Complexity of Simulating R Systems by P Systems

Artiom Alhazov[1], Bogdan Aman[2], Rudolf Freund[3], and Sergiu Ivanov[3]

[1] Institute of Mathematics and Computer Science, Academy of Sciences of Moldova
Str. Academiei 5, Chişinău, MD 2028, Moldova
E-mail: artiom@math.md
[2] Romanian Academy, Institute of Computer Science, Iaşi, Romania
Blvd. Carol I no.8, 700505 Iaşi, Romania
E-mail: bogdan.aman@gmail.com
[3] Faculty of Informatics, TU Wien
Favoritenstraße 9-11, 1040 Vienna, Austria
E-mail: rudi@emcc.at
[4] Université Paris Est, France
E-mail: sergiu.ivanov@u-pec.fr

Summary. We show multiple ways to simulate R systems by non-cooperative P systems with atomic control by promoters and/or inhibitors, or with matter-antimatter annihilation rules, with a slowdown by a factor of constant. The descriptional complexity is also linear with respect to that of simulated R system. All these constants depend on how general the model of R systems is, as well as on the chosen control ingredients of P systems. Special attention is paid to the differences in the mode of rule application in these models.

1 Introduction. Differences between P and R

Membrane systems, also called P systems (non-distributed, with symbol-objects) are a formal model of (possibly controlled) multiset rewriting [8]. Reaction systems, also called R systems, is also a formal rewriting-like model of set evolution introduced in [6], see also a recent survey [5]. Both P systems and R systems are inspired by the functioning of the living cells. It is a natural task to compare R systems, which was introduced later, to P systems, by simulation. The application of a successful solution would be possibilities to use membrane computing tools and perspective for studying reaction systems. Some research comparing them was done in [10], more exactly, this paper considers P systems with no-persistence aspect of R systems, from the viewpoint of the computational power. We, however, first focus on comparing standard R systems to standard P systems by simulating the former with latter, and then revisit the direction of bringing aspects of R systems to the P systems model, verifying how closer this can make the models.

We start the explanation of the simplest case – triples of single objects. Rules in R systems have form (a, b, c), which loosely correspond to $a \to c|_{\neg b}$ in P systems, i.e., the first element is the reactant (in this paper we may also call it the left side) the second element is the product (in this paper we may also call it the right side), and the third element is the inhibitor, with the following differences in the mode of application.

The first difference is that the configuration is a set, not a multiset, and thus simultaneously producing the same symbol by multiple rules yields a single object. P systems with sets of objects instead of multisets of objects have been considered in [1], where they have been shown to be universal in the distributed P systems, both for the transitional model, and for the model with active membranes. However, in [1] the goal of showing universality was reached without actually using this first aspect (automatic reduction of multiple copies of identical object into one object), but rather by avoiding to ever need multiple copies of the same object (in the same region). This aspect, combined with the one below, are called the *threshold principle* in the literature. However, it is also meaningful to view them individually.

The second difference is that, if multiple rules with the same a in the left side exists, (if a is present in the configuration, for all of these rules where the inhibitors are not present in the configuration) **all** these rules are applied, simultaneously producing the corresponding products. (This comes from an inspiration that either the abundance of objects a is sufficient, or the replication and, possibly, proper control take place to guarantee the application of all such rules.) This second aspect is standard, e.g., in *H systems* [11] (together with the first one). The second aspect has been already considered also in P systems area, see, e.g., [3].

The third difference is that the objects are not persistent. This means that, even if an object does not undergo any rule, it still disappears from the configuration of the next step, unless, of course, it is produced by some rule. This third aspect is standard in time-varying distributed H systems [9, 12], (together with the first and second ones), and they relate especially naturally to *TVDH1* systems, see [7].

In the general case, the elements of the triples describing the rules of R systems are **sets** of objects. Hence, the meaning of the triple (A, B, C) is: the joint presence of objects in A, in the case when all objects in B are absent, leads to production of objects in C, and, moreover, the subsequent configuration is precisely equal to the union of the right sides of applicable rules (possibly united with the input context).

2 Preliminaries

The reader is assumed to be familiar with the basic notions of formal languages and membrane computing, see [13] for a comprehensive introduction and the webpage [15] of P systems.

The notation $(ncoo, pro_{k,l} + inh_{k',l'})$ describes the possible class of rules: non-cooperative evolution with at most k promoters of weight at most l and at most k'

inhibitors of weight at most l', see [4, 14]; the sign "+" here means both promoters and inhibitors are allowed to be used in the same rule, if it is not the case, we write a comma instead of a plus sign.

The notation $(ncoo, antim/pri)$ stands for non-cooperative evolution rules and matter-antimatter annihilation rules, with weak priority of *all* annihilation rules assumed over all other rules (the most studied variant of P systems with antimatter), see [2].

3 Using promoters and inhibitors

In fact, in terms of intuition from P systems, A is more similar to a promoter than a reactant (and there is no difference between a set of distinct atomic promoters and a corresponding one higher-weight promoter), and B corresponds to a set of atomic inhibitors (if B were a single higher-weight inhibitor, it would disable the rule when all its elements are present, not just any of them, which would not correspond to the correct definition). However, within the traditional P systems mode, we would additionally need to restrict the rule application to only once per step.

