{L}(REG)$, $\mathcal{L}(CF)$, and $\mathcal{L}(RE)$, respectively. For a family of languages FL, the family of Parikh images of languages in FL is denoted by $PsFL$.

As $Ps\mathcal{L}(REG) = Ps\mathcal{L}(CF)$, in the area of multiset rewriting $\mathcal{L}(CF)$ plays no role at all, and in the area of membrane computing we often only get characterizations of $Ps\mathcal{L}(REG)$ and $Ps\mathcal{L}(RE)$.

For further notions and results in formal language theory, we refer to textbooks like [14] and [27].

In the rest of this section, we briefly recall P systems and the related concepts. For more extensive overviews, we refer the reader to [20, 26].

A *(transition) P system* is a construct

$$\Pi = (O, T, \mu, w_1, \ldots, w_n, R_1, \ldots R_n, \delta, h_i, h_o) \text{ where}$$

- O is the alphabet of *objects*,
- $T \subseteq O$ is the alphabet of *terminal objects*,
- μ is the *membrane structure* injectively labeled by the numbers from $[1..n]$ and usually given by a sequence of correctly nested brackets,
- w_i are the multisets giving the *initial contents* of each membrane i, $1 \leq i \leq n$,
- R_i is the finite *set of rules* associated with membrane i, $1 \leq i \leq n$,
- δ is the *derivation mode*, and
- h_i and h_o are the labels of the *input membrane* and the *output membrane*, respectively; $1 \leq h_i \leq n, 1 \leq h_o \leq n$.

Taking into account the well-known flattening process for P systems, which means that computations in a P system with an arbitrary (static) membrane structure can be simulated in a P system with only one membrane, e.g., see [16], often only *simple P systems* are considered, i.e., with the simplest membrane structure of only one membrane region, and then we write:

$$\Pi = (O, T, w, R, \delta)$$

Quite often, the rules associated with membranes are multiset rewriting rules (or special cases of such rules). Multiset rewriting rules have the form $u \to v$, with $u \in O^\circ \setminus \{\Lambda\}$ and $v \in O^\circ$. If $|u| = 1$, the rule $u \to v$ is called *non-cooperative*, otherwise it is called *cooperative*. In *communication P systems*, rules are additionally allowed to send symbols to the neighboring membranes. In this case, for rules in R_i, $v \in (O \times Tar_i)^\circ$, where Tar_i contains the symbols *out* (corresponding to sending the symbol to the parent membrane), *here* (indicating that the symbol should be kept in membrane i), and in_j (indicating that the symbol should be sent into the child membrane j of membrane i). When writing out the multisets over $O \times Tar_i$, the indication *here* is often omitted.

In P systems, rules are often applied in the maximally parallel way: in one derivation step, only a non-extendable multiset of rules can be applied. The rules are not allowed to consume the same instance of a symbol twice, which creates competition for objects and may lead to non-deterministic choices between the maximal collections of rules applicable in one step. The *maximally parallel derivation mode* is generally denoted by *max*. Other derivation modes include the *sequential derivation mode—seq*—in which exactly one rule is applied in every step, the *set maximally parallel derivation mode—smax*—only allowing multisets of rules in which every rule has multiplicity 1, as well as the *asynchronous derivation mode—asyn*—under which no restriction is imposed on the applied multiset of rules. We refer to the papers [5, 6, 8, 18] for an in-depth discussion of the matter.

A *configuration* of a simple P system is simply the multiset of objects contained in its only membrane at a

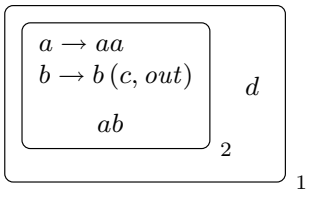


Fig. 1 An example of a simple P system

given step. A *computation* of a P system is traditionally considered to be a sequence of configurations it can successively visit, stopping at the halting configuration. A *halting* configuration is a configuration in which no rule can be applied any more, in any membrane. The *result of a computation* in a P system Π as defined above is the contents of the output membrane h_o in the halting configuration projected over the terminal alphabet T.

We will use the notations $N(\Pi)$ and $Ps(\Pi)$ to refer to the number language and the language of multisets, respectively, generated by Π. The notation $OP_n(\delta, \tau)$ will refer to the family of P systems with at most n membranes, operating under the derivation mode δ and relying on the rules of type τ, where $\tau = coo$ if cooperative rules are allowed and $\tau = ncoo$ if all rules are non-cooperative. Finally, we use the notations $NOP_n(\delta, \tau)$ and $PsOP_n(\delta, \tau)$ to refer to the family of number languages and multiset languages, respectively, generated by the P systems in the family $OP_n(\delta, \tau)$.

Example 1 Fig. 1 shows the graphical representation of the P system formally given by

$$\begin{aligned}\Pi &= (\{a, b, c, d\}, \{a, d\}, [_1[_2]_2]_1, d, ab, R_1, R_2, max, 1, 1),\\ R_2 &= \{a \to aa, b \to b\,(c, out)\},\\ R_1 &= \emptyset.\end{aligned}$$

In the maximally parallel mode, the inner membrane 2 of Π will apply as many instances of the rules as possible, thereby doubling the number of objects a, and ejecting a copy of c into the surrounding (skin) membrane in each step. The symbol d in the skin membrane is not used. Therefore, after k steps of evolution, membrane 2 will contain the multiset $a^{2^k}b$ and membrane 1 the multiset $c^k d$. Since all rules are always applicable in Π, this P system never halts. □

3 Reactive membranes

A *P system with reactive membranes* is the following construct:

$\Pi = (O, T, W_0, R, \delta)$ where

- O is the alphabet of *objects*,
- $T \subseteq O$ is the alphabet of *terminal objects*,
- $W_0 \in (O^\circ)^\circ$ is the (finite) *initial multiset of multisets* over O,
- $R \subseteq O^\circ \times O^\circ$ is the set of *evolution rules*, and
- δ is the *derivation mode*.

