P Systems with Randomized Right-hand Sides of Rules

Artiom Alhazov[1,2*], Rudolf Freund[3], and Sergiu Ivanov[4,5]

[1] Institute of Mathematics and Computer Science
Academy of Sciences of Moldova
Academiei 5, Chişinău, MD-2028, Moldova
artiom@math.md

[2] Key Laboratory of Image Information Processing
and Intelligent Control of Education Ministry of China
School of Automation,
Huazhong University of Science and Technology
Wuhan 430074, China

[3] Faculty of Informatics, TU Wien
Favoritenstraße 9–11, 1040 Vienna, Austria
rudi@emcc.at

[4] LACL, Université Paris Est – Créteil Val de Marne
61, av. Général de Gaulle, 94010, Créteil, France
sergiu.ivanov@u-pec.fr

[5] TIMC-IMAG/DyCTiM, Faculty of Medicine of Grenoble
5 avenue du Grand Sablon, 38700, La Tronche, France
sergiu.ivanov@univ-grenoble-alpes.fr

Summary. P systems are a model of hierarchically compartmentalized multiset rewriting. We introduce a novel kind of P systems in which rules are dynamically constructed in each step by non-deterministic pairing of left-hand and right-hand sides. We define three variants of right-hand side randomization and compare each of them with the power of conventional P systems. It turns out that all three variants enable non-cooperative P systems to generate exponential (and thus non-semi-linear) number languages. We also give a binary normal form for one of the variants of P systems with randomized rule right-hand sides. Finally, we also discuss extensions of the three variants to tissue P systems, i.e., P systems on an arbitrary graph structure.

* The work is supported by National Natural Science Foundation of China (61320106005 and 61033003) and the Innovation Scientists and Technicians Troop Construction Projects of Henan Province (154200510012).

1 Introduction

Membrane computing is a research field originally founded by Gheorghe Păun in 1998, see [13]. Membrane systems (also known as P systems) are a model of computing based on the abstract notion of a membrane. Formally, a membrane is treated as a container delimiting a region; a region may contain objects which are acted upon by the rewriting rules associated with the membranes. Quite often, the objects are plain symbols coming from a finite alphabet, but P systems operating on more complex objects (e.g., strings, arrays) are often considered, too, e.g., see [9].

A comprehensive overview of different flavors of membrane systems and their expressive power is given in the handbook which appeared in 2010, see [14]. For a state of the art snapshot of the domain, we refer the reader to the P systems website [17], as well as to the bulletin of the International Membrane Computing Society [16].

Dynamic evolution of the set of available rules has been considered from the very beginning of membrane computing. Already in 1999, generalized P systems were introduced in [8]; in these systems the membranes, alongside the objects, contain *operators* which act on these objects, while the P system itself acts on the operators, thereby modifying the transformations which will be carried out on the objects in the subsequent steps. Among further ideas on dynamic rules, one may list rule creation [4], activators [1], inhibiting/deinhibiting rules [7], and symport/antiport of rules [6]. One of the more recent developments in this direction are *polymorphic P systems* [2, 3, 12], in which rules are defined by pairs of membranes, whose contents may be modified by moving objects in or out.

We remark that the previous studies on dynamic rule sets either treated the rules as atomic entities (symport/antiport of rules, operators in generalized P systems), or allowed virtually unlimited possibilities of tampering with their shape (polymorphic P systems). In the present work, we propose a yet different approach which can be seen as an intermediate one.

In *P systems with randomized rule-right-hand sides* (or with randomized RHS, for short), the available left-hand sides and right-hand sides of rules are fixed, but the associations between them are *re-evaluated in every step*: a left-hand side may pick a right-hand side arbitrarily (randomly). In Section 3, we present three different formal definitions of this intuitive idea of randomized RHS:

1. rules *exchange* their RHS,
2. each rule randomly picks an RHS from a *common* collection of RHS, *shared* between the rules,
3. each rule randomly picks an RHS from a possible collection of *RHS associated with the rule itself.*

P systems with randomized RHS may have a real-world (possibly biological) application for representing systems in a hostile environment. The modifications such P systems effect on their rules may be used to represent perturbations caused by the environment (mutations), somewhat in the spirit of faulty Turing machines (e.g., see [5]).

In this article, we will focus on the expressive power of P systems with randomized RHS, as well as on comparing them to the classical model with or without cooperative rules. One of the central conclusions of the present work is that non-cooperative P systems with randomized RHS can generate *exponential* number languages, thus (partially) surpassing the power of conventional P systems.

This paper is structured as follows. Section 2 recalls some preliminaries about multisets, strings, permutations, as well as conventional P systems. Section 3 defines the three variants of RHS randomization. Section 4 discusses the computational power of the three variants of P systems with randomized RHS. Section 5 shows a binary normal form for one of the variants of P systems with randomized RHS. Section6 discusses extensions of the three variants of RHS randomization to tissue P systems. Finally, Section 7 summarizes the results of the article and gives some directions for future work.

2 Preliminaries

In this paper, the set of positive natural numbers $\{1, 2, \dots\}$ is denoted by $\mathbb{N}^+$, the set of natural numbers also containing 0, i.e., $\{0, 1, 2, \dots\}$, is denoted by $\mathbb{N}$. Given $k \in \mathbb{N}^+$, we will call the set ${\mathbb{N}^+}_k = \{x \in \mathbb{N}^+ \mid 1 \leq x \leq k\}$ an *initial segment* of $\mathbb{N}^+$.

An *alphabet* V is a finite set. The families of recursively enumerable, context-free, linear, and regular languages, and of languages generated by tabled Lindenmayer systems are denoted by RE, CF, LIN, REG, and $ET0L$, respectively. The families of sets of Parikh vectors as well as of sets of natural numbers (multiset languages over one-symbol alphabets) obtained from a language family F are denoted by PsF and NF, respectively.

For further introduction to the theory of formal languages and computability, we refer the reader to [14, 15].

2.1 Linear Sets over $\mathbb{N}$

A *linear* set over $\mathbb{N}$ generated by a set of vectors $A = \{\mathbf{a}_i \mid 1 \leq i \leq d\} \subset_{fin} \mathbb{N}^n$ (here $A \subset_{fin} B$ indicates that A is a finite subset of B) and an offset $\mathbf{a}_0 \in \mathbb{N}^n$ is defined as follows:

$$\langle A, \mathbf{a}_0 \rangle_{\mathbb{N}} = \left\{ \mathbf{a}_0 + \sum_{i=1}^{d} k_i \mathbf{a}_i \,\middle|\, k_i \in \mathbb{N},\ 1 \leq i \leq d \right\}.$$

If the offset $\mathbf{a}_0$ is the zero vector $\mathbf{0}$, we call the corresponding linear set *homogeneous*; we also use the short notation $\langle A \rangle_{\mathbb{N}} = \langle A, \mathbf{0} \rangle_{\mathbb{N}}$.

We use the notation $\mathbb{N}^n LIN_{\mathbb{N}} = \{\langle A, \mathbf{a}_0 \rangle_{\mathbb{N}} \mid A \subset_{fin} \mathbb{N}^n,\ \mathbf{a}_0 \in \mathbb{N}^n\}$, to refer to the class of all linear sets of n-dimensional vectors over $\mathbb{N}$. Semi-linear sets are defined as finite unions of linear sets. We use the notation $\mathbb{N}^n SLIN_{\mathbb{N}}$ to refer to the classes of semi-linear sets of n-dimensional vectors. In case no restriction is imposed on the dimension, n is replaced by $*$. We may omit n if $n = 1$. A finite union of linear sets which only differ in the starting vectors is called *uniform* semilinear:

$$\mathbb{N}^n SLIN_{\mathbb{N}}^U = \left\{ \bigcup_{\mathbf{b} \in B} \langle A, \mathbf{b} \rangle_{\mathbb{N}} \mid A \subset_{fin} \mathbb{N}^n,\ B \subset_{fin} \mathbb{N}^n \right\}$$

Let us denote such a set by $\langle A, B \rangle_{\mathbb{N}}$.

Note that a uniform semilinear set $\langle A, B \rangle_{\mathbb{N}}$ can be seen as a pairwise sum of the finite set B and the homogeneous linear set $\langle A \rangle_{\mathbb{N}}$:

$$\langle A, B \rangle_{\mathbb{N}} = \{ \mathbf{a} + \mathbf{b} \mid \mathbf{a} \in \langle A \rangle_{\mathbb{N}}, \mathbf{b} \in B \}.$$

This observation immediately yields the conclusion that the sum of two uniform semilinear sets $\langle A_1, B_1 \rangle_{\mathbb{N}}$ and $\langle A_2, B_2 \rangle_{\mathbb{N}}$ is uniform semilinear as well and can be computed in the following way:

$$\langle A_1, B_1 \rangle_{\mathbb{N}} + \langle A_2, B_2 \rangle_{\mathbb{N}} = \{ \mathbf{a} + \mathbf{b} \mid \mathbf{a} \in \langle A_1 \cup A_2 \rangle_{\mathbb{N}}, \mathbf{b} \in B_1 + B_2 \}.$$

As is folklore,

$$PsCF = PsLIN = PsREG = \mathbb{N}^* SLIN_{\mathbb{N}}.$$

2.2 Multisets

A *multiset* over V is any function $w : V \to \mathbb{N}$; $w(a)$ is the *multiplicity* of a in w. A multiset w is often represented by one of the strings containing exactly $w(a)$ copies of each symbol $a \in V$. The set of all multisets over the alphabet V is denoted by V°. By abusing string notation, the empty multiset is denoted by λ. The *projection* (restriction) of w over a sub-alphabet $V' \subseteq V$ is the multiset $w|_{V'}$ defined as follows:

$$w|_{V'}(a) = \begin{cases} w(a), & a \in V'; \\ 0, & a \in V \setminus V'. \end{cases}$$

Example 1. The string aab can represent the multiset $w : \{a, b\} \to \mathbb{N}$ with $w(a) = 2$ and $w(b) = 1$. The projection $w|_{\{a\}} = w'$ is defined as $w'(a) = w(a) = 2$ and $w'(b) = 0$.