3.1 Using powerful rules

Hence, an arbitrary general R system with alphabet V of k symbols and rules (A_i, B_i, C_i), $1 \leq i \leq n$ could be written as the following P systems (non-cooperative, but with powerful promoters and inhibitors), having additional objects I_1 and d_i for all $1 \leq i \leq n$:

$$\begin{aligned}
&\Pi_0 = (O, \mu = [\ ]_1, w_1, R_1) \text{ where} \\
&O = V \cup \{d_i \mid 1 \leq i \leq n\} \cup I_1, \\
&R_1 = \{d_i \to \prod\nolimits_{c \in C_i} c|_{A_i, \{\neg b \mid b \in B_i\}},\ d_i \to \lambda|_{\neg A_i},\ d_i \to \lambda|_b \mid b \in B_i,\ 1 \leq i \leq n\} \\
&\quad \cup \{a \to \lambda \mid a \in V\} \cup \{I_1 \to I_1 \prod\nolimits_{1 \leq i \leq n} d_i\}.
\end{aligned}$$

This combination of features only takes one step to simulate a step of P systems, $n + k + 1$ symbols and $2n + k + 1 + \sum_{1 \leq i \leq n} |B_i|$ rules. Note that the first rule in the description of R_1 uses a higher-weight promoter *together* with a *set* of atomic inhibitors. Also note that in a special case when the rules of the simulated R system are triples of *single* symbols, the control used becomes atomic promoters *together* with atomic inhibitors.

In the rest of the paper we show how to achieve the same goal with P systems having more restricted rules, also discussing how to produce only *one* copy of symbols present in the simulated R system. We use promoters and inhibitors, then consider only one kind of these features, then we replace them by matter-antimatter annihilation rules, and finally, we discuss how much the problem is simplified if some of the aspects of R systems are assumed by the P systems model.

3.2 Triples of symbols

We start with the simplest case - when the elements of triples describing the rules are single elements. Consider such an R system S with alphabet V and rules $\{(a_i, b_i, c_i) \mid 1 \leq i \leq n\}$. We construct a P system Π_1 simulating S, where the initial configuration w_1 matches the initial configuration of S, and the following rules, the simulation taking only 2 steps:

$$\begin{aligned}
\Pi_1 &= (O, \mu = [\ ]_1, w_1, R_1) \text{ where} \\
O &= V \cup \{a' \mid a \in V\} \cup \{d_i \mid 1 \leq i \leq n\}, \\
R_1 &= \{a \to a' \prod\nolimits_{1 \leq i \leq n, a_i = a} d_i \mid a \in V\} \\
&\quad \cup \{d_i \to c_i|_{\neg (b_i)'},\ d_i \to \lambda|_{(b_i)'} \mid 1 \leq i \leq n\} \cup \{a' \to \lambda \mid a \in V\}.
\end{aligned}$$

The simulation task here is simple for two reasons: we took the simpler model of R systems, and using promoters besides inhibitors makes it possible to remove unneeded objects easily. We also note that the number of objects and rules can be decreased by not producing a' when a participates in the left side of any rule, and using $d_{\min\{j \mid 1 \leq j \leq n,\ a_j = b\}}$ instead of b' as promoter and inhibitor.

If $|V| = k$, then $|O| = 2k + n$ and $|R_1| = 2k + 2n$. Moreover, the optimization described in the previous paragraph decreases both $|O|$ and $|R_1|$ by the number of symbols appearing on the left side of some rule of S.

The multiplicities of symbols may grow. When the same symbol is produced simultaneously by multiple rules, the multiplicative effect happens. It is, however, fairly easy to reset the multiplicities of objects in V to one, at a cost of one more step, $2k + 3$ additional symbols in O and $3k + 3$ additional rules, also using an additional object I_1 in the initial configuration:

$$\begin{aligned}
\Pi_2 &= (O, \mu = [\ ]_1, w_1, R_1 = R_i \cup R_{ii} \cup R_{iii}) \text{ where} \\
O &= V \cup \{a', a'', \overline{a} \mid a \in V\} \cup \{d_i \mid 1 \leq i \leq n\}, \cup \{I_1, I_2, I_3\} \\
R_i &= \{a \to a' \prod\nolimits_{1 \leq i \leq n, a_i = a} d_i \mid a \in V\} \cup \{I_1 \to I_2\}, \\
R_{ii} &= \{d_i \to (c_i)''|_{\neg (b_i)'},\ d_i \to \lambda|_{(b_i)'} \mid 1 \leq i \leq n\} \\
&\quad \cup \{a' \to \lambda \mid a \in V\} \cup \{I_2 \to I_3 \prod\nolimits_{a \in V} \overline{a}\}, \\
R_{iii} &= \{\overline{a} \to a|_{a''},\ \overline{a} \to \lambda|_{\neg a''},\ a'' \to \lambda \mid a \in V\} \cup \{I_3 \to I_1\}.
\end{aligned}$$

3.3 Triples of sets

Now the task is more complicated. While generating a set C_i instead of symbol c_i is straightforward, instead of verifying that a_i is present and b_i is absent, rule applicability is defined as presence of **all** symbols from set A_i and absence of **all** symbols from set B_i. We recall that our task is a constant-time solution. Notice

that the rule is not applicable if and only if some symbol from A_i is absent or some symbol from B_i is present.