For all rules in R, we will require at least one of the sides to be non-empty, i.e.,

$\forall u \to v \in R : u \neq \lambda$ or $v \neq \lambda$.

We immediately stress two major features of this definition. On the one hand, we do not include any membrane structure. Indeed, as W_0 hints, we simply use individual multisets to represent the contents of the individual membranes, without explicitly representing the membranes themselves. Incidentally, this means that membranes do not nest in this model. On the other hand, the evolution of all symbols in (the multisets in) all the membranes is governed by the same common set of rules R.

A configuration of Π is any multiset of multisets over O. Similarly to networked models of computing like networks of evolutionary processors (e.g., [21]) or tissue P systems with vesicles of multisets [7], a computation step in P systems with reactive membranes consists of two stages, occurring in this order:

1. splitting and merging,
2. evolution.

Informally, the splitting and merging stage implements the non-deterministic evolution of the membranes—individual multisets under this definition—as described in the introduction: any two multisets may merge, and any multiset may split in two. The evolution stage consists in applying the evolution rules in R to every multiset of the configuration resulting after splitting and merging, according to the derivation mode δ. In the following paragraphs, we give a formal description of both stages, applied to a configuration $W_i \in (O^\circ)^\circ$.

Splitting and merging stage

1. Non-deterministically partition W_i into 3 submultisets:

   $$W_i = M_i \cup S_i \cup I_i, \quad M_i, S_i, I_i \in (O^\circ)^\circ,$$

   such that $|M_i|$ is even, and the multisets S_i, M_i, and I_i are mutually disjoint, i.e., $S_i \cap M_i = S_i \cap I_i = M_i \cap I_i = \Lambda$. The multisets in M_i will be merged pairwise, the

multisets in S_i will be split, and the multisets in I_i will remain intact.

2. Partition M_i into disjoint pairs by picking a string $w_1 w_2 \dots w_{k-1} w_k \in \mathsf{str}(M_i)$ non-deterministically, with $k = |M_i|$, pairwise merging the pairs of successive multisets to obtain the string $\hat{M}_i = (w_1 \cup w_2) \dots (w_{k-1} \cup w_k)$, and then constructing the corresponding multiset of multisets:

   $M'_i = \mathsf{supp}^*(\hat{M}_i)$.

3. Define $\mathsf{split}(v)$ to be the set of all possible ways to split the multiset v into two multisets:

   $\mathsf{split}(v) = \{(\alpha, \beta) \mid \alpha \cup \beta = v,\ \alpha, \beta \in O^\circ\}$.

   Consider $v_1 \dots v_t \in \mathsf{str}(S_i)$, $t = |S_i|$, and non-deterministically construct the new string $\hat{S}_i = \alpha_1 \beta_1 \dots \alpha_t \beta_t$ such that $(\alpha_i, \beta_i) \in \mathsf{split}(v_i)$, $1 \le i \le t$. Finally, define:

   $S'_i = \mathsf{supp}^*(\hat{S}_i)$.

4. Compute the new intermediate configuration $W'_i \in (O^\circ)^\circ$ as

   $W'_i = M'_i \cup S'_i \cup I_i$.

In the above presentation, we describe merging before splitting, but the order of the two substeps does not matter since they occur on disjoint multisets M_i and S_i. Furthermore, we stress that multiple intermediate configurations W'_i may be obtained from the same configuration W_i.

Evolution stage The evolution stage is defined in the conventional way by applying the rules in R to every multiset in W'_i individually, according to the derivation mode δ, and respecting the multiplicities of the elements of W'_i. More concretely, pick any string $w'_1 \dots w'_l \in \mathsf{str}(W'_i)$ and construct the new string $\hat{W}'_i = \hat{w}'_1 \dots \hat{w}'_l$ with the property that $w'_i \overset{\delta,R}{\Rightarrow} \hat{w}'_i$. Finally, define the new configuration as

$W_{i+1} = \mathsf{supp}^*(\hat{W}'_i)$.

A configuration W is *halting* if no rules are applicable in the evolution stage, for any intermediate configuration W' which can be obtained from W in the splitting and merging stage. An *n-step halting computation* of a P system with reactive membranes Π is a finite sequence of configurations $(W_i)_{0 \le i \le n}$ such that W_{i+1} is obtained from W_i by the computation step described above, and W_n is a halting configuration.

As the *result of a computation* in a P system with reactive membranes Π as defined above we take all the terminal objects appearing in the membranes present in a halting configuration W_n:

$$\left(\bigcup_{w \in W_n} w \right) \Big|_T = \bigcup_{w \in W_n} w|_T,$$

where the union $\bigcup_{w \in W_n}$ iterates over the elements of W_n respecting their multiplicities. As for simple P systems, we will use the notations $N(\Pi)$ and $Ps(\Pi)$ to refer to the number language and the language of multisets, respectively, generated by the P systems with reactive membranes Π.

In what follows, we will use the set builder notation to describe configurations which are multisets of multisets. For example, we will write $\{ab, ab, c\}$ to refer to a configuration containing twice the multiset ab and once the multiset c.

To conclude the introduction of P systems with reactive membranes, we again stress that the splitting and merging of multisets (or membranes) is non-deterministic, imposed in every computation step, and independent of the features of the configuration or of the rules in R. More concretely, the rules in R cannot directly influence which symbols will appear next to which after the splitting and merging stage.

Example 2 Consider the following P system with reactive membranes:

$$\begin{aligned} \Pi &= (O, T, W_0, R, max), \text{ where} \\ O &= \{a, b, c, d, e, f\}, \\ T &= \{d, f\}, \\ W_0 &= \{a, b, c\}, \\ R &= \{ab \to d, abc \to f, a \to e\}. \end{aligned}$$

For the first step of the computation, Π may decide to not split or merge any multisets ($M_0 = S_0 = \Lambda$, $I_0 = W_0$), meaning that the evolution rules will be applied directly to singleton multisets a, b, and c. While no rules are applicable to b or c individually, the rule $a \to e$ will have to be applied to a, yielding the next configuration $W_1 = \{b, c, e\} \in (O^\circ)^\circ$, containing three singleton multisets. We can immediately conclude that the rules $ab \to d$ and $abc \to f$ will never be applicable any more later in this computation, as there is no way to reintroduce a. In sum, this halting computation yields the result Λ.