We will (ab)use the symbol $\in$ to denote the relation "is a member of" for multisets. Therefore, for a multiset w, $a \in w$ will stand for $w(a) > 0$.

2.3 Strings and Permutations

A (non-empty) *string* s over an alphabet V traditionally is defined as a finite ordered sequence of elements of V. Equivalently, we can define a string of length

k as a function assigning symbols to positions: $s : \mathbb{N}^+{}_k \to V$. Thus, the string $s = aab$ can be equivalently defined as the function $s : \mathbb{N}^+{}_3 \to \{a, b\}$ with $s(1) = a$, $s(2) = a$, and $s(3) = b$. We will use the traditional notation $|s|$ to refer to the length of the string s (i.e., the size k of the initial segment $\mathbb{N}^+{}_k$ it is defined on). In addition, the size of the empty string λ is 0.

A string $s : \mathbb{N}^+{}_k \to V$ is not necessarily surjective (there may be symbols from V that do not appear in s). We will use the notation $set(s)$ to refer to the set of symbols appearing in s (the image of s):

$$set(s) = \left\{a \in V \mid a = s(i) \text{ for some } i \in \mathbb{N}^+{}_{|s|}\right\}.$$

Given a string $s : \mathbb{N}^+{}_k \to V$, a *prefix* of length $k' \leq k$ of s is the restriction of s to $\mathbb{N}^+{}_{k'} \subseteq \mathbb{N}^+{}_k$. For example, aa is a prefix of length 2 of the string aab. We will use the notation $\mathrm{pref}_{k'}(s)$ to denote the prefix of length k' of s.

Given a finite set A, a *permutation* of A is any bijection $\rho : A \to A$. Given a permutation $\sigma : \mathbb{N}^+{}_k \to \mathbb{N}^+{}_k$ and a string $s : \mathbb{N}^+{}_k \to V$ of length k, *applying* σ *to* s is defined as $\sigma(s) = s \circ \sigma$ (where $\circ$ is the function composition operator).

Example 2. Following the widespread tradition, we will write permutations in Cauchy's two-line notation. The permutation σ_{rev} of $\mathbb{N}^+{}_3$ which "reverses the order" of the numbers, can be written as follows:

$$\sigma_{rev} = \begin{pmatrix} 1 & 2 & 3 \\ 3 & 2 & 1 \end{pmatrix}.$$

Applying σ_{rev} to a string reverses it:

$$\sigma_{rev}(aab) = baa.$$

Any finite set B trivially can be represented by one of the strings listing all of its elements exactly once. All such strings are equivalent modulo permutations. Given a fixed enumeration $B = \{b_1, \ldots, b_n\}$, we define the *canonical string representation* of B to be the string $\delta(B) = b_1 \ldots b_n$.

2.4 Rule Sides

We consider arbitrary labeled multiset rules $r : u \to v$ over an alphabet V, where r is the rule label we attach for convenience, and u and v are strings over V representing multisets. As usual, the application of such a rule means replacing the multiset represented by u by the multiset represented by v.

For a given rule $r : u \to v$, we define the left-hand-side and the right-hand-side functions as follows:

$$\begin{aligned} lhs(u \to v) &= lhs(r) = (u), \\ rhs(u \to v) &= rhs(r) = (v). \end{aligned}$$

Using the brackets (and), for a given string w, the notation (w) is used to describe the multiset represented by w. As usual, we will extend the notations for these functions lhs and rhs lifted to sets of rules: given a set of rules R, $lhs(R) = \{lhs(r) \mid r \in R\}$ and $rhs(R) = \{rhs(r) \mid r \in R\}$. Furthermore, for any *string* (finite ordered sequence) of rules $\rho : \mathbb{N}^+{}_k \to R$ we define the strings of left-hand sides $lhs(\rho) = lhs \circ \rho$ and of right-hand sides $rhs(\rho) = rhs \circ \rho$.

Example 3. Take $R = \{r_1 : aa \to ab, r_2 : cc \to cd\}$ and consider the string of rules $\rho = r_1 r_1 r_2$. Then $lhs(\rho) = (aa)(aa)(cc)$ and $rhs(\rho) = (ab)(ab)(cd)$. Thus, $lhs(\rho)$ and $rhs(\rho)$ can be considered as *strings of multisets*.

We will (ab)use the symbol $\to$ for *combining* two strings of multisets $\alpha, \beta : \mathbb{N}^+{}_k \to V^\circ$ of the same length k. The *string* $\alpha \to \beta$ will be defined as follows, for any $i \in \mathbb{N}^+{}_k$:

$$(\alpha \to \beta)(i) = \alpha(i) \to \beta(i).$$

Example 4. Consider the following two strings of multisets: $\alpha = (aa)(aa)(cc)$ and $\beta = (ab)(ab)(cd)$. $\alpha \to \beta$ is simply the string of rules that can be obtained by taking the multisets from α as left-hand sides and β as right-hand sides, in the given order: $\alpha \to \beta = (aa) \to (ab)(aa) \to (ab)(cc) \to (cd)$ (which exactly corresponds with ρ from Example 3).

2.5 (Hierarchical) P Systems

A (hierarchical) *P system* is a construct

$$\Pi = (O, T, \mu, w_1, \dots, w_n, R_1, \dots R_n, h_i, h_o),$$

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 $\{1, \dots, n\}$ and usually given by a sequence of correctly nested brackets, w_i are the multisets giving the initial contents of each membrane i ($1 \le i \le n$), R_i is the finite set of rules associated with membrane i ($1 \le i \le n$), and h_i and h_o are the labels of the input and the output membranes, respectively ($1 \le h_i \le n$, $1 \le h_o \le n$).

In the present work, we will mostly consider the *generative case*, in which Π will be used as a multiset language-generating device. We therefore will systematically omit specifying the input membrane h_i.

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 $u \in O^o \setminus \{\lambda\}$ and $v \in O^o$. If $|u| = 1$, the rule $u \to v$ is called *non-cooperative*; otherwise it is called *cooperative*. Rules may additionally be allowed to send symbols to the neighboring membranes. In this case, for rules in R_i, $v \in O \times Tar_i$, where Tar_i contains the targets *out* (corresponding to sending the symbol to the parent membrane), *here* (indicating that the symbol should be kept in membrane i), and in_h (indicating that the symbol should be sent into the child membrane h of membrane i). Note that all variants of the function rhs, as well as the operator $\to$ from the previous section can be naturally extended to rules having right-hand sides with target indications (from $O \times Tar_i$).

In P systems, rules are often applied in the maximally parallel way: in any derivation step, a non-extendable multiset of rules has to 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 the P system choosing non-deterministically between the maximal collections of rules applicable in one step.

A computation of a P system is traditionally considered to be a sequence of configurations it can successively pass through, 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 of a P system Π as defined above is the contents of the output membrane h_o projected over the terminal alphabet T.

Example 5. For readability, we will often prefer a graphical representation of P systems. For example, the P system $\Pi_1 = (\{a, b\}, \{b\}, [_1\ ]_1, a, R, 1)$ with the rule set $R = \{a \to aa, a \to b\}$ may be depicted as in Figure 1.

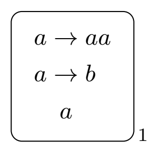

Fig. 1. The example P system Π_1

Due to maximal parallelism, at every step Π_1 may double some of the symbols a, while rewriting some other instances into b.

Note that, even though Π_1 might express the intention of generating the set of numbers of the powers of two, it will actually generate the whole of $\mathbb{N}^+$ (due to halting). Indeed, for any $n \in \mathbb{N}^+$, a^n can be generated in n steps by choosing to apply, in the first $n - 1$ steps, $a \to aa$ to exactly one instance of a and $a \to b$ to all the other instances, and by applying $a \to b$ to every a in the last step (in fact, for $n > 1$, in each step except the last one, in which $a \to b$ is applied twice, both rules are applied exactly once, as exactly two symbols a are present, whereas all other symbols are copies of b).

While maximal parallelism and halting by inapplicability are staple ingredients, various other derivation modes and halting conditions have been considered for P systems, e.g., see [14].

We will use the notation $OP_n(coo)$ to denote the family of P systems with at most n membranes, with cooperative rules. To denote the family of such P systems with *non-cooperative* rules, we replace *coo* by *ncoo*. To denote the family of languages of multisets generated by these P systems, we prepend Ps to the notation, and to denote the family of the generated number languages, we prepend N.