Consider such an R system S with alphabet V and rules $\{(A_i, B_i, C_i) \mid 1 \leq i \leq n\}$. We construct a P system Π_3 simulating S, where the initial configuration matches the initial configuration of S, plus an additional object I_1, and the following rules, the simulation taking only 3 steps:

$$\begin{aligned}
\Pi_3 &= (O, \mu = [\ ]_1, w_1, R_1) \text{ where} \\
O &= V \cup \{d_i \mid 1 \leq i \leq n\} \cup \{I_1, I_2, I_3\}, \\
R_1 &= \{I_1 \to d_1 \cdots d_n I_2\} \\
&\quad \cup \{d_i \to \lambda|_{\neg a},\ d_i \to \lambda|_b \mid a \in A_i,\ b \in B_i,\ 1 \leq i \leq n\} \cup \{I_2 \to I_3\} \\
&\quad \cup \{d_i \to \prod\nolimits_{c \in C_i} c|_{I_3} \mid 1 \leq i \leq n\} \cup \{a \to \lambda|_{I_3} \mid a \in V\} \cup \{I_3 \to I_1\}.
\end{aligned}$$

If $|V| = k$, then $|O| = k + n + 3$ and $|R_1| = k + n + 3 + \sum_{1 \leq i \leq k}(|A_i| + |B_i|)$. Notice also that, besides objects from V, no object ever appears in multiple copies. As for each object from V, its multiplicity represents the number of rules in S that has produced it in the last simulated step. Unlike the construction from the previous section, the multiplicative effect does not carry over to the next step of computation of S, since each object from V (except the instances in the starting configuration) is produced from some object d_i, produced in one copy, effectively resetting the multiplicities of the previous step. However, producing objects in V in a single copy requires additional overhead. Similarly to obtaining Π_2 from Π_1, we can obtain Π_4 from Π_3, at the price of one more step, $2k+1$ additional symbols in O and $3k + 1$ additional rules. We skip the details.

3.4 Using only promoters

It should not be any surprise that (in the maximally parallel mode) the effect of inhibitors can be obtained by non-cooperative rules with promoters only. Informally, to verify that some object b is absent, we first check if b is present by some rule $a \to a'|_b$, and it suffices to check in the next step whether a is unchanged. The reverse, i.e. replacing promoters with inhibitors, is even easier to see, since promoting a rule by b can be modeled by inhibiting a rule by some immediately-erased object b', creation of which is inhibited by b. We still think it is interesting to consider the use of only promoters or only inhibitors, for two reasons. First, the reduction of promoters/inhibitors in the *general* case of P systems is too complicated, and second, we would like to explore how little overhead in terms of slowdown and descriptional complexity would suffice to achieve our task.

First, as an exercise, we construct a P system for an R system S with triples of symbols $\{(a_i, b_i, c_i)\}$ as rules. The initial configuration matches the initial configuration of S, plus an additional object I_1.

$$
\begin{aligned}
\Pi_5 &= (O, \mu = [\ ]_1, w_1, R_1) \text{ where} \\
O &= V \cup \{a' \mid a \in V\} \cup \{d_i \mid 1 \le i \le n\} \cup \{I_1, I_2, I_3\}, \\
R_1 &= \{I_1 \to I_2\} \cup \{a \to a' \prod\nolimits_{1 \le i \le n, a_i = a} d_i \mid a \in V\} \\
&\quad \cup \{I_2 \to I_3\} \cup \{d_i \to \lambda|_{(b_i)'} \mid 1 \le i \le n\} \cup \{a' \to \lambda \mid a \in V\} \\
&\quad \cup \{I_3 \to I_1\} \cup \{d_i \to c_i|_{I_3}, \mid 1 \le i \le n\}.
\end{aligned}
$$

This construction is obtained from the first one with promoters and inhibitors, implementing the group of rules with inhibitors (contrasted with existing rules with the same objects as promoters) in the next step, promoted by "timer" I_3. We also note, similarly to Π_1, that the number of objects and rules can be decreased by not producing a' when a participates in the left side of any rule, and using $d_{\min\{j \mid 1 \le j \le n,\ a_j = b\}}$ instead of b' as promoter. Once again, this simulation has multiplicative effect, and the multiplicities can be reset to one, at the price of one more step, $2k+1$ additional symbols in O and $3k+1$ additional rules. Let us call the obtained system Π_6. We omit the details, only mentioning that instead of rules $\overline{a} \to \lambda|_{\neg a''}$ as in Π_2, we can erase these symbols in the next step by rules $\overline{a} \to \lambda|_{I_1}$.