Now suppose that Π decides to merge the multisets a and b in the first step, yielding the intermediate configuration $W'_0 = \{ab, c\}$. In this case, a non-deterministic choice will appear in the evolution stage between applying the rule $ab \to d$ or $a \to e$ (both singleton sets of rules are non-extendable). As a consequence, the following two possibilities exist for the second configuration: $\{d, c\}$ and $\{eb, c\}$. In sum, these halting computations yield the results d and Λ, respectively.

Finally, note that the rule $abc \to f$ will never be applicable with $W_0 = \{a, b, c\}$, since putting a, b, and c together in one membrane requires at least two mergers, and a will necessarily be consumed by $a \to e$ or $ab \to d$ along the way. On

the other hand, if we put together a, b and c in one multiset from the start, or even if we put ab together and c apart, the rule $abc \to f$ will have a chance to be applied. In particular, in the case in which the initial configuration is $\{ab, c\}$, it suffices to consider the branch of the computation along which Π decides to merge the two multisets in the first splitting and merging stage. In sum, with the initial sets $\{ab, c\}$ and $\{abc\}$, we can get the results Λ, d, and f. □

As indicated by the example discussed above, it makes a difference in how many multisets the initial multiset of objects is divided. Thus, we will use the notation $Re_nOP(\delta, \tau)$ to refer to the family of P systems with reactive membranes starting with n initial multisets, running under the mode δ and using rules of type $\tau \in \{coo, ncoo\}$, as well as the notations $NRe_nOP(\delta, \tau)$ and $PsRe_nOP(\delta, \tau)$ to refer to the family of number languages and multiset languages, respectively, generated by the P systems with reactive membranes from $Re_nOP(\delta, \tau)$.

Whereas, on the one hand, the previous example shows the effect of having more than one initial membrane, prohibiting the application of some evolution rules, the next example shows that the halting condition can be fulfilled due to the fact that symbols are distributed over several membranes, although some rule could be applied if all symbols on its left-hand side could be put into the same membrane by a merge operation. As merging can only combine the contents of two membranes, we can already get the situation that a rule with three symbols in its left-hand side cannot be applied any more.

Example 3 Consider the following P system with reactive membranes:

$$\begin{aligned}
\Pi &= (O, T, W_0, R, max), \text{ where} \\
O &= \{a_i, a'_i, a''_i \mid 1 \le i \le 3\} \cup \{f\}, \\
T &= \{a''_1, a''_2, a''_3, f\}, \\
W_0 &= \{a_1 a_2 a_3\}, \\
R &= \{a_i \to a'_i, a'_i \to a''_i, a''_1 a''_2 a''_3 \to f\}.
\end{aligned}$$

If in the first two steps of the computation, Π decides to not split or merge any multisets, from $W_0 = \{a_1 a_2 a_3\}$ with applying the rules $\{a_i \to a'_i \mid 1 \le i \le 3\}$, after the first evolution step, we obtain $W_1 = \{a'_1 a'_2 a'_3\}$, and by then applying the rules $\{a'_i \to a''_i \mid 1 \le i \le 3\}$, after the second evolution step, we obtain $W_2 = \{a''_1 a''_2 a''_3\}$. Keeping $\{a''_1 a''_2 a''_3\}$ in the same membrane then allows for applying the rule $a''_1 a''_2 a''_3 \to f$, thus obtaining the terminal result $W_3 = \{f\}$, as W_3 is a halting configuration.

Yet with two splits, but still applying the rules $\{a_i \to a'_i \mid 1 \le i \le 3\}$ in the first evolution step and the rules $\{a'_i \to a''_i \mid 1 \le i \le 3\}$ in the second evolution step, we get a two-step halting computation

$$\{a_1 a_2 a_3\} \Longrightarrow \{a'_1, a'_2 a'_3\} \Longrightarrow \{a''_1, a''_2, a''_3\}$$

yielding the terminal result $a''_1 a''_2 a''_3$.

We also mention that with having $T = \{f\}$ only, this halting computation yields the result Λ. □

4 Computational power: first results

In this section, we give some first results regarding the computational power of P systems with reactive membranes. We first show that we one can always pad the initial configuration with empty membranes, a result which holds for non-cooperative rules and cooperative rules.

Lemma 1 *For every* $\Pi \in Re_1OP(\delta, \tau)$ *and for every* $n > 1$, *there exists a P system* $\Pi' \in Re_nOP(\delta, \tau)$, *such that* $Y(\Pi) = Y(\Pi')$, *for every* $Y \in \{Ps, N\}$.

Proof Given a P system with reactive membranes using rules of type τ

$$\Pi = (O, T, \{w\}, R, \delta),$$

an equivalent P system with reactive membranes using rules of type τ with n initial membranes is

$$\Pi' = (O, T, \{w, w_2 = \Lambda, \ldots, w_n = \Lambda\}, R, \delta).$$

In an empty membrane Λ, no non-cooperative rules or cooperative rules are applicable. Moreover, merging a membrane X with Λ yields X again, so no additional applications of rules can happen. Hence, in any computation in Π', no multiset can appear in any of its configurations which not also can be obtained in a configuration of a computation in Π. □

In what follows, we study the impact of splitting and merging separately in the cases of cooperative and non-cooperative rules.

4.1 Non-cooperative rules and derivation trees

This subsection explores the formal ramifications of the following remark stating that the applicability of non-cooperative rules is not affected by splitting and merging of membranes.