3 P Systems with Randomized RHS

In this section we consider three different variants of defining P systems with randomized RHS. We immediately point out that, despite the common intuitive background, the details of the resulting semantics vary quite a lot.

3.1 Variant 1: Random RHS Exchange

In this variant of P systems, rules randomly exchange right-hand sides at the beginning of every evolution step. This variant was the first to be conceived and is the closest to the classical definition.

A *P system with random RHS exchange* is a construct

$$\Pi = (O, T, \mu, w_1, \dots, w_n, R_1, \dots R_n, h_o),$$

where the components of the tuple are defined as in the classical model (Section 2.5).

As different from conventional P systems, Π does not apply the rules from R_i directly. Instead, for each membrane $1 \leq i \leq n$, we take the canonical representation of R_i, i.e., $\delta(R_i)$, and non-deterministically (randomly) choose a permutation $\sigma : \mathbb{N}^+{}_{|R_i|} \to \mathbb{N}^+{}_{|R_i|}$ to compute the canonical representation of R_i^σ from $\delta(R_i)$ as follows:

$$\delta(R_i^\sigma) = lhs(\delta(R_i)) \to \sigma(rhs(\delta(R_i))).$$

We now extract the set of rules $R_i^\sigma = set(\delta(R_i^\sigma))$ described by the string $\delta(R_i^\sigma)$ as constructed above. Π will then apply the rules from R_i^σ according to the usual maximally parallel semantics in membrane i.

In other words, Π *non-deterministically permutes* the right-hand sides of rules in each membrane i, and then applies the obtained rules according to the maximally parallel semantics.

Note that we first have to transform the set R_i into its canonical string representation $\delta(R_i)$ in order to be able to obtain a correct representation of the $|R_i|$ rules and from that a correct representation of the $|R_i|$ rules in R_i^σ, even if the number of different left-hand sides and/or different right-hand sides of rules does not equal $|R_i|$.

Example 6. Consider the P system $\Pi_2 = (\{a, b\}, \{b\}, [_1\ ]_1, a, R, 1)$ with the rule set $R = \{a \to aa, c \to b\}$. Π_2 is graphically represented in Figure 2.

The number language generated by Π_2 (the set of numbers of instances of b that may appear in the skin after Π_2 has halted) is exactly $\{2^n \mid n \in \mathbb{N}^+\}$. Indeed, while Π_2 applies the identity permutation on the right-hand sides, $a \to aa$ will double the number of symbols a, while the rule $c \to b$ will never be applicable. When Π_2 exchanges the right-hand sides of the rules, the rule $a \to b$ will rewrite every symbol a into a symbol b. After this has happened, no rule will ever be applicable any more and Π_2 will halt with 2^n symbols b in the skin, where $n + 1$ is the number of computation steps taken.

$$\left[\begin{array}{c} a \to aa \\ c \to b \\ a \end{array}\right]_1$$

Fig. 2. The P system Π_2 with random RHS exchange generating the number language $\{2^n \mid n \in \mathbb{N}\}$.

We will use the notation

$$OP_n(rhsExchange, coo)$$

to denote the family of P systems with random RHS exchange, with at most n membranes, with cooperative rules. To denote the family of such P systems with *non-cooperative* rules, we replace *coo* by *ncoo*. To denote the family of languages of multisets generated by these P systems, we prepend Ps to the notation, and to denote the family of the generated number languages, we prepend N.

3.2 Variant 2: Randomized Pools of RHS

In this variant of P systems, every membrane has some fixed left-hand sides and a *pool* of available right-hand sides to build rules from. An RHS from the pool can only be used once.

A *P system with randomized pools of RHS* is a construct

$$\Pi = (O, T, \mu, w_1, \dots, w_n, H_1, \dots H_n, h_o),$$

where H_i defines the left- and right-hand sides available in membrane i and the other components of the tuple are defined as in the classical model (Section 2.5).

For $1 \leq i \leq n$, $H_i = (l_i, r_i)$ is a pair of strings of multisets over O. The string r_i may contain target indications (i.e., be a string of multisets over $O \times Tar_i$). The strings l_i and r_i are not necessarily of the same length. The length of the shortest of the two strings l_i and r_i is denoted by

$$k_i = \min(|l_i|, |r_i|).$$

At the beginning of every computation step in Π, for every membrane i, we construct the set of rules it will apply in the following way:

1. non-deterministically choose two (random) permutations
$$\sigma_l : \mathbb{N}^+{}_{|l_i|} \to \mathbb{N}^+{}_{|l_i|}, \quad \sigma_r : \mathbb{N}^+{}_{|r_i|} \to \mathbb{N}^+{}_{|r_i|};$$
2. take the first k_i elements out of $\sigma_l(l_i)$ and $\sigma_r(r_i)$:
$$l_i' = \mathrm{pref}_{k_i}(\sigma_l(l_i)), \quad r_i' = \mathrm{pref}_{k_i}(\sigma_r(r_i));$$

3. construct the set of rules R_i to be applied in membrane i by combining the left- and right-hand sides from l'_i and r'_i:

$$R_i = set(l'_i \to r'_i).$$

In step (3), we combine the strings l'_i and r'_i using the operator $\to$ defined in Subsection 2.4 and then apply the operator *set* to obtain the corresponding set of rules from the string representation.

After having constructed the set R_i for each membrane i, Π will proceed to applying the obtained rules according to the usual maximally parallel semantics.

When computing the strings l'_i and r'_i, we apply *two* different permutations σ_l and σ_r to l_i and r_i, in order to ensure fairness for the participation of left-hand and right-hand sides when $|l_i| \neq |r_i|$. For example, if we only permuted r_i in the case in which $|l_i| > |r_i|$, the left-hand sides located at positions $k > |r_i|$ in l_i would never be used.

We do not explicitly prohibit repetitions in l_i or in r_i, but we avoid repeated rules by constructing R_i using the *set* function.

Example 7. Consider the following P system with randomized pools of RHS: $\Pi_3 = (\{a, b\}, \{b\}, [_1\ ]_1, a, H, 1)$, with $H = \big((a), (aa)(b)\big)$; (a) stands for the multiset containing an instance of a, while $(aa)(b)$ is the string denoting the two multisets (aa) and (b). The graphical representation of Π_3 is given in Figure 3.

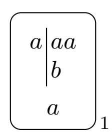

Fig. 3. The P system Π_3 with randomized pools of RHS generating the number language $\{2^n \mid n \in \mathbb{N}\}$.

The pair $H = (l, r)$ of strings of multisets is represented by listing the multisets of l and r in two columns and by drawing a vertical line between the two columns.

Π_3 follows exactly the same pattern as Π_2 from Example 6: while the identity permutation is applied to r, Π_3 keeps doubling the symbols a in the skin. Once the multisets (aa) and (b) are permuted in r, and thus the rule $a \to b$ is formed, all symbols a are rewritten into symbols b in one step and Π_3 must halt. Note that randomly taking the right-hand sides from a given pool avoids having the extra dummy rule $c \to b$ in Π_2.

We will use the notation

$$OP_n(rhsPools, coo)$$

to denote the family of P systems with randomized pools of RHS, with at most n membranes, with cooperative rules. To denote the family of such P systems with *non-cooperative* rules, we replace *coo* by *ncoo*. To denote the family of languages of multisets generated by these P systems, we prepend Ps to the notation, and to denote the family of the generated number languages, we prepend N.

3.3 Variant 3: Individual Randomized RHS

In this variant of P systems, each rule is constructed from a left-hand side and a set of possible right-hand sides.

A *P system with individual randomized RHS* is a construct

$$\Pi = (O, T, \mu, w_1, \ldots, w_n, P_1, \ldots P_n, h_o),$$

where P_i is the set of *productions* associated with the membrane i and the other components of the tuple are defined as in the classical model (Section 2.5).

A production is a pair $u \to R$, where $u \in O^\circ$ is the left-hand side and $R \subseteq O^\circ$ is a finite set of right-hand sides. The right-hand sides in R may have target indications, i.e., for a production in membrane i, we may consider $R \subseteq (O \times Tar_i)^\circ$. At the beginning of each computation step, for every membrane i, for each production $u \to R \in R_i$, Π will non-deterministically (randomly) pick a right-hand side v from R and will construct the rule $u \to v$ (this happens once per production). Π will then apply the rules thus constructed according to the maximally parallel semantics.

Example 8. Generating the language of the powers of two is the easiest compared with Variants 1 and 2. Indeed, consider the P system with individual randomized RHS $\Pi_4 = (\{a, b\}, \{b\}, [_1\ ]_1, a, P, 1)$ with only one production: $P = \{a \to \{aa, b\}\}$. Its graphical representation is given in Figure 4.

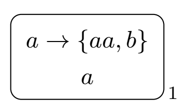

Fig. 4. The P system Π_4 with individual randomized RHS generating the number language $\{2^n \mid n \in \mathbb{N}\}$.

Π_4 works exactly like Π_2 and Π_3 from Examples 6 and 7: it doubles the number of symbols a and halts by rewriting them to b in the last step.

We will use the notation

$$OP_n(rndRhs, coo)$$

to denote the family of P systems with individual randomized RHS, with at most n membranes, with cooperative rules. To denote the family of such P systems with *non-cooperative* rules, we replace *coo* by *ncoo*. To denote the family of languages of multisets generated by these P systems, we prepend Ps to the notation, and to denote the family of the generated number languages, we prepend N.