Now consider the general case of simulating an R system S with alphabet V and rules $\{(A_i, B_i, C_i) \mid 1 \le i \le n\}$. The simulating P system below has the initial configuration which matches the initial configuration of S, plus additional objects I_1 and a' for each $a \in V$.

$$
\begin{aligned}
\Pi_7 &= (O, \mu = [\ ]_1, w_1, R_1 = R_i \cup R_{ii} \cup R_{iii}) \text{ where} \\
O &= V \cup \{a' \mid a \in V\} \cup \{d_i \mid 1 \le i \le n\} \cup \{I_1, I_2, I_3\}, \\
R_i &= \{I_1 \to d_1 \cdots d_n I_2\} \cup \{a' \to \lambda|_a \mid a \in V\}, \\
R_{ii} &= \{d_i \to \lambda|_{a'},\ d_i \to \lambda|_b \mid a \in A_i,\ b \in B_i,\ 1 \le i \le n\} \\
&\quad \cup \{a \to \lambda|_{I_2},\ a' \to \lambda|_{I_2} \mid a \in V\} \cup \{I_2 \to I_3\}, \\
R_{iii} &= \{d_i \to \prod\nolimits_{c \in C_i} c|_{I_3} \mid 1 \le i \le n\} \cup \{I_3 \to I_1 \prod\nolimits_{a \in V} a'\}.
\end{aligned}
$$

This construction is obtained from the second one with promoters and inhibitors, as follows. The role of objects a' is to survive for one step if and only if the corresponding object a is present, to be used as a promoter instead of inhibitor a; objects a' are recreated in the last step, for the next simulation cycle. Moreover, as now objects from V are no longer used as inhibitors, they can be removed one step earlier.

The system above needs only 3 steps to simulate a step of S, and if $|V| = k$, then $|O| = 2k + n + 3$ and $|R_1| = 3 + 3k + n + \sum_{1 \le i \le n}(|A_i| + |B_i|)$. Of course, alternatively, objects a' could be created from one additional initial object, at a price of an additional step and a few extra rules, but we currently focus on constructions that are efficient in time and descriptional complexity. We again comment that, although this construction has no multiplicative effect, the number

of copies of a symbol in V produced in the end of the simulation equals the number of rules in S that have produced this symbol in the last step. Producing exactly one copy needs one more step, $2k+1$ additional symbols in O and $3k+1$ additional rules. We call this system Π_8 and give no more details, since obtaining it from Π_7 is exactly like obtaining Π_6 from Π_5.

3.5 Using only inhibitors

First, as an exercise, we construct a P system for an R system S with triples of symbols $\{(a_i, b_i, c_i)\}$ as rules. The initial configuration matches the initial configuration of S, plus an additional object I_1.

$$
\begin{aligned}
\Pi_9 &= (O, \mu = [\ ]_1, w_1, R_1) \text{ where} \\
O &= V \cup \{a' \mid a \in V\} \cup \{d_i \mid 1 \le i \le n\} \cup \{I_1, I_2\}, \\
R_1 &= \{I_1 \to I_2\} \cup \{a \to a' \prod\nolimits_{1 \le i \le n, a_i = a} d_i \mid a \in V\} \\
&\quad \cup \{I_2 \to I_1\} \cup \{d_i \to c_i|_{\neg (b_i)'} \mid 1 \le i \le n\} \\
&\quad \cup \{d_i \to \lambda|_{\neg I_2} \mid 1 \le i \le n\} \cup \{a' \to \lambda|_{\neg I_2} \mid a \in V\}.
\end{aligned}
$$

This construction is obtained from the one with promoters and inhibitors, implementing the group of rules with promoters (contrasted with existing rules with the same objects as inhibitors) in the next step, inhibited by "timer" I_2. Moreover, removing objects a' is delayed for one step, to make sure that the rules inhibited by them in the second step are not applied in the third step. Notice also that the simulation of a computation step of S only takes two steps of computation in Π; the third step of computation cleaning objects d_i and a' overlaps with the first step of simulation of the next step in S. However, this produces no interference, since sub-alphabets $\{d_i \mid 1 \le i \le n\} \cup \{a' \mid a \in V\}$ and $\{I_1\} \cup V$ are disjoint. We also note, similarly to Π_1, that the number of objects and rules can be decreased by not producing a' when a participates in the left side of any rule, and using $d_{\min\{j \mid 1 \le j \le n,\ a_j = b\}}$ instead of b' as promoter.

The problem of multiplicative effect can be solved in the usual way, resetting multiplicities to one: produce one copy of each candidate-object, and erase the objects where the multiplicity is zero. However, with inhibitors it takes longer: one additional step to erase objects $\overline{a}$ when the corresponding object a'' is absent, and one further step to rewrite $\overline{a}$ into a.

$$
\begin{aligned}
\Pi_{10} &= (O, \mu = [\ ]_1, w_1, R_1) \text{ where} \\
O &= V \cup \{a', a'', \overline{a} \mid a \in V\} \cup \{d_i \mid 1 \le i \le n\} \cup \{I_1, I_2, I_3, I_4\}, \\
R_1 &= \{I_1 \to I_2\} \cup \{a \to a' \prod\nolimits_{1 \le i \le n, a_i = a} d_i \mid a \in V\} \\
&\quad \cup \{I_2 \to I_3 \prod\nolimits_{a \in V} \overline{a}\} \cup \{d_i \to (c_i)''|_{\neg (b_i)'} \mid 1 \le i \le n\}
\end{aligned}
$$

$$\cup \{I_3 \to I_4\} \cup \{d_i \to \lambda|_{\neg I_2} \mid 1 \leq i \leq n\} \cup \{a' \to \lambda|_{\neg I_2},\ \overline{a} \to \lambda|_{\neg a''} \mid a \in V\}$$
$$\cup \{I_4 \to I_1\} \cup \{\overline{a} \to a|_{\neg I_3},\ a'' \to \lambda|_{\neg I_3} \mid a \in V\}.$$

Hence, the total additional price for resetting the multiplicities of elements of V to one using only inhibitors is 2 more steps, $2k+2$ additional objects, and $3k+2$ rules.