Remark 1 Recall that in P systems with reactive membranes, all membranes share the same set of rules and consider a non-cooperative rule $r : a \to v$. Its applicability is not affected by the membrane in which a is located,

meaning that it can be applied the same number of times to any configuration containing the same total number of copies of a, independently of the way in which these copies are distributed across the membranes. One immediate consequence is that the halting condition for a configuration W can be checked without considering all possible splits and mergers and then the non-applicability of the rules in all membranes. Instead, it suffices to check the non-applicability of the rules to the objects in $\bigcup_{w \in W} w$ —the union of the multisets in all the membranes of W.

We now briefly recall the concept of derivation trees, which we will use to prove the main results of this subsection. We refer to the [26] for an extensive presentation and discussion.

Consider a simple P system $\Pi = (O, T, w_0, R, \delta) \in OP_1(\delta, ncoo)$ and the following computation

$$w_0 \overset{\delta,R}{\Rightarrow} w_1 \overset{\delta,R}{\Rightarrow} \ldots \overset{\delta,R}{\Rightarrow} w_n,$$

in which Π derives the final multiset v from the initial multiset w by applying at every step the rules from R according to the mode δ. We build the corresponding derivation tree in the following way. The root of the tree is the empty multiset Λ, and every copy of a symbol from w_0 is a direct child of the root Λ. For every rule $r : a \to b_1 \ldots b_k$ applied in the first step $w_0 \overset{\delta,R}{\Rightarrow} w_1$, we add the symbols b_1 to b_k as the children of the symbol a, and we additionally label the node a with the rule name r, i.e., the corresponding node in the tree becomes (a, r). We proceed similarly for the rest of the steps of the computation. In the end, we will obtain a tree whose leaves are exactly the symbols of w_n, in the corresponding number of copies.

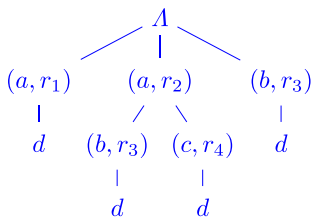
Example 4 Consider the following simple P system:

$$\begin{aligned}
\Pi &= (O, T, w, R, max), \text{ where} \\
O &= \{a, b, c, d\} \\
T &= \{d\}, \\
w &= aab, \\
R &= \{r_1 : a \to d, r_2 : a \to bc, r_3 : b \to d, r_4 : c \to d\}.
\end{aligned}$$

The following is one of the computations of Π in the derivation mode *max*:

$$aab \overset{r_1 r_2 r_3}{\Rightarrow} dbcd \overset{r_3 r_4}{\Rightarrow} dddd.$$

Here is the corresponding derivation tree constructed according to the procedure described above:



The internal nodes of this tree have the form (x, r), where $x \in O$ is a symbol and $r \in R$ is the rule which was applied to x. The leaves of the tree have the form $x \in O$ since no rules are applied to them. Collecting the leaves yields the multiset $dddd$, corresponding to the final multiset of the computation we picked. Since in P systems we are dealing with multiset rewriting, the levels of the derivation tree are unordered, i.e., the order in which one writes the children of a particular node is unimportant.

Derivation trees can be used to prove the following folklore result, stating that simple P systems with non-cooperative rules and without any additional control mechanisms generate exactly the regular multiset languages.

Proposition 1 ([8, 26]) *For any* $\delta \in \{asyn, seq, max, smax\}$ *and* $Y \in \{N, Ps\}$, *it holds that* $YOP_1(\delta, ncoo) = Y\mathcal{L}(REG)$.

The idea of derivation trees can be extended to P systems with reactive membranes with non-cooperative rules without any change, as the following remark states.

Remark 2 Recall from Remark 1 that in a P system with reactive membranes $\Pi^r e = (O, T, W_0, R, \delta)$ all membranes share the same set of rules R, and that the applicability conditions of non-cooperative rules do not depend on how the symbols are distributed across membranes. Consider now the following computation of $\Pi^r e, \sigma^r e : W_0 \Rightarrow W_1 \Rightarrow \cdots \Rightarrow W_n$, where each computation step $W_i \Rightarrow W_{i+1}$ is carried out according to the procedure described in Sect. 3. Construct the sequence of multisets $(w_i)_{0 \le i \le n}$ such that w_i is the union of all membrane contents of the configuration W_i. Furthermore, let ρ_i be the union of all multisets of rules applied in all membranes of W_i in $\sigma^r e$. Consider the following object:

$$\sigma : w_0 \overset{\rho_0}{\Rightarrow} w_1 \overset{\rho_1}{\Rightarrow} \ldots \overset{\rho_{n-1}}{\Rightarrow} w_n.$$

It is a sequence of multisets over O, in which w_{i+1} is obtained from w_i by applying a multiset of applicable rules, and w_n is the halting multiset in the sense that no further rules from R are applicable to it. In other words, σ is a computation of the simple P system $\Pi = (O, T, w_0, R, asyn)$. A derivation tree can be constructed for σ, which will also capture which

rules were applied in the original P system with reactive membranes $\Pi^r e$ to obtain the final configuration W_n.

Informally, Remark 2 improves on Remark 1 by highlighting that the mutual independence of rule applicability conditions and splitting and merging allows for constructing derivation trees for any computation of a P system with reactive membranes. Remark 2 also puts forward the strong connection between the computations of P systems with reactive membranes and simple P systems working under the asynchronous mode. In order to formalize this connection, we show how translations between different derivation modes can be explicitly operated in the case of non-cooperative rules and simple P systems.

We recall from [18] that $Appl(\Pi, w)$ refers to the set of multisets of rules of Π applicable to the configuration w, and $Appl(\Pi, w, \delta)$ refers to the set of multisets of rules applicable to the configuration w according to the mode δ.[1]

Definition 1 A mode δ is called *normal* if for any two multisets of objects w_1 and w_2 the following conditions hold:

1. $Appl(\Pi, w_1) = Appl(\Pi, w_2) \implies Appl(\Pi, w_1, \delta) = Appl(\Pi, w_2, \delta)$,
2. $Appl(\Pi, w_1) \neq \emptyset \implies Appl(\Pi, w_1, \delta) \neq \emptyset$.