We will sometimes want to set an upper bound k on the number of right-hand sides per production. To refer to the family of P systems with individual randomized RHS with such an upper bound, we will replace *rndRhs* by $rndRhs^k$ in the notation above.

3.4 Halting with Randomized RHS

The conventional (total) halting condition for P systems can be naturally lifted to randomized RHS: a P system Π with randomized RHS (Variant 1, 2, or 3) halts on a configuration C if, however it permutes rule right-hand sides in Variant 1, or however it builds rules out of the available rule sides in Variants 2 and 3, no rule can be applied in C, in any membrane.

Note that, for Variants 1 and 3, the permutations chosen do *not* affect the applicability of rules, because applicability only depends on left-hand sides, which are always the same in any membrane. The situation is different for Variant 2, because the number of available left-hand sides in a membrane of Π may be bigger than the number of available right-hand sides. Therefore, if Π is a P system with randomized pools of RHS, the way rule sides are permuted may affect the number of rules applicable in a given configuration. This is why, for Π to halt on C, we require no rule to be applicable for any permutation.

In this paper, we will mainly consider P systems with randomized pools of RHS in which, in every membrane, there are at least as many right-hand sides as there are left-hand sides. To refer to P systems with this restriction, we will use the notation $rhsPools'$. In these systems, the problem with the applicability of rules as described above can be avoided.

3.5 Equivalence Between Variants 1 and 2

Before discussing the computational power of the P systems with randomized RHS in general, we will briefly point out a strong relationship between P systems with random RHS exchange and P systems with randomized pools of RHS, *with* the restriction that every membrane contains at least as many right-hand sides as it has left-hand sides, i.e., for P systems with randomized RHS of type $rhsPools'$.

Theorem 1. *For any $k \in \{coo, ncoo\}$, the following holds:*

$$PsOP_n(rhsExchange, k) = PsOP_n(rhsPools', k).$$

Proof. Any membrane with random RHS exchange trivially can be transformed into a membrane with randomized pools of RHS by listing the left-hand sides of the rules in the pool of LHS and the right-hand sides of the rules in the pool of RHS.

Conversely, consider a membrane i with randomized pools of RHS, with the string l_i of LHS and the string r_i of RHS, $|l_i| \leq |r_i|$. We can transform it into a membrane with random RHS exchange as follows. For every LHS u from l_i, pick (and remove) an RHS v from r_i, and construct the rule $u \to v$. According to our supposition, we will exhaust the LHS before (or at the same time as) the RHS. For every RHS v' which is left, we add a new (dummy) symbol z' to the alphabet, and add the rule $z' \to v'$. Since the symbol z' is new and does not appear in any RHS, it will never be produced and the rule $z' \to v'$ will essentially serve as a stash for the RHS v'. □

3.6 Flattening

The folklore flattening construction (see [14] for several examples as well as [10] for a general construction) is quite directly applicable to P systems with individual randomized RHS.

Proposition 1 (flattening). *For any $k \in \{coo, ncoo\}$, the following is true:*

$$PsOP_1(rndRhs, k) = PsOP_n(rndRhs, k).$$

Proof (sketch). Since in the case of individual randomized RHS, randomization has per rule granularity (whereas in the other two variants randomization occurs at the level of membranes), we can simulate multiple membranes by attaching membrane labels to symbols. For example, a production $ab \to \{cd, f\}$ in membrane h becomes $a_h b_h \to \{c_h d_h, f_h\}$, while the send-in production $a \to \{(b, in_i), (b, in_j)\}$ becomes $a_h \to \{b_i, b_j\}$. □

On the other hand, for Variants 1 and 2 similar results cannot be proved in such a way, a situation which happens very seldom in the area of P systems, especially in the case of variants of the standard model. Yet intuitively, it is easy to understand why this happens, as in both Variants 1 and 2 the right-hand sides in just one membrane can randomly be chosen for any left-hand side, whereas different membranes can separate the possible combinations of left-hand sides and right-hand sides of rules. A formal proof showing that flattening is impossible for the types *rhsExchange* and *rhsPools'* will be given in the succeeding section by constructing a suitable example.

4 Computational Power of Randomized RHS

In this section, we look into the computational power of the three different versions of P systems with randomized right-hand sides. We first shortly consider the case of cooperative rules and then focus on the case of non-cooperative rules.

4.1 Cooperative Rules

The following result concerning the relationship between P systems with individual randomized RHS and conventional P systems holds for both cooperative and non-cooperative rules:

Proposition 2. *For any* $n \in \mathbb{N}^+$ *and* $\alpha \in \{ncoo, coo\}$, $PsOP_n(rndRhs, \alpha) \supseteq PsOP_n(\alpha)$.

Proof. Any conventional P system can be trivially seen as a P system with individual randomized RHS in which every production has exactly one right-hand side. □

Now, the computational completeness of *cooperative* P systems trivially implies the computational completeness of P systems with individual randomized RHS.

Corollary 1. *For any* $n \in \mathbb{N}^+$, $PsOP_n(rndRhs, coo) = PsRE$.

4.2 Non-cooperative Rules

First we mention an upper bound for the families $PsOP_n(\rho, ncoo)$, for any variant $\rho \in \{rhsExchange, rhsPools', rndRhs\}$:

Proposition 3. *For any* $n \in \mathbb{N}^+$ *and* $\rho \in \{rhsExchange, rhsPools', rndRhs\}$,

$$PsOP_n(\rho, ncoo) \subseteq PsET0L.$$

Proof. No matter how the rule sets are constructed in the three different variants, we always get a finite set of different sets of rules—*tables*—corresponding to tables in $ET0L$-systems, which can also mimic the contents of different membranes in the usual way by using symbols marked with the corresponding membrane label. □

Next we show one of the central results of this paper: randomized rule right-hand sides allow for generating *non-semilinear languages* already in the non-cooperative case.

Theorem 2. *The following is true for* $\rho \in \{rhsExchange, rhsPools', rndRhs\}$:

$$\{2^m \mid m \in \mathbb{N}\} \in NOP_n(\rho, ncoo) \setminus NOP_n(ncoo).$$

Proof. The statement follows (for $n \geq 1$) from the constructions given in Examples 6, 7, and 8 and from the well-known fact that non-cooperative P systems operating under the total halting condition cannot generate non-semilinear number languages (for example, see [14]). □

This result is somewhat surprising at a first glance, but becomes less so when one remarks that the constructions from all three examples only effectively use *one rule* to do the multiplication, which is non-deterministically changed to a “halting” rule. Since there is only one rule acting at any time, randomized right-hand sides allow for clearly delimiting different *derivation phases*.

It turns out that this approach of synchronization by randomization can be exploited to generate even more complex non-semilinear languages.

Theorem 3. *Given a fixed subset of natural factors* $\{f_1, \dots, f_k\} \subseteq \mathbb{N}$*, the following is true for any* $\rho \in \{rhsExchange, rhsPools', rndRhs\}$*:*

$$L = \{f_1^{n_1} \cdot \dots \cdot f_k^{n_k} \mid n_1, \dots, n_k \in \mathbb{N}\} \in NOP_1(\rho, ncoo).$$

Proof. First consider the P system with randomized pools of RHS $\Pi_5 = (\{a, b\}, \{b\}, [_1\ ]_1, a, H, 1)$ with $H = (l, r)$, $l = (a)$ and $r = \left(a^{f_1}\right) \dots \left(a^{f_k}\right) \left(b\right)$. This P system is graphically represented in Figure 5.

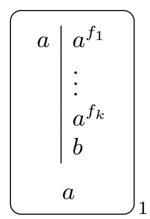

Fig. 5. The P system Π_5 with randomized pools of RHS generating the number language $\{f_1^{n_1} \cdot \dots \cdot f_k^{n_k} \mid n_1, \dots, n_k \in \mathbb{N}\}$.

Similarly to the P systems from Examples 6, 7, and 8, Π_5 halts by choosing to pick the right-hand side b and constructing the rule $a \to b$. If Π_5 picks a different right-hand side, it will multiply the contents of the skin membrane (membrane 1) by one of the factors f_i, $1 \le i \le k$. This proves that $L \in NOP_1(rhsPools', ncoo)$, and, according to Theorem 1, $L \in NOP_1(rhsExchange, ncoo)$ as well: take the P system with the rules $\{a \to a^{f_1}, z_2 \to a^{f_2}, \dots, z_k \to a^{f_k}, z_{k+1} \to b\}$ (the rules with z_j in their left-hand sides are dummy rules).

To show that $L \in NOP_1(rndRhs, ncoo)$, just construct a P system with the only production $a \to \{a^{f_1}, \dots, a^{f_k}, b\}$. □

Therefore, randomizing the right-hand sides of rules in non-cooperative P systems allows for generating non-semilinear languages which cannot be generated without randomization. A natural question to ask is whether randomizing the RHS leads to a *strict increase* in the computational power. The answer is trivially positive for P systems with individual randomized RHS (Variant 3).