Now consider the general case of simulating an R system S with alphabet V and rules $\{(A_i, B_i, C_i) \mid 1 \leq i \leq n\}$. The simulating P system below has the initial configuration which matches the initial configuration of S, plus additional objects I_1, J and b' for each $b \in V$.

$$\begin{aligned}
\Pi_{11} &= (O, \mu = [\ ]_1, w_1, R_1 = R_i \cup R_{ii} \cup R_{iii}) \text{ where} \\
O &= V \cup \{b', b'' \mid b \in V\} \cup \{d_i \mid 1 \leq i \leq n\} \cup \{I_1, I_2, I_3, J\}, \\
R_i &= \{I_1 \to d_1 \cdots d_n I_2 J\} \cup \{b' \to b''|_{\neg b} \mid b \in V\} \cup \{J \to \lambda\}, \\
R_{ii} &= \{d_i \to \lambda|_{\neg a},\ d_i \to \lambda|_{\neg b''} \mid a \in A_i,\ b \in B_i,\ 1 \leq i \leq n\} \\
&\quad \cup \{b' \to \lambda|_{\neg I_1} \mid b \in V\} \cup \{I_2 \to I_3,\ J \to \lambda\}, \\
R_{iii} &= \{d_i \to \prod\nolimits_{c \in C_i} c|_{\neg I_2} \mid 1 \leq i \leq n\} \\
&\quad \cup \{a \to \lambda|_{\neg J} \mid a \in V\} \cup \{b'' \to \lambda|_{\neg I_2} \mid b \in V\} \cup \{I_3 \to I_1 J \prod\nolimits_{b \in V} b'\}.
\end{aligned}$$

This construction is obtained from the second one with promoters and inhibitors, as follows. The role of objects b' is to change into b'' if and only if the corresponding object b is present, so b'' can be used as an inhibitor instead of promoter b; objects b' are recreated in the last step, for the next simulation cycle. Moreover, to make sure the rules erasing d_i in the absence of a are not applied in the third step, objects a can only be removed in the third step. This is why an additional object J is present in each of the first two steps of the simulation, inhibiting premature removal of objects a. The rule erasing J is written both in R_i and R_{ii} only to highlight that it is applied both in the first and in the second step.

The system above needs only 3 steps to simulate a step of S, and if $|V| = k$, then $|O| = 3k + n + 4$ and $|R_1| = 4 + 4k + n + \sum_{1 \leq i \leq n}(|A_i| + |B_i|)$. Of course, alternatively, objects b' could be created from one additional initial object, at a price of an additional step and a few extra rules, but we currently focus on constructions that are efficient in time and descriptional complexity. Resetting to one the multiplicities of objects in V can be done exactly how Π_{10} was constructed from Π_9. Hence, the new system Π_{12} will have, compared to Π_{11}, 2 more steps, $2k + 2$ additional objects, and $3k + 2$ rules.

4 Using antimatter

This section is devoted to a different control mechanism: matter-antimatter annihilation rules are used instead of promoters and/or inhibitors. The weak priority of annihilation rules over non-cooperative rules is assumed, which is the most common variant of the antimatter model. First, we notice that erasing with a promoter, say, $d \to \lambda|_b$, in the case the promoting object b is erased without being used anywhere else, and when the number of copies of d is bounded, can be modeled by antimatter as follows:

- replace the promoting object b by anti-object d^- of the promoted object, in sufficient copies to erase all possible copies of promoted object d,
- add erasing rules for this anti-object d^- to remove the copies of the anti-objects which did not annihilate.

We now construct the P system equivalent to Π_1 using antimatter.

$$\begin{aligned}\Pi_{13} &= (O = V \cup \{d_i, d_i^- \mid 1 \le i \le n\}, \mu = [\ ]_1, w_1, R_1) \text{ where}\\ R_1 &= \{a \to \prod\nolimits_{1\le i\le n, b_i=a} d_i^- \prod\nolimits_{1\le i\le n, a_i=a} d_i \mid a \in V\}\\ &\quad \cup \{d_i d_i^- \to \lambda,\ d_i \to c_i,\ d_i^- \to \lambda \mid 1 \le i \le n\}.\end{aligned}$$

In each rule of the first group of R_1, it is enough to produce a single copy of d_i^-, because at most one d_i may be generated by the system in the same step, since the rule uniquely determines its left side. The simulation only takes two steps, and uses $k + 2n$ objects and $k + 3n$ rules.