Informally, the way in which a normal mode picks the multisets of rules to apply out of $Appl(\Pi, w)$ does not depend on the configuration w other than via the applicability conditions of the rules. Furthermore, if some rules can be applied in w ($Appl(\Pi, w_1) \neq \emptyset$), then a normal mode δ allows applying some of them ($Appl(\Pi, w_1, \delta) \neq \emptyset$). In the case of non-cooperative rules, this means that all normal modes eventually result in the same rule applications, as the following theorem shows.

Theorem 1 *Let δ_1 and δ_2 be two normal modes, and $\Pi_1 = (O, T, w, R, \delta_1)$ and $\Pi_2 = (O, T, w, R, \delta_2)$ two simple P systems differing only in the mode. Then $Y(\Pi_1) = Y(\Pi_2)$, for any $Y \in \{N, Ps\}$.*

Proof Consider a computation σ_1 of Π_1 under the mode δ_1 and take the corresponding derivation tree. In what follows, we present an algorithm for constructing a derivation σ_2 of Π_2 under the mode δ_2 having the same derivation tree. This algorithm progressively marks the nodes of the derivation tree, until all of the nodes are marked. We refer to the set of nodes which are immediate children of marked nodes as the *fringe* and denote it by the symbol F_i for step i.

1. Set $i \leftarrow 0$, $w_0 \leftarrow w$, $\sigma_2 \leftarrow w_0$, and mark the root Λ of the tree.
2. Take the multiset of nodes in the fringe to which a rule was applied in σ_1:

   $\rho = \{(a, r) \in F_i \mid a \in O, r \in R\}$.

   If ρ is empty, stop.
3. Find a submultiset ρ' of ρ such that $\bar{\rho}' = \{r \mid (a, r) \in \rho'\} \in R^\circ$ satisfies the mode δ_2.
4. Take a multiset $w_{i+1} \in O^\circ$ with the property $w_i \overset{\bar{\rho}'}{\Rightarrow} w_{i+1}$ (i.e., apply the rules in $\bar{\rho}'$ to w_i).
5. Add w_{i+1} to the derivation σ_2, mark the nodes of the tree appearing in ρ', build the new fringe F_{i+1}, and increment $i \leftarrow i + 1$.
6. Go to 1..

It follows from the way in which the algorithm above operates that the fringe always contains the symbols in the current configuration w_i of the newly constructed derivation σ_2, i.e., $w_i = \{a \mid (a, r) \in F_i\}$. δ_2 is a normal mode and the rules in R are non-cooperative, it is therefore always possible to construct ρ' which is both a subset of ρ and $\bar{\rho}'$ satisfies the conditions of δ_2, in particular $\bar{\rho}'$ is applicable to w_i. Furthermore, every iteration of the algorithm marks a non-empty set of nodes of the tree, which guarantees termination. Finally, in σ_2, the same rules are applied to the same symbols, implying that σ_1 and σ_2 have the same (unordered) derivation trees, and consequently produce the same final multiset. We conclude the proof by remarking that the symmetric construction can be used to translate any derivation of Π_2 into a derivation of Π_1. □

The derivation σ_2 in the proof above does not need to contain the same number of steps as σ_1. Indeed, if δ_2 only allows for smaller multisets than δ_1 (e.g., $\delta_2 = seq$ and $\delta_1 = max$), then σ_2 may be longer. Furthermore, a single derivation σ_1 may translate into different derivations in Π_2, as picking ρ' and w_{i+1} is non-deterministic.

The following corollary follows immediately from Theorem 1 and from the fact that *asyn* is a normal mode.

Corollary 1 *For any normal mode δ and any $Y \in \{N, Ps\}$, it holds that $YOP_1(\delta, ncoo) = Y\mathcal{L}(REG)$.*

Using the above results, we can now show that the behavior of P systems with reactive membranes under normal modes and with non-cooperative rules is fundamentally similar to that of simple P systems. We first show that the behavior of any simple P system can be faithfully reproduced by a P system with reactive membranes.

[1] We directly specialize here the more general definition from [18] to the case in which the configuration contains exactly one multiset of symbols.

Lemma 2 *For any normal mode δ and any simple P system $\Pi \in OP_1(\delta, ncoo)$, there exists a P system with reactive membranes $\Pi^r e \in Re_n OP(\delta, ncoo)$ such that $Y(\Pi) = Y(\Pi^r e)$, for any $Y \in \{N, Ps\}$.*

Proof Let $\Pi = (O, T, w_0, R, \delta)$. We claim that the P system with reactive membranes $\Pi^r e = (O, T, \{w_0\}, R, \delta)$ satisfies the statement of the lemma, i.e., $Y(\Pi) = Y(\Pi^r e)$.

Indeed, $\Pi^r e$ can directly reproduce any computation of Π by never splitting or merging—i.e., M_i and S_i from the splitting and merging stage are always empty—and applying exactly the same multisets of rules at the same steps. Since $\Pi^r e$ carries out the entire computation in a single membrane, all multisets of rules applied in Π are applicable and satisfy the mode δ.

Consider on the other hand a computation $\sigma^r e$ of $\Pi^r e$ under the mode δ and translate it into a computation σ' of a P system $\Pi' = (O, T, w_0, R, asyn)$ working under the asynchronous mode, following the procedure shown in Remark 2. Then translate σ' into a computation σ of Π using the construction from the proof of Theorem 1. σ will have the same derivation tree as $\sigma^r e$, which proves the statement of the lemma. □

We now show the converse result: the behavior of any P system with reactive membranes can be faithfully reproduced by a simple P system.

Lemma 3 *For any normal mode δ, any $n \geq 1$, and any P system with reactive membranes $\Pi^r e \in Re_n OP(\delta, ncoo)$ there exists a simple system $\Pi \in OP_1(\delta, ncoo)$ such that $Y(\Pi) = Y(\Pi^r e)$, for any $Y \in \{N, Ps\}$.*

Proof Let $\Pi^r e = (O, T, W_0, R, \delta)$. We claim that the simple P system $\Pi = (O, T, w_0, R, \delta)$, with $w_0 = \bigcup_{w \in W_0} w$, satisfies the statement of the lemma, i.e., $Y(\Pi) = Y(\Pi^r e)$.