Proposition 4. *For any* $n \in \mathbb{N}^+$*,* $PsOP_n(rndRhs, ncoo) \supsetneq PsOP_n(ncoo)$*.*

Proof. The inclusion follows from Proposition 2, as any conventional P system can be trivially seen as a P system with individual randomized RHS in which every production has exactly one right-hand side. Theorem 3 proves the strictness of the inclusion. □

On the other hand, the other two variants of randomizing right-hand sides—random RHS exchange (Variant 1) and randomized pools of RHS (Variant 2)—actually *prevent* one-membrane P systems with non-cooperative rules from generating some semilinear languages, which result also shows that flattening is not possible for these two variants.

In what follows, we will use the expression "only one rule is applied" to refer to the fact that only one given rule $u \to v$ is applied in a certain configuration, possibly in multiple copies. Dually, by saying "at least two rules are applied", we mean that at least two different rules, $u \to v$ and $u' \to v'$, are applied, possibly in multiple copies each.

Theorem 4. *For $\rho \in \{rhsExchange, rhsPools'\}$, the following holds:*

$$L_{ab} = \{a^n \mid n \in \mathbb{N}\} \cup \{b^n \mid n \in \mathbb{N}\} \notin PsOP_1(\rho, ncoo).$$

Proof. Consider a P system Π with randomized RHS of the variant given by ρ and with non-cooperative rules. We immediately remark that no left-hand side in Π may be a or b, because in this case Π will never be able to halt with its only (skin) membrane containing either the multiset a^n or b^n. Furthermore, any RHS of Π contains combinations of symbols a, b, or LHS symbols. Indeed, if an RHS contained a symbol not belonging to these three classes, instances of this symbol would pollute the halting configuration. Finally, Π contains no RHS v such that $a \in v$ and $b \in v$. If Π did contain such an RHS, then any computation could be hijacked to produce a mixture of symbols a and b.

With these remarks in mind, the statement of the theorem follows from the contradicting Lemmas 1 and 2, which are shown immediately after this proof. □

Lemma 1. *Take a $\Pi \in OP_1(\rho, ncoo)$, $\rho \in \{rhsExchange, rhsPools'\}$, such that it generates the number language $Ps(\Pi) = L_{ab}$. Then it must have a computation in which more than one rule is applied (two different left-hand sides are employed) in at least one step.*

Proof. Suppose that Π applies exactly one rule in every step of every computation. We make the following two remarks:

1. Since the words in L_{ab} are of unbounded length, Π must have an LHS t and an RHS v such that $t \in v$, otherwise all computations of Π would have one step and would only produce words of bounded length.
2. Every such RHS v must contain at most one kind of LHS, i.e., if t_1 and t_2 are two LHS of Π then $t_1 \in v$ and $t_2 \in v$ implies $t_1 = t_2$. If this were not the case, after using v, Π would *have* to apply two different rules (assuming that Π has at least as many RHS as LHS).

According to these observations, as well as to those from the proof of Theorem 4, any RHS v of Π is the of the form $v = \alpha\beta$, where $\alpha \in \{a^k, b^k \mid k \in \mathbb{N}\}$, $\beta \in \{t^k \mid k \in \mathbb{N}\}$, and t is an LHS of Π. Note that both α and β may be empty. According to observation (1), Π must have at least an RHS for which $\beta \neq \lambda$ and there exists such an RHS which must be applied an unbounded number of times.

In what follows, we will separately treat the cases in which Π contains or does not contain mixed RHS, i.e., RHS for which both $\alpha \neq \lambda$ and $\beta \neq \lambda$.

No mixed RHS:

Suppose that any RHS of Π which contains a left-hand side is of the form t_2^k. Then, according to our previous observations on the possible forms of the RHS of Π, all RHS containing a are of the form a^i and all RHS containing b are of the form b^j. According to the remarks from the proof of Theorem 4, a and b must not be LHS of Π. Therefore, in any computation of Π, all of a's and b's are produced in the last step. But then, the number of terminal symbols Π produces in a computation can be calculated as a product of the sizes of the RHS of the rules it has applied, which implies that there exists such a $p \in \mathbb{N}$ such that $a^p \notin Ps(\Pi)$ and therefore $Ps(\Pi) \neq L_{ab}$. (p may be picked to be the smallest prime number greater than the length of the longest RHS of Π.)

Mixed RHS:

It follows from the previous paragraph that, in order to generate the number language L_{ab}, Π should contain and apply at least one RHS of the form $a^i t_1^{k_1}$ and at least one RHS of the form $b^j t_2^{k_2}$. Take a computation C of Π producing a and applying the rule $t \to a^i t_1^{k_1}$ at a certain step. Instead of this rule, apply $t \to b^j t_2^{k_2}$, and, in the following step, the rule $t_2 \to a^i t_1^{k_1}$. (We can do so because Π is allowed to pick any permutation of RHS.) Now, Π may continue applying the same rules as in C and eventually halt with a configuration containing *both* a and b. This implies that $Ps(\Pi) \neq L_{ab}$.

It follows from our reasoning that, if Π applies exactly one rule in any step of any computation, it cannot produce L_{ab}, which proves the lemma. □

Lemma 2. *Take a $\Pi \in OP_1(\rho, ncoo)$, $\rho \in \{rhsExchange, rhsPools'\}$, such that it generates the number language $Ps(\Pi) = L_{ab}$. Then, in every computation of Π, exactly one rule is applied (one left-hand side is employed) in every step.*

Proof. Suppose that, in every computation of Π, there exists a step at which at least two different rules are applied. This immediately implies that Π has no RHS of the form a^i or b^j, for $i, j \geq 0$. Indeed, consider a computation producing the multiset a^n and a step in it at which more than one rule is applied. Then Π can replace one of the RHS introduced into the system at this step by b^j and thus end up with a mix of a's and b's in the halting configuration. Therefore, all RHS of Π containing a have the form $a^i v_a$ and all RHS containing b have the form $b^j v_b$, where v_a and v_b are non-empty multisets which only contain LHS symbols (which are neither a nor b).

Now, consider a computation C_a of Π halting on the multiset a^n, and take the *last* step s_a at which at least two different rules are applied. We will consider three different cases, based on whether a and an LHS t appear in the configurations of C_a *after* step s_a.

Both a and t are present:

Suppose both a and an LHS t are present at step $s_a + 1$ in computation C_a. Then t is the only LHS present, because, by our hypothesis, only one rule is applied (maybe in multiple instances) at step $s_a + 1$. In this case, replace the rule applied at step $s_a + 1$ in C_a by $t \to b^j v_b$, where $b^j v_b$ is a right-hand side of Π used in a computation C_b producing b's. From step $s_a + 2$ on in the modified computation, just apply the same rules as applied to the symbols of v_b (and to those derived from v_b) in C_b. The modified computation will reach a halting configuration containing a mix of a's and b's.

Only a is present:

Suppose only a is present at step $s_a + 1$ in computation C_a. Then all of the RHS used at step s_a are λ, because Π has no RHS of the form a^i. Then, replace one of these empty RHS by $b^j v_b$, where $b^j v_b$ is a right-hand side of Π used in a computation C_b producing b's. As before, just apply the same rules as in C_b in the modified computation to get a mix of a's and b's in the halting configuration.

No symbols a are present:

Suppose now that there are no instances of a present at step $s_a + 1$ in computation C_a. Recall that Π has no RHS of the form a^i. Since we suppose that s_a is the last step at which at least two different rules are applied, this means that, in order to produce any a's in C_a, Π must have and use an RHS of the form $a^i t^k$. This RHS contains (multiple copies of) exactly one kind of LHS symbol: t.

Consider a computation C_b halting on the multiset b^n. We pick n sufficiently big to ensure that C_b uses at least two RHS containing b: $b^j v_b$ and $b^{j'} v'_b$ (possibly the same). Without losing generality, we may suppose that these two RHS are either used at the same step in C_b or that $b^{j'} v'_b$ is used at a later step than $b^j v_b$. Then, replace $b^{j'} v'_b$ by $a^i t^k$, pick one of the LHS symbols $t' \in v'_b$ and apply the same rules to t (and to the symbols derived from t) in the modified derivation as were applied to t' (and to the symbols derived from t') in C_b. The modified derivation will therefore contain a mix of a's and b's in the halting configuration.

It follows from our reasoning that, if in any derivation of Π there is a step at which at least two different rules are applied, then $Ps(\Pi) \neq L_{ab}$, which proves the lemma. □

The previous two lemmas are contradicting each other, which means that there exist no one-membrane P systems with random RHS exchange or with random pools of RHS which generate the union language $L_{ab} = \{a^n \mid n \in \mathbb{N}\} \cup \{b^n \mid n \in \mathbb{N}\}$ (this is the statement of Theorem 4). Together with Theorem 3, this leads us to the curious conclusion that one-membrane non-cooperative P systems with random RHS exchange or with randomized pools of RHS are *incomparable* in power to the conventional P systems.

Corollary 2. *For $\rho \in \{rhsExchange, rhsPools'\}$, the following two statements are true:*

$$PsOP_1(\rho, ncoo) \setminus PsOP_1(ncoo) \neq \emptyset, \quad (1)$$

$$PsOP_1(ncoo) \setminus PsOP_1(\rho, ncoo) \neq \emptyset. \quad (2)$$

Proof. Statement (1) follows from Theorem 3. Statement (2) follows from Theorem 4. □

Theorem 4 also allows us to draw a negative conclusion as to the computational completeness of one-membrane non-cooperative P systems with random RHS exchange (Variant 1) and non-cooperative P systems with randomized pools of RHS (Variant 2).