This construction, too, has multiplicative effect. Resetting multiplicities to one can be done by two-step annihilation. Say, we got some number (possibly zero) of objects c'', and we only want to know whether this number is positive. Then we produce one copy of $(c'')^-$ and rewrite it to c' if it does not immediately annihilate. One step later, we produce one copy of $(c')^-$, and rewrite it to c if it does not immediately annihilate. As a result, c will appear if and only if $(c')^-$ did not annihilate, i.e., c' did not appear one step before. But this happened if and only if $(c'')^-$ was annihilated, i.e., there was at least one copy of c'' two steps before. Performing this routine to objects in V of Π_{13}, we obtain the following system, using an additional starting object I_1:

$$\begin{aligned}\Pi_{14} &= (O, \mu = [\ ]_1, w_1, R_1) \text{ where}\\ O &= V \cup \{a', a'', (a')^-, (a'')^- \mid a \in V\} \cup \{d_i, d_i^- \mid 1 \le i \le n\} \cup \{I_1, I_2, I_3, I_4\},\\ R_1 &= \{a \to \prod\nolimits_{1\le i\le n, b_i=a} d_i^- \prod\nolimits_{1\le i\le n, a_i=a} d_i \mid a \in V\} \cup \{I_1 \to I_2\}\\ &\quad \cup \{d_i d_i^- \to \lambda,\ d_i \to (c_i)'',\ d_i^- \to \lambda \mid 1 \le i \le n\} \cup \{I_2 \to I_3 \prod\nolimits_{a\in V} (a'')^-\}\\ &\quad \cup \{a''(a'')^- \to \lambda,\ (a'')^- \to a' \mid a \in V\} \cup \{I_3 \to I_4\}\\ &\quad \cup \{a'(a')^- \to \lambda,\ (a')^- \to a \mid a \in V\} \cup \{I_4 \to I_1\}.\end{aligned}$$

As you can see, resetting multiplicities with antimatter has a price of two more steps, $4k+4$ additional objects and $4k+4$ additional rules.

Now consider the general case of simulating an R system S with alphabet V and rules $\{(A_i, B_i, C_i) \mid 1 \leq i \leq n\}$. The simulating P system below has the initial configuration which matches the initial configuration of S, plus additional object I_1.

$$
\begin{aligned}
\Pi_{15} &= (O, \mu = [\ ]_1, w_1, R_1 = R_i \cup R_{ii} \cup R_{iii} \text{ where} \\
O &= V \cup \{a', (a')^-, a'', \mid a \in V\} \cup \{d_i, d_i^- \mid 1 \leq i \leq n\} \cup \{I_1, I_2, I_3\}, \\
R_i &= \{I_1 \to I_2 d_1 \cdots d_n \prod\nolimits_{a \in V} a'\} \cup \{a \to (a')^- a'' \mid a \in V\}, \\
R_{ii} &= \{a'(a')^- \to \lambda,\ a' \to d_i^-,\ b'' \to d_i^-,\ (a')^- \to \lambda \\
&\qquad \mid a \in A_i,\ b \in B_i,\ 1 \leq i \leq n\} \cup \{I_2 \to I_3\}, \\
R_{iii} &= \{d_i d_i^- \to \lambda,\ d_i \to \prod\nolimits_{c \in C_i} c,\ d_i^- \to \lambda \mid 1 \leq i \leq n\} \cup \{I_3 \to I_1\}.
\end{aligned}
$$

Symbols from C_i are produced from d_i if and only if it is not annihilated, i.e., neither a' nor b'' should produce d_i^- for any $a \in A_i$, $b \in B_i$. Since a' is annihilated if and only if a is present, and b'' is not produced if and only if b is absent, the simulation of an application of rule i of the R system happens if and only if all symbols from the first set are present and all symbols from the second set are absent. The simulation takes 3 steps, using the alphabet of $4k + 2n + 3$ symbols and the set of $3k + 3n + 3 + \sum_{1 \leq i \leq n}(|A_i| + |B_i|)$ rules.

This construction produces each symbol in multiplicity equal to the number of rules of S that produced it, not carrying the multiplicative effect to the next step. If needed, resetting multiplicities can be done costing two more steps, $4k+2$ additional objects and $4k+2$ additional rules. We call this system Π_{16}, and provide no more details since it is obtained from Π_{15} exactly as Π_{14} is obtained from Π_{13}.

5 Non-standard P systems

Some difficulty of simulation of R systems by P systems lie in the difference of their standard derivation modes. We would like to discuss how varying this may affect the problem.

First, if we consider P systems **with sets** instead of multisets, where production of a symbol multiple times still yields a single copy of the result, then *all* constructions in this paper still hold literally, i.e., no changes in the description of these P systems is needed. However, some things may become simpler, e.g., in this case resetting multiplicities to one is done by the model, and does not require additional time, symbol and rule complexity.

We note that, in a non-distributed case, P systems with sets of objects are no longer universal, since the number of possible configuration is bounded by two to

the power of the cardinality of the alphabet. However, universality is not needed to simulate R systems (which has also been shown in the case of deterministic P systems with promoters and/or inhibitors).