The proof follows essentially the same reasoning as that of Lemma 2. The P system Π can simulate any computation $\sigma^r e$ of $\Pi^r e$, because $\sigma^r e$ can be translated into a computation of Π with the same derivation tree. On the other hand, $\Pi^r e$ can directly reproduce any computation of Π by never splitting or merging. □

Combining the two previous lemmas yields the following result.

Theorem 2 *For any normal mode δ, any $n \geq 1$, the following holds:*

$$YRe_n OP(\delta, ncoo) = YOP_1(\delta, ncoo).$$

An immediate corollary of Theorem 2 and Corollary 1 is the following characterization.

Corollary 2 *For any normal mode δ, any $n \geq 1$, $YRe_n OP(\delta, ncoo) = YREG$.*

4.2 Cooperative rules

In this subsection, we show that P systems with reactive membranes working under the maximally parallel mode and using *cooperative rules* can simulate partially blind register machines. As a reminder, we mention that partially blind register machines (PBRM) have programs consisting of the following two types of instructions for incrementing and decrementing a register:

- $(p, \text{ADD}(r), q, s)$: in state p increment register r and jump to state q or state s;
- $(p, \text{SUB}(r), q)$: in state p try to decrement register r; if successful, jump to state q, otherwise abort the computation without producing a result.

Partially blind register machines feature a final zero check: the register machine only halts with producing a result if all non-output registers are empty when the machine reaches the halting instruction uniquely labeled by h.

We will refer to the set of multiset languages generated by partially blind register machines by $PsPBRM$.

Theorem 3 *For any $\delta \in \{asyn, seq, max, smax\}$,*

$$PsPBRM \subseteq PsRe_1 OP(\delta, coo).$$

Proof The main idea of the proof is that throughout the simulation of the partially blind register machine, the configurations of the P system with reactive membranes Π always contain exactly one instance of the symbol representing the label of the instruction to be carried out next. The contents of a register r is represented by the total number of symbols a_r in the configurations of Π.

The increment instruction $(p, \text{ADD}(r), q, s)$ can be simulated directly by the rules $p \to qa_r$ and $p \to sa_r$.

The decrement instruction $(p, \text{SUB}(r), q)$ can be simulated by the following two rules: $pa_r \to q$, $p \to p$. Moreover, for every register symbol a_r with r not being an output register, we add the unit rules $a_r \to a_r$.

More concretely, this is how we define the components of Π:

$$
\begin{aligned}
\Pi &= (O, T, \{w_0\}, R, max), \text{ where} \\
O &= Q \cup \{a_r \mid r \in \mathcal{R}\}, \\
T &= \{a_r \mid r \in \mathcal{R}_o ut\}, \\
R &= \{p \to qa_r, p \to sa_r \mid (p, \mathrm{ADD}(r), q, s) \in \mathcal{P}\} \\
&\cup \{pa_r \to q, p \to p \mid (p, \mathrm{SUB}(r), q) \in \mathcal{P}\} \\
&\cup \{a_r \to a_r \mid r \in \mathcal{R} \setminus \mathcal{R}_o ut\} \\
&\cup \{h \to \lambda\}.
\end{aligned}
$$

where $\mathcal{P}$ is the set of instructions of the register machine, Q is the set of its instruction labels, $\mathcal{R}$ is the set of its registers, $\mathcal{R}_o ut \subseteq \mathcal{R}$ is the set of its output registers. We define w_0 as the multiset of the form $q_0 a_{r_1}^{r_1} \ldots a_{r_m}^{r_m}$, where $q_0 \in Q$ is the initial state of the machine, and $r_i, 1 \leq i \leq m = |\mathcal{R}|$, are the initial values of its registers.

In what follows, we describe the behavior of Π and prove that it indeed faithfully simulates the computations of the partially blind register machine.

If p and a copy of a_r find themselves in the same membrane, then a successful decrement can be simulated: the total number of copies of a_r in the system is reduced by one.

If there are no copies of a_r left in the system, then p only has the chance to be used with the unit rule $p \to p$; observe that in any derivation mode at least one rule has to be applied if the system is not halting, i.e., as long as there still is a rule which can be applied to some symbol. In this case, either $p \to p$ and/or some unit rule $a_r \to a_r$ can be applied in every future derivation step, hence, the computation will never halt.

If copies of a_r do appear in the system, but not in the membrane containing p, then p can use the unit rule $p \to p$, and in any derivation mode either only this rule and/or other unit rules $a_r \to a_r$ can be applied. If in some future step, p and a_r appear in the same membrane, possibly $pa_r \to q$ can be applied. Otherwise, again we obtain just non-halting computation branches.

However, there must exist another branch in which no splits and mergers have happened at all, i.e., p and a_r appear together in the same membrane, and in which the simulation therefore will be able to proceed correctly. The same alternative holds if p and a_r share the same membrane, but the system non-deterministically would choose to only apply $p \to p$ rather than $pa_r \to p$.

As soon as the halting label h appears, we have to use the final rule $h \to \lambda$. The final zero check is simulated by the unit rules $a_r \to a_r$ for all non-output registers r, which keep the computation going on forever if at least one such symbol a_r is still present. Observe that this argument does not depend on the distribution of the symbols in the membranes of a configuration, since $a_r \to a_r$ is a non-cooperative rule, and hence Remark 1 applies.

In sum, we conclude that the P system with reactive membranes Π can simulate the computations of the given partially blind register machine correctly, but on the other hand cannot yield more results. □

Finally, we remark that the construction we show here is non-deterministic, even if the simulated partially blind register machine is deterministic, i.e., all increment instructions are of the form $(p, \mathrm{ADD}(r), q, q)$, which in a simpler way can be written as $(p, \mathrm{ADD}(r), q)$.