Corollary 3. *For $\rho \in \{rhsExchange, rhsPools'\}$, the following is true:*

$$PsOP_1(\rho, ncoo) \subsetneq PsRE.$$

It turns out that allowing multiple membranes strictly increases the expressive power of Variants 1 and 2 and allows for easily generating *all semilinear* languages, as shown by the following theorem.

Theorem 5. *For $\rho \in \{rhsExchange, rhsPools'\}$, the following holds:*

$$\mathbb{N}^* SLIN_{\mathbb{N}} \in PsOP_*(\rho, ncoo).$$

Proof. Consider the following semilinear language of d-dimensional vectors $L = \bigcup_{1 \leq i \leq n} \langle A_i, \mathbf{b}_i \rangle_{\mathbb{N}}$, where $A_i \subset_{fin} \mathbb{N}^d$ and $\mathbf{b}_i \in \mathbb{N}^d$. We construct the corresponding P system with randomised pools of RHS:

$$\Pi_6 = \Big(O, T, [\ [\ ]_2 \dots [\ ]_{n+1}\,]_1, w_0, \lambda, \dots, \lambda, H_1, \dots H_{n+1}, 1\Big),$$

with the alphabet and the initial contents of the skin defined as follows:

- $O = \{a_1, \dots, a_d, t\}$ contains a symbol per each dimension of the vectors, plus the special symbol t,
- $T = \{a_1, \dots, a_d\}$ contains exactly one symbol per dimension of vectors,
- $w_0 = t$.

The pools of LHS and RHS $H_1 = (l_1, r_1)$ associated with the skin membrane 1 of Π_6 are:

$$l_1 = (t), \quad r_1 = \big(u_1\,(t, in_2)\big) \dots \big(u_n\,(t, in_{n+1})\big),$$

where the multiset u_i corresponds to the offset $\mathbf{b}_i$: $Ps(u_i) = \mathbf{b}_i$, $1 \le i \le n$. Finally, the pools of rule sides $H_{i+1} = (l_{i+1}, r_{i+1})$ associated with inner membrane $i+1$ are defined as follows:

$$l_{i+1} = (t), \quad r_{i+1} = \big(t\,(v_{i1}, out)\big) \dots \big(t\,(v_{ik_i}, out)\big)\,\big(\lambda\big),$$

where the multisets v_{ij}, $1 \le j \le k_i$, correspond to the vectors of the set $A_i = \{\mathbf{a}_{i1}, \dots, \mathbf{a}_{ik_i}\}$: $Ps(v_{ij}) = \mathbf{a}_{ij}$, $1 \le j \le k_i$. By abuse of notation, we write (w, out) to mean that every symbol instance in w gets the target indication *out*. Π_6 is graphically represented in Figure 6.

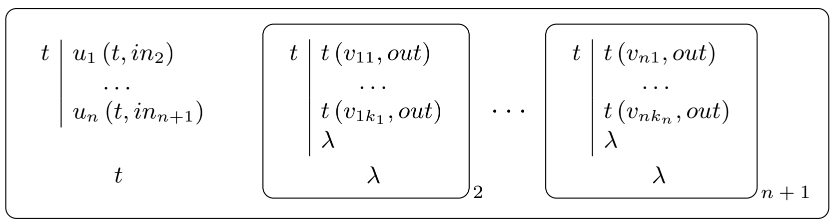

Fig. 6. The P system Π_6 with randomized pools of RHS generating the semilinear language $L = \bigcup_{1 \le i \le n} \langle A_i, \mathbf{b}_i \rangle_{\mathbb{N}}$.

Π_6 starts by non-deterministically building one of the rules $t \to u_i\,(t, in_{i+1})$ in the skin membrane. An application of this rule adds the multiset corresponding to the offset $\mathbf{b}_i$ to the skin membrane and puts t into inner membrane $i+1$. In the following steps only rules in membrane $i+1$ may become applicable. In this membrane, Π_6 may build rules of the form $t \to t\,(v_{ij}, out)$, $1 \le j \le k_i$, which will sustain t while also sending the multiset v_{ij} corresponding to the vector $\mathbf{a}_{ij} \in A_i$ out into the skin. Alternatively, Π_6 may choose to build the rule $t \to \lambda$, an application of which will erase t and halt the system. In such a computation, Π_6 generates the multiset language corresponding to $\langle A_i, \mathbf{b}_i \rangle_{\mathbb{N}}$. Since Π_6 can choose to send t into any one of its inner membranes in the first step and since the computations of said membranes cannot interfere, we conclude that $Ps(\Pi_6) = L$.

To complete the proof, we evoke Theorem 1 to show that there exists a P system with random RHS exchange (Variant 1) generating the same language L.

This theorem allows us to draw a definitive conclusion about the impossibility of flattening for non-cooperative Variants 1 and 2, in contrast to Proposition 1 showing the opposite result for Variant 3.

Corollary 4. *For $\rho \in \{rhsExchange, rhsPools'\}$ and any $k \ge 2$, the following holds:*

$$PsOP_1(\rho, ncoo) \subsetneq PsOP_k(\rho, ncoo).$$

We conclude this section with two more observations regarding the computational power of the Variants 1 and 2. We have seen that, with a single membrane and without cooperation, such P systems cannot generate all semilinear languages; yet it turns out they can generate all *uniform semilinear* languages.

Theorem 6. *For $\rho \in \{rhsExchange, rhsPools'\}$, the following is true:*

$$\mathbb{N}^* SLIN_{\mathbb{N}}^U \subseteq PsOP_1(\rho, ncoo).$$

Proof. Consider two finite sets of d-dimensional vectors $A, B \subset_{fin} \mathbb{N}^d$, $A = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $B = \{\mathbf{y}_1 \ldots, \mathbf{y}_m\}$, and the uniform semilinear set $\langle A, B\rangle_{\mathbb{N}}$. We will now construct the P system $\Pi = (O, T, [\ ]_1, w_0, H, 1)$ with pools of randomized RHS in the following way:

- $O = \{a_1, \ldots, a_d, t\}$ contains a symbol per each dimension of the vectors, plus the special symbol t,
- $T = \{a_1, \ldots, a_d\}$ contains exactly one symbol per dimension of vectors,
- $w_0 = t$,
- $H = (l, r)$, with $l = (t)$ and $r = (w_1' t) \ldots (w_n' t)\,(v_1') \ldots (v_m')$, such that $Ps(w_i') = \mathbf{x}_i$, $1 \leq i \leq n$, and $Ps(v_j') = \mathbf{y}_j$, $1 \leq j \leq m$.

In every step, Π either chooses one of the RHS $(w_i' t)$ which will enable it to reuse the left-hand side symbol t in the following step, or it constructs a rule of the form $t \to v_j'$, which erases the only instance of t and halts the system. Thus, Π performs arbitrary additions of vectors $\mathbf{x}_i \in A$ and then, in the last step of the computation, introduces one of the initial offsets $\mathbf{y}_j \in B$. Therefore, $Ps(\Pi) = \langle A, B\rangle_{\mathbb{N}}$. The fact that we can construct such a P system Π for any uniform semilinear set proves the statement of the theorem. $\square$

Even though one-membranenon-cooperative P systems with random RHS exchange and with randomized pools of RHS cannot generate all unions of linear languages (Theorem 4), they can still generate some limited unions of exponential languages.

Theorem 7. *For $\rho \in \{rhsExchange, rhsPools'\}$, the following is true:*

$$L'_{ab} = \left\{a^{2^n} \mid n \in \mathbb{N}\right\} \cup \left\{b^{2^n} \mid n \in \mathbb{N}\right\} \in PsOP_1(\rho, ncoo).$$

Proof. A P system Π_7 generating the language L'_{ab} can be constructed as follows: $\Pi_7 = (\{a, b, t\}, \{a, b\}, [\ ]_1, t, H, 1)$, where $H = (l, r)$, $l = (t)$ and $r = (tt)(a)(b)$. A graphical representation of Π_7 is given in Figure 7.

Π_7 works by sequentially multiplying the number of symbols t by 2, until it decides to rewrite every instance of t to a or every instance of t to b. Therefore, $Ps(\Pi_7) = L'_{ab}$. According to Proposition 1, there also exists a P system with random RHS exchange generating L'_{ab}, which completes the proof. $\square$

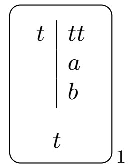

Fig. 7. The P system Π_7 with randomized pools of RHS generating the union language $L'_{ab} = \left\{ a^{2^n} \mid n \in \mathbb{N} \right\} \cup \left\{ b^{2^n} \mid n \in \mathbb{N} \right\}$

The construction from the previous proof can be clearly extended to any number of distinct terminal symbols and to any function of the number of steps $f(n)$ given by a product of exponentials (like in Theorem 3). That is, one can construct a P systems with random RHS exchange or with randomized pools of RHS generating the union language $\left\{ a_i^{f(n)} \;\middle|\; n \in \mathbb{N}, 1 \le i \le m \right\}$, for some fixed number m. Note, however, that we cannot use the same approach to generate unions of two different exponential functions. We conjecture that generating such unions is entirely impossible with Variants 1 and 2 of randomized RHS.