Second, if we consider P systems which deterministically apply *all* individually applicable rules, even with overlapping left sides (i.e., competing for resources), then of course the existing solutions still literally hold, but in some cases there are much easier ways: we would have no need to explicitly produce multiple objects from one. For instance, the constructions in this paper usually involve production of rule labels d_i, either from the corresponding reactant a_i, or from some "timer" object I_j, and then have different rules processing these label objects. In this "auto-replication" mode, these various processing rules could be applied directly to the corresponding original object a_i or I_j, the replication being done by the model itself. This would definitely simplify the simulation. Let us refer to this aspect as **auto-replication**. For example, the set of rules of system Π_1 can be simplified to the following:

$$\{a_i \to c_i|_{\neg b_i} \mid 1 \le i \le n\} \cup \{a \to \lambda \mid a \in V\},$$

i.e., just one step, no additional objects and k additional rules. The problem with resetting the multiplicities is also simpler:

$$\begin{aligned}
&\Pi = (O, \mu = [\ ]_1, w_1, R_1 \text{ where}\\
&O = V \cup \{a' \mid a \in V\} \cup \{I_1, I_2\},\\
&R_1 = \{I_1 \to I_2\} \cup \{a_i \to (c_i)'|_{\neg b_i} \mid 1 \le i \le n\} \cup \{a \to \lambda \mid a \in V\}\\
&\quad \cup \{I_2 \to I_1\} \cup \{I_2 \to c|_{c'}\} \cup \{c' \to \lambda \mid c \in V\},
\end{aligned}$$

i.e., requiring only one more step, $k+2$ additional symbols and $2k+2$ additional rules (compared to increase of complexity of Π_2 over Π_1 by one step, $2k+3$ symbols and by $3k+3$ rules).

Third, if we consider P systems where idle objects (i.e., those not consumed by applied rules) do not contribute to the next configuration, we call this aspect "**no persistence**", then many erasing rules (in particular, all erasing promoted or inhibited by some "timer" I_j) would no longer be needed, while occasionally some renaming rules should be added when object was designed to be used later than in the next step after its production. For instance, in case of no-persistence, all $n+k'$ erasing rules of Π_1 may be removed, leaving just $n+k$ rules. Similarly, all erasing rules of Π_2 may be removed; hence, the subtask of resetting the multiplicities to one in this case only needs $k+3$ additional rules instead of $3k+3$.

However, testing for presence of some object b by "failing to apply a rule with inhibitor b and finding the reactant unchanged in the next step" would not work. The working solution is to use b as an inhibitor in a rule producing some object b', and to use b' as an inhibitor in the next step. Testing for absence by "failing to apply a rule with a promoter and finding the reactant unchanged in the next

step" would be no longer possible, so the model with promoters only seems to be considerably weaker in the case without persistence of idle objects.

We would like to note that, in case of P systems with sets and auto-replication, the aspect of no-persistence can be simulated as follows: add rules $a \to \lambda$ for each $a \in V$; they will make sure that such objects are not carried over to the next step, in the same time not adding anything to the result (as for productive objects, erasing them is just another option, which in the auto-replication case neither grows nor shrinks the set of objects obtained from them). This simulation takes one step, k objects and $n + k$ rules.

And, of course, if we consider P systems with all these differences, i.e., with sets, auto-replication, and without object persistence, then rule (a, b, c) of R systems becomes *identical* to rule $a \to c|_{\neg b}$ of P systems, while rule (A, B, C) of R systems becomes *identical* to rule $\prod_{a \in A} a \to \prod_{c \in C} c|_{\{\neg b | b \in B\}}$, so the simulation is trivial, requiring one step, k objects and n rules of type $(ncoo, pro_{1,*} + inh_{*,1})$.

6 Conclusions

We recall that although deterministic P systems with promoters and/or inhibitors are not universal and have subregular characterizations, their power is sufficient to simulate R systems.

All constructions presented in this paper (except those in previous section) simulate R systems (in their standard derivation mode) by P systems (in *their* standard derivation mode), with the slowdown by a factor of constant, where the descriptional complexity of the simulating P system is linear with respect to the descriptional complexity of the simulating P system. The proportionality constants vary depending on whether R systems are defined as triples of symbols or as triples of sets of symbols, and on whether promoters, inhibitors or both are used in P systems. All constructions are deterministic: while the multiset of rules to be applied to a given configuration may not be unique, the next configuration is unique. Indeed, in all these constructions, if two rules have the same left side, then either their applicability is mutually exclusive (one is being promoted and the other one is being inhibited by the same symbol), or also the right side is the same (and thus, if there are multiple choices which object would promote or inhibit the rule, such choice would not influence the result).

Seventeen constructions are presented, see Table 1: 0)(general and simple in particular) R systems using single higher-weight promoters together with multiple atomic inhibitors, 1)simple R systems using promoters and inhibitors, 2)simple R systems using promoters and inhibitors and resetting multiplicities to one, 3)general R systems using promoters and inhibitors, 4)general R systems using promoters and inhibitors and resetting multiplicities to one, 5)simple R systems using promoters, 6)simple R systems using promoters and resetting multiplicities to one, 7)general R systems using promoters, 8)general R systems using promoters and resetting multiplicities to one, 9)simple R systems using inhibitors, 10)simple