5 Extensions

Given the motivation to use P systems with reactive membranes for thinking about the emergence of space and space separations in abiotic environments, and also the richness of the ecosystem of P systems variants, multiple extensions can be proposed.

A natural one to be considered would be *limiting the size* of individual membranes, as real membranes do not generally grow very big. Limitations on the number of symbols have already been considered in P systems [2], but combined with constant splitting and merging this ingredient may have a drastically different impact. It would be necessary to decide what happens when a membrane attains its maximal capacity. The approach in [2] is to prevent it to accept new symbols, but in the context of reactive membranes it may be appropriate to bias the splitting and merging stage of the computational step to force such full membranes to split. The contribution of such limitations to the computational power is yet unclear, but probably in some strong relation to the size of the left-hand sides of the evolution rules.

An extension in the spirit of generalized P systems [15] would be to *subject the rules to splitting and merging*: for example, a rule $u \to v$ could split into two rules $u \to \alpha$ and $\alpha \to v$, which could later merge back into $u \to v$. With such an extension, membranes would contain objects and rules, and splitting and merging would affect not only which symbols can interact, but also which rules will ensure their interaction. Similarly to splitting and merging of membranes, splitting and merging of rules may delay some interactions. Relevance to thinking about the origins of life and the computational power of this variant remain to be explored.

6 Conclusion and perspectives

This paper is a first attempt at using P systems for thinking about the origins of life, and in particular about the emergence of individual compartments separated by membranes. We introduced P systems with reactive membranes, in which every symbol is conceptually surrounded by elementary membranes, which then can merge to form bigger membranes, or split. Mimicking biochemistry, the set of rules is

common to all membranes—the differences in the processes in different membranes should come from the symbols. Cooperative rules are allowed, and probably even necessary to meaningfully implement distinctions between membranes.

It is still an open research direction to actually illustrate some processes believed to have happened during abiogenesis in P systems with reactive membranes. Perhaps the most promising would be to implement autocatalytic cycles (e.g., [13]). The next step would be to implement *self-replication*, as suggested by José M. Sempere in a discussion. Indeed, in P systems with reactive membranes the membrane structure emerges spontaneously, which makes them a promising candidate for implementing self-replication of something other than symbol objects.

A parallel research direction which we started to explore in this paper is the computational power of P systems with reactive membranes. We have shown here that splitting and merging do not affect the computational power of P systems with reactive membranes using non-cooperative rules—P systems with reactive membranes using non-cooperative rules have the same computational power as simple P systems provided we only start with one singleton multiset, no matter which derivation mode we use. Based on this result, we have shown that P systems with reactive membranes can characterize the family of Parikh sets of semilinear languages when using only non-cooperative rules in any derivation mode.

Finally, when cooperative rules are allowed, P systems with reactive membranes can generate all multiset languages generated by partially blind register machines.

Several questions still remain to be addressed, in particular: can splitting and merging *augment* the computational power? It would indeed be surprising, but it has already been shown that non-deterministic shuffling of rule right-hand sides allows for generating non-semilinear languages [1], meaning that random shuffling of symbol neighborhoods as described in this paper may boost the power of the variant in some specific cases.

A subtle aspect which we do not discuss in depth in this paper is the halting condition and the procedure for retrieving the result. There is an asymmetry between these two: halting occurs when no more evolution rules are applicable after all possible splits and mergers. On the other hand, getting out the result essentially happens by merging *all* membranes into a single one.

Since the computational results we give in this paper seem to depend directly on the halting condition and on the procedure for obtaining the result, it would be relevant to explore how slight variations in these two affect the computational power of P systems with reactive membranes.

Finally, in Sect. 5, we have suggested several possible extensions of the new variant. A formal exploration of the computational power of such extensions would be quite relevant. Even more importantly, it would be very relevant to identify which extensions are more useful for using P systems with reactive membranes in thinking about the origins of life.

Acknowledgements The authors would like to thank the Organizing Committee of the 19th Brainstorming Week on Membrane Computing (http://www.gcn.us.es/19bwmc) (BWMC 2023) for organizing this fruitful event, which allowed the authors to jointly develop the ideas presented in this paper. The authors would also like to thank José M. Sempere for multiple helpful suggestions about self-replication and emergence of space. Artiom Alhazov acknowledges project 20.80009.5007.22 "Intelligent information systems for solving ill-structured problems, processing knowledge and big data" by the National Agency for Research and Development. David Orellana-Martín and Antonio Ramírez-de-Arellano acknowledge the Zhejiang Lab BioBit Program (Grant No. 2022BCF05).

References

1. Alhazov, A., Freund, R., & Ivanov, S. (2020). P systems with randomized right-hand sides of rules. *Theoretical Computer Science, 805*, 144–160.
2. Alhazov, A., Freund, R., & Ivanov, S. (2021). P systems with limited number of objects. *Journal of Membrane Computing, 3*(1), 1–9.
3. Alhazov, A., Freund, R., Ivanov, S., Orellana-Martín, D., de Arellano, A.R., Rodríguez-Gallego, J.-A. (2023). P systems with reactive membranes. In *Proceedings of Nineteenth Brainstorming Week on Membrane Computing*, pages 9–20
4. Alhazov, A., Freund, R., Ivanov, S., Orellana-Martín, D., de Arellano, A.R., & Rodríguez-Gallego, J.-A. (2023). P systems with reactive membranes. In *Proceedings of the Twenty-fourth International Conference on Membrane Computing (CMC2023)*, pages 27–42
5. Alhazov, A., Freund, R., Ivanov, S., & Oswald, M. (2022). Variants of derivation modes for which purely catalytic P systems are computationally complete. *Theoretical Computer Science, 920*, 95–112.
6. Alhazov, A., Freund, R., Ivanov, S., & Verlan, S. (2021). Variants of derivation modes for which catalytic P systems with one catalyst are computationally complete. *Journal of Membrane Computing, 3*(4), 233–245.
7. Alhazov, A., Freund, R., Ivanov, S., & Verlan, S. (2022). Tissue P systems with vesicles of multisets. *International Journal of Foundations of Computer Science, 33*(3 &4), 179–202.
8. Alhazov, A., Freund, R., Verlan, S. (2016). P systems working in maximal variants of the set derivation mode. In Alberto Leporati, Grzegorz Rozenberg, Arto Salomaa, and Claudio Zandron, editors, *Membrane Computing – 17th International Conference, CMC 2016, Milan, Italy, July 25-29, 2016, Revised Selected Papers*, volume 10105 of *Lecture Notes in Computer Science*, pages 83–102. Springer
9. Alhazov, A., Ishdorj, T.-O. (2004). Membrane operations in P systems with active membranes. *Second Brainstorming Week on Membrane Computing*, pages 37–44
10. Aman, B., & Ciobanu, G. (2009). Simple, enhanced and mutual mobile membranes. *Transactions on Computational Systems Biology, 11*, 26–44.
11. Cardelli, L., & Păun, G. (2006). An universality result for a (mem) brane calculus based on mate/drip operations. *International Journal of Foundations of Computer Science, 17*(1), 49–68.