5 Variant 3: A Binary Normal Form

In this section we present a binary normal form for P systems with individual randomized RHS: we prove that, for any such P system, there exists an equivalent one in which every production has at most two right-hand sides.

We now introduce a (rather common) construction: symbols with finite timers attached to them. Given an alphabet O, we define the following two functions:

$$\begin{aligned} timers_o(t, O) &= \bigcup_{i=1}^{t} \{ \langle a, i \rangle \mid a \in O \}, \\ timers_r(t) &= \{ \langle a, i \rangle \to \langle a, i-1 \rangle \mid 2 \le i \le t \} \\ &\cup \{ \langle a, 1 \rangle \to a \mid a \in O \}. \end{aligned}$$

Informally, $timers_o(t, O)$ attaches a t-valued timer to every symbol in O, while $timers_r(t)$ contains the rules making this timer work.

We also define the following function setting a timer to the value $t > 0$ for each symbol in a given string $a_1 \dots a_n$:

$$wait(t, a_1 \dots a_n) = \langle a_1, t \rangle \dots \langle a_n, t \rangle.$$

For $t = 0$, $wait$ is defined to be the identity function: $wait(0, a_1 \dots a_n) = a_1 \dots a_n$.

We can now show that, for any P system with individual randomized RHS there exists an equivalent one having at most two RHS per production.

Theorem 8 (normal form). *For any $\Pi \in OP_n(rndRhs, k)$, $k \in \{coo, ncoo\}$, there exists a $\Pi' \in OP_n(rndRhs^2, k)$ such that $Ps(\Pi') = Ps(\Pi)$.*

Proof. Consider the following P system with individual randomized RHS $\Pi = (O, T, \mu, w_1, \dots, w_n, P_1, \dots P_n, h_o)$ that has at least one production with more than two RHS. We will construct another P system with individual randomized RHS $\Pi' = (O', T, \mu, w_1, \dots, w_n, P'_1, \dots P'_n, h_o)$ such that $Ps(\Pi') = Ps(\Pi)$. The new alphabet will be defined as

$$O' = O \cup timers_o(t, O) \cup \{p_1, \dots, p_t \mid p \in V_p\},$$

where $t + 2$ is the number of right-hand sides in the productions of Π having the most of them, and V_p is an alphabet containing a symbol for each of the individual productions of Π. (If there are two identical productions in Π which belong to two different membranes, V_p will contain one different symbol for each of these two productions.)

For every membrane $1 \leq i \leq n$, the new set of productions P'_i is constructed by applying the following procedure to every production $p \in P_i$:

- If p has the form $u \to \{v\}$, we add the production $u \to \{wait(t, v)\}$ to P'_i.
- If p has the form $u \to \{v_1, v_2\}$, we add $u \to \{wait(t, v_1), wait(t, v_2)\}$ to P'_i.
- If p has the form $u \to \{v_1, \dots, v_k\}$, with $k \geq 3$, we add the following productions to P_i:

$$\begin{aligned} &\big\{u \to \{wait(t, v_1), p_1\}\big\} \\ \cup &\big\{p_j \to \{wait(t - j, v_{j+1}), p_{j+1}\} \mid 1 \leq j < k - 2\big\} \\ \cup &\big\{p_{k-2} \to \{wait(t - k + 2, v_{k-1}), wait(t - k + 2, v_k)\}\big\}. \end{aligned}$$

These productions are graphically represented in Figure 8, in which arrows go from LHS to the associated RHS.

$$\begin{tikzcd} & wait(t, v_1) & & & wait(t - j, v_{j+1}) & & & wait(t - k + 2, v_{k-1}) \\ u \arrow[r] \arrow[ur] & p_1 \arrow[r] & \dots \arrow[r] & p_j \arrow[r] \arrow[ur] & p_{j+1} \arrow[r] & \dots \arrow[r] & p_{k-2} \arrow[r] \arrow[ur] & wait(t - k + 2, v_k) \end{tikzcd}$$

Fig. 8. Timers allow sequential choice between any number of right-hand sides.

Finally we add the rules from $timers_r(t)$, treated as one-RHS production, to every P'_i.

Instead of directly choosing between the right hand-sides of a production $p : u \to \{v_1, \dots, v_k\}$ in one step, Π' chooses between v_1 and delaying the choice to the next step, by producing p_1. This choice between settling on an RHS or continuing the enumeration in the next step may be kept on until $k - 2$ RHS have been discarded. If p_{k-2} is reached, Π' must choose one of the two remaining RHS.

Thus, Π' evolves in "macro-steps", each consisting of exactly t steps. In the first step of a "macro-step", Π' acts on the symbols from O, producing some symbols with timers and delaying some of the choices by producing symbols p_j. All symbols with timers wait exactly until the t-th step of the "macro-step" to turn into the corresponding clean versions from O. Since $t+2$ is the number of RHS in the biggest production of Π, Π' has the time to enumerate all of the RHS of this production.

Since every delayed choice of Π' is uniquely identified by a production-specific symbol p_j, and since only the productions from $timers_r(t)$ can act upon the symbols with timers in Π', the simulations of two different productions of Π cannot interfere. This concludes the proof of the normal form. $\square$

6 Tissue P Systems with Randomized Right-hand Sides of Rules

We now extend the idea of randomized right-hand sides of rules to tissue P systems, where the underlying graph structure is an arbitrary graph structure and not a rooted tree as in the case of hierarchical P systems. Moreover, we also might allow every cell to interact with the environment in case the underlying variant of tissue P system allows/requires that, yet in the following we will assume one of the n cells to figure as the environment, thus being the only cell in which some elementary objects may appear infinitely often

Following the general notation as described for networks of cells in [11], we define a tissue P system as follows:

A *tissue P system* is a construct

$$\Pi = (n, O, T, w_1, \dots, w_n, R, h_i, h_o),$$

where n is the number of cell, labeled by 1 to n, O is the alphabet of objects, $T \subseteq O$ is the alphabet of terminal objects, w_i are the multisets giving the initial contents of each cell i ($1 \le i \le n$), R is the finite set of rules, and h_i and h_o are the labels of the input and the output cells, respectively ($1 \le h_i \le n$, $1 \le h_o \le n$). If e is the label of the environment, then w_e may contain an infinite part. The rules in R are of the form

$$(u_1, \dots, u_n) \to (v_1, \dots, v_n)$$

interpreted as follows: the multisets u_i are replaced by the multisets v_i, $1 \le i \le n$. Such a rule can also be written as follows:

$$\prod_{i=1}^{n}(i, u_i) \to \prod_{i=1}^{n}(i, v_i)$$

Special ingredients can be added to the rules, for example promoters P_i (which have to be present in cell i) and/or inhibitors Q_i (which must not be present in cell i), with P_i and Q_i being finite sets of multisets from O; then a rule

$$((u_1, \dots, u_n) \to (v_1, \dots, v_n); (P_1, \dots, P_n), (Q_1, \dots, Q_n))$$

is applicable to a configuration if and only if cell i contains all elements of P_i and no element from Q_i, $1 \le i \le n$.

Now let m rules be given as

$$\prod_{i=1}^{n}(i,u_i^{(k)}) \to \prod_{i=1}^{n}(i,v_i^{(k)}), \quad 1 \leq k \leq m.$$

According to the general definition of tissue P systems as given above, the rules are not assigned to specific cells but to the whole tissue P system (although assigning rules to cells is another interesting variant to be investigated in the future). For the rules we now have several possibilities to interpret the randomization of the right-hand sides of rules:

Variant A This variant in the strictest way resembles the way randomization was defined for hierarchical P systems:
For a rule $\prod_{i=1}^{n}(i,u_i^{(k)}) \to \prod_{i=1}^{n}(i,v_i^{(k)})$, we simply take $\prod_{i=1}^{n}(i,v_i^{(k)})$ as the right-hand side of the rule and then define Variants 1, 2, and 3 as for hierarchical P systems.

Variant B For the Variants 1 and 2, the right-hand sides $\prod_{i=1}^{n}(i,v_i^{(k)})$ of the m rules are separated into the elements $v_1^{(k)}$ to $v_n^{(k)}$ and the elements $v_i^{(k)}$ for each cell i, $1 \leq i \leq n$, are randomized independently, i.e., we take the multisets

$$M_i = \langle v_i^{(k)} \mid 1 \leq k \leq m \rangle$$

as starting points for randomization and for constructing the rules by taking out one element from M_i for each i, $1 \leq i \leq n$, to construct the right-hand side of a rule.

Variant C As a special variant of Variant B, for randomization in Variants 1 and 2 we only take those $v_i^{(k)}$ for which $v_i^{(k)} \neq \lambda$, i.e., we now instead take the multisets

$$M_i' = \langle v_i^{(k)} \mid v_i^{(k)} \neq \lambda, 1 \leq k \leq m \rangle = \langle x \in M_i \mid x \neq \lambda \rangle.$$

Moreover, we may consider two subvariants how to construct the new right-hand sides of rules:

Variant C.1 If M_i' is empty, then we cannot construct any randomized rule.

Variant C.2 If M_i' is empty, then we take (i,λ) for every constructed randomized rule.