P	R	mult	features	steps	$\|O\|$	$\|R_1\|$
Π_0	s	L	$(ncoo, pro_{1,1} + inh_{1,1})$	1	$n+k+1$	$3n+k+1$
Π_0	G	L	$(ncoo, pro_{1,*} + inh_{*,1})$	1	$n+k+1$	$2n+k+1+T'$
Π_1	s	M	$(ncoo, pro_{1,1}, inh_{1,1})$	2	$n+k+k'$	$2n+k+k'$
Π_2	s	1	$(ncoo, pro_{1,1}, inh_{1,1})$	3	$n+3k+k'+3$	$2n+4k+k'+3$
Π_3	G	L	$(ncoo, pro_{1,1}, inh_{1,1})$	3	$n+k+3$	$n+k+3+T$
Π_4	G	1	$(ncoo, pro_{1,1}, inh_{1,1})$	4	$n+3k+4$	$n+4k+4+T$
Π_5	s	M	$(ncoo, pro_{1,1})$	3	$n+k+k'+3$	$2n+k+k'+3$
Π_6	s	1	$(ncoo, pro_{1,1})$	4	$n+3k+k'+4$	$2n+4k+k'+4$
Π_7	G	L	$(ncoo, pro_{1,1})$	3	$n+2k+3$	$n+3k+3+T$
Π_8	G	1	$(ncoo, pro_{1,1})$	4	$n+4k+4$	$n+6k+4+T$
Π_9	s	M	$(ncoo, inh_{1,1})$	2	$n+k+k'+2$	$2n+k+k'+2$
Π_{10}	s	1	$(ncoo, inh_{1,1})$	4	$n+3k+k'+4$	$2n+4k+k'+4$
Π_{11}	G	L	$(ncoo, inh_{1,1})$	3	$n+3k+4$	$n+4k+4+T$
Π_{12}	G	1	$(ncoo, inh_{1,1})$	5	$n+5k+6$	$n+6k+6+T$
Π_{13}	s	M	$(ncoo, antim/pri)$	2	$2n+k$	$3n+k$
Π_{14}	s	1	$(ncoo, antim/pri)$	4	$2n+5k+4$	$3n+5k+4$
Π_{15}	G	L	$(ncoo, antim/pri)$	3	$2n+4k+3$	$3n+3k+3+T$
Π_{16}	G	1	$(ncoo, antim/pri)$	5	$2n+8k+5$	$3n+7k+5+T$

Table 1. Comparative table of simulation of R systems by P systems

R systems using inhibitors and resetting multiplicities to one, 11)general R systems using inhibitors, 12)general R systems using inhibitors and resetting multiplicities to one, 13)simple R systems using antimatter, 14)simple R systems using antimatter and resetting multiplicities to one, 15)general R systems using antimatter, 16)general R systems using antimatter and resetting multiplicities to one. The table below shows the number of steps of simulating P system to simulate one step of R system, alphabet size and the number of rules in these simulations (n is the number of rules in S, k is the number of symbols in S, k' is the number of symbols that do not appear in the left side of any rule of the simulated system; by T we denote $\sum_{1 \leq i \leq k}(|A_i| + |B_i|)$ and by T' we denote $\sum_{1 \leq i \leq k} |B_i|$). Column R describes the type of simulated system, where s stands for simple (rules with triples of symbols) and G stands for general (rules with triples of sets). Column mult describes the multiplicities of symbols in the simulating P system, where M stands for multiplicative effect, L stands for last multiplicity, and 1 stands for multiplicities 0 and 1. Column features describes the kinds of rules used.

Note: in Π_6, Π_8 and Π_9, intermediate objects are removed one step later, in parallel with the first step of simulation of the next step of evolution of the simulated R system, but not interfering with it.

Finally, in the previous section we discussed how (qualitatively and quantitatively) adopting some aspects of R systems (such as sets instead of multisets, auto-replication or no-persistence) into the working model of P systems simplifies simulation of R systems.

References

1. A. Alhazov. P systems without multiplicities of symbol-objects. *Information Processing Letters*, 100(3):124–129, 2006.
2. A. Alhazov, B. Aman, and R. Freund. P systems with anti-matter. In M. Gheorghe, G. Rozenberg, A. Salomaa, P. Sosík, and C. 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 66–85. Springer, 2014.
3. A. Alhazov, S. Cojocaru, A. Colesnicov, L. Malahova, and M. Petic. A P system for annotation of Romanian affixes. In A. Alhazov, S. Cojocaru, M. Gheorghe, Yu. Rogozhin, G. Rozenberg, and A. Salomaa, editors, *Membrane Computing - 14th International Conference, CMC 2013, Chişinău, Republic of Moldova, August 20-23, 2013, Revised Selected Papers*, volume 8340 of *Lecture Notes in Computer Science*, pages 80–87. Springer, 2013.
4. A. Alhazov and R. Freund. Asynchronous and maximally parallel deterministic controlled non-cooperative P systems characterize NFIN and coNFIN. In E. Csuhaj-Varjú, M. Gheorghe, G. Rozenberg, A. Salomaa, and Gy. Vaszil, editors, *Membrane Computing - 13th International Conference, CMC 2012, Budapest, Hungary, August 28-31, 2012, Revised Selected Papers*, volume 7762 of *Lecture Notes in Computer Science*, pages 101–111. Springer, 2012.
5. R. Brijder, A. Ehrenfeucht, M. G. Main, and G. Rozenberg. A tour of reaction systems. *International Journal of Foundations of Computer Science*, 22(7):1499–1517, 2011.
6. A. Ehrenfeucht and G. Rozenberg. Reaction systems. *Fundamenta Informaticae*, 75(1):263–280, 2007.
7. M. Margenstern, Yu. Rogozhin, and S. Verlan. Time-varying distributed H systems with parallel computations: The problem is solved. In J. Chen and J. H. Reif, editors