12. Cobb, M. (2017). 60 years ago, Francis Crick changed the logic of biology. *PLOS Biology*, 15(9):1–8, 09.
13. Damer, B., & Deamer, D. (2020). The hot spring hypothesis for an origin of life. *Astrobiology, 20*(4), 429–452.
14. Dassow, J., Păun, G. (1989). *Regulated Rewriting in Formal Language Theory*. Springer
15. Freund, R. (1999). Generalized P-systems. In Gabriel Ciobanu and Gheorghe Păun, editors, *Fundamentals of Computation Theory, 12th International Symposium, FCT '99, Iasi, Romania, August 30 – September 3, 1999, Proceedings*, volume 1684 of *Lecture Notes in Computer Science*, pages 281–292. Springer,
16. Freund, R., Leporati, A., Mauri, G., Porreca, A.E., Verlan, S., & Zandron, C. (2014). Flattening in (tissue) P systems. In Artiom Alhazov, Svetlana Cojocaru, Marian Gheorghe, Yurii Rogozhin, Grzegorz Rozenberg, and Arto Salomaa, editors, *Membrane Computing*, volume 8340 of *Lecture Notes in Computer Science*, pages 173–188. Springer
17. Freund, R., Oswald, M. (2006). Tissue P systems and (mem)brane systems with mate and drip operations working on strings. In Nadia Busi and Claudio Zandron, editors, *Proceedings of the First Workshop on Membrane Computing and Biologically Inspired Process Calculi, MeCBIC@ICALP 2006, Venice, Italy, July 9, 2006*, volume 171 (2) of *Electronic Notes in Theoretical Computer Science*, pages 105–115. Elsevier,
18. Freund, R., Verlan, S. (2007). A formal framework for static (tissue) P systems. In George Eleftherakis, Petros Kefalas, Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa, editors, *Membrane Computing, 8th International Workshop, WMC 2007, Thessaloniki, Greece, June 25-28, 2007. Revised Selected and Invited Papers*, volume 4860 of *Lecture Notes in Computer Science*, pages 271–284. Springer
19. Glade, N. (2022).*Le Vivant Rare, Faible et Amorphe - Évolution depuis les Origines jusqu'à la Vie telle qu'elle nous Apparaît. (Rare, Weak and Amorphous Life – Evolution of Life from the Origins until Life as it Appears Nowadays)*. HAL Open Archive
20. Bulletin of the International Membrane Computing Society (IMCS). http://membranecomputing.net/IMCSBulletin/index.php.
21. Ivanov, S., Rogozhin, Y., & Verlan, S. (2014). Small universal networks of evolutionary processors. *Journal of Automata, Languages and Combinatorics, 19*(1–4), 133–144.
22. Shankara Narayanan Krishna and Gheorghe Păun. (2005). P systems with mobile membranes. *Natural Computing4*(3), 255–274.
23. Orellana-Martín, D., Valencia-Cabrera, L., & Pérez-Jiménez, M.J. (2022). P systems with evolutional communication and separation rules. In Jérôme Durand-Lose and György Vaszil, editors, *Machines, Computations, and Universality – 9th International Conference, MCU 2022, Debrecen, Hungary, August 31 – September 2, 2022, Proceedings*, volume 13419 of *Lecture Notes in Computer Science*, pages 143–157. Springer
24. Pan, L., & Ishdorj, T.-O. (2004). P systems with active membranes and separation rules. *Journal of Universal Computer Science, 10*(5), 630–649.
25. Păun, G. (2000). Computing with membranes. *Journal of Computer and System Sciences, 61*(1), 108–143.
26. Păun, G., Rozenberg, G., Salomaa, A. (2010). editors. *The Oxford Handbook of Membrane Computing*. Oxford University Press
27. Rozenberg, G., Salomaa, A. (1997). editors. *Handbook of Formal Languages*. Springer
28. Segretain, Rémi, Ivanov, Sergiu, Trilling, Laurent, & Glade, Nicolas. (2020). A methodology for evaluating the extensibility of Boolean networks' structure and function. In Rosa M. Benito, Chantal Cherifi, Hocine Cherifi, Esteban Moro, Luis Mateus Rocha, and Marta Sales-Pardo, editors, *Complex Networks & Their Applications IX - Volume 2, Proceedings of the Ninth International Conference on Complex Networks and Their Applications, COMPLEX NETWORKS 2020, 1–3 December 2020, Madrid, Spain*, volume 944 of *Studies in Computational Intelligence*, pages 372–385. Springer
29. Segretain, Rémi, Trilling, Laurent, Glade, Nicolas, & Ivanov, Sergiu. (2021). Who plays complex music? On the correlations between structural and behavioral complexity measures in sign Boolean networks. In *21st IEEE International Conference on Bioinformatics and Bioengineering, BIBE 2021, Kragujevac, Serbia, October 25–27, 2021*, pages 1–6. IEEE