We observe that for Variant 3, i.e., for individual randomized RHS, we only consider Variant A. Therefore, for all three Variants 1 to 3 we will use the notation

$$OtP_n(\alpha, X)$$

to denote the family of tissue P systems with at most n cells using rules of type X with α denoting the type of randomization according to Variants 1 to 3. To denote the family of languages of multisets generated by these P systems, we prepend Ps to the notation, and to denote the family of the generated number languages, we prepend N. For the Variants 1 and 2, we may also add an additional parameter $\beta \in \{B, C.1, C.2\}$ (to indicate how to deal with empty $v_i^{(k)}$) thus obtaining the notations $OtP_n(\alpha, \beta, X)$ etc.

6.1 Equivalence Between Variants 1 and 2 for Variant A

For randomized pools of RHS, again we consider the restriction that there are at least as many right-hand sides as it has left-hand sides for the rules to be constructed, i.e., the type *rhsPools*$'$. Then again we obtain the equivalence between tissue P systems with random RHS exchange and tissue P systems with randomized pools of RHS of type *rhsPools*$'$. The proof follows the same lines as the proof of Theorem 1, now taking into account that we only have to consider the whole system (or, if rules are assigned to cells, we simply replace *membrane* by *cell*).

Proposition 5. *For any* $n \in \mathbb{N}^+$ *and* $X \in \{coo, ncoo\}$*, the following holds:*

$$PsOtP_n(\mathit{rhsExchange}, X) = PsOtP_n(\mathit{rhsPools}', X).$$

7 Conclusions and Open Problems

In this article, we introduced and partially studied P systems with randomized rule right-hand sides. This is a model of P systems with dynamic rules, in which the matching between left-hand and right-hand sides is non-deterministically changed during the evolution. In each step, such P systems first construct the rules from the available rule sides and then apply them, in a maximally parallel way.

We defined three different randomization semantics: random RHS exchange (Variant 1), randomized pools of RHS (Variant 2), and individual randomized RHS (Variant 3). We studied the computational power of the three variants and showed that Variant 3 is quite different in power from Variants 1 and 2. Indeed, P systems with individual randomized RHS (Variant 3) appear as a strict extension of conventional P systems, while random RHS exchange (Variant 1) and randomized pools of RHS (Variant 2) seem to increase the power when only one LHS is used, but to decrease the power when more LHS are present. Finally, we gave a binary normal form for P systems with individual randomized RHS (Variant 3).

7.1 Open Questions

The present work leaves open quite a number of open questions. We list the ones appearing important to us, in no particular order.

Full power of Variants 1 and 2:

Are cooperative, multi-membrane P systems with random RHS exchange (Variant 1) or with randomized pools of RHS (Variant 2) computationally complete? If not, what would be the upper bound on their power? In this article, we showed that applying these two randomization semantics to the non-cooperative, one-membrane case, yields a family of multiset languages incomparable with the family of semi-linear vector sets. How much more can be achieved with cooperativity?

We conjecture that, even with LHS containing more than one symbol, Variants 1 and 2 will *not* be computationally complete. However, we expect that considering systems with multiple membranes may actually bring a substantial boost in computational power, because, in both Variants 1 and 2, randomization happens over each single membrane, meaning that one might use a rich membrane structure to finely control its effects.

Compare the variants:

How do the three variants of RHS randomization compare among one another when applied to non-cooperative rules? We saw that, in all three cases, exponential number languages can be generated. We also saw that individual randomized RHS (Variant 3) produce a strict superset of the semi-linear languages (Proposition 4). Does it imply that Variant 3 is strictly more powerful than Variants 1 and 2? We conjecture a positive answer to this question.

Excess of LHS:

In the case of P systems with randomized pools of RHS (Variant 2), what is the consequence of having *more LHS* available in a membrane than there are RHS? The results in this paper concern a "restricted" version of Variant 2, in which we require that LHS are never in excess. How strong is this restriction? Our conjecture is that allowing an excess of LHS does not increase the computational power.

Applications to vulnerable systems:

As noted in the introduction to the present work, randomized RHS can be seen as a representation of systems mutating in a toxic environment. However, we did not give any concrete examples. It would be interesting to look up any such concrete cases and to evaluate the relevance of this unconventional modeling approach.

7.2 Further Variants

Forbidding identical rules:

In any of the three variants, it may happen that identical rules are constructed, in any membrane. In the previous chapters, in this case this rule was simply taken into the set of rules. Yet we could also forbid such a situation to happen and in such a case completely abandon the whole rule set. Another solution can be to take out all rules having been constructed more than once from the constructed rule set.

The situation of getting identical rules can easily be avoided by avoiding identical RHS: the right-hand sides of rules can be made different by adding suitable powers of a dummy symbol d, which does not count for the final result (i.e., d is no terminal symbol). As d also does not appear on the left-hand side of a rule, the computational power of any of the P systems variant considered in this paper will not be changed by this changing of the set of RHS available for constructing the set of rules.

Identical RHS in Variant 3:

In P systems with individual randomized RHS the computational power mainly arises from the possibility to specify different sets of RHS for the left-hand sides of rules. What happens if the set R of RHS must be the same for all left-hand sides?

References

1. Artiom Alhazov. A note on P systems with activators. In Gheorghe Păun, Agustín Riscos-Núñez, Alvaro Romero-Jiménez, and Fernando Sancho-Caparrini, editors, *Second Brainstorming Week on Membrane Computing, Sevilla, Spain, February 2-7 2004*, pages 16–19, 2004.
2. Artiom Alhazov, Rudolf Freund, Sergiu Ivanov, and Marion Oswald. Observations on P systems with states. In Marian Gheorghe, Ion Petre, Mario J. Pérez-Jiménez, Grzegorz Rozenberg, and Arto Salomaa, editors, *Multidisciplinary Creativity. Hommage to Gheorghe Păun on His 65th Birthday*. Spandugino, 2015.
3. Artiom Alhazov, Sergiu Ivanov, and Yurii Rogozhin. Polymorphic P systems. In Marian Gheorghe, Thomas Hinze, Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa, editors, *Membrane Computing*, volume 6501 of *Lecture Notes in Computer Science*, pages 81–94. Springer, 2011.
4. Fernando Arroyo, Angel V. Baranda, Juan Castellanos, and Gheorghe Păun. Membrane computing: The power of (rule) creation. *Journal of Universal Computer Science*, 8:369–381, 2002.
5. Ilir Çapuni and Péter Gács. A Turing machine resisting isolated bursts of faults. *CoRR*, abs/1203.1335, 2012.
6. Matteo Cavaliere and Daniela Genova. P systems with symport/antiport of rules. In Gheorghe Păun, Agustín Riscos-Núñez, Alvaro Romero-Jiménez, and Fernando Sancho-Caparrini, editors, *Second Brainstorming Week on Membrane Computing, Sevilla, Spain, February 2–7 2004*, pages 102–116, 2004.
7. Matteo Cavaliere, Mihai Ionescu, and Tseren-Onolt Ishdorj. Inhibiting/de-inhibiting rules in P systems. In *Pre-proceedings of the Fifth Workshop on Membrane Computing (WMC5), Milano, Italy, June 2004*, pages 174–183, 2004.
8. Rudolf Freund. Generalized P-Systems. In Gabriel Ciobanu and Gheorghe Păun, editors, *Fundamentals of Computation Theory, 12th International Symposium, FCT '99, Iaşi, Romania, August 30–September 3,1999, Proceedings*, volume 1684 of *Lecture Notes in Computer Science*, pages 281–292. Springer, 1999.
9. Rudolf Freund. P systems working in the sequential mode on arrays and strings. In Cristian Calude, Elena Calude, and Michael J. Dinneen, editors, *Developments in Language Theory, 8th International Conference, DLT 2004, Auckland, New Zealand, December 13-17, 2004, Proceedings*, volume 3340 of *Lecture Notes in Computer Science*, pages 188–199. Springer, 2004.
10. Rudolf Freund, Alberto Leporati, Giancarlo Mauri, Antonio E. Porreca, Sergey Verlan, and Zandron. 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, 2014.

11. Rudolf Freund and Sergey Verlan. 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*, pages 271–284. Springer, 2007.
12. Sergiu Ivanov. Polymorphic P systems with non-cooperative rules and no ingredients. In Marian Gheorghe, Grzegorz Rozenberg, Arto Salomaa, Petr Sosík, and Claudio Zandron, editors, *Membrane Computing - 15th International Conference, CMC 2014, Prague, Czech Republic, August 20–22, 2014, Revised Selected Papers*, volume 8961 of *Lecture Notes in Computer Science*, pages 258–273. Springer, 2014.
13. Gheorghe Păun. Computing with Membranes. *Journal of Computer and System Sciences*, 61:108–143, 1998.
14. Gheorghe Păun, Grzegorz Rozenberg, and Arto Salomaa. *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., New York, NY, USA, 2010.
15. Grzegorz Rozenberg and Arto Salomaa, editors. *Handbook of Formal Languages, 3 volumes*. Springer, New York, NY, USA, 1997.
16. Bulletin of the International Membrane Computing Society (IMCS). `http://membranecomputing.net/IMCSBulletin/index.php`.
17. The P Systems Website. `http://ppage.psystems.eu/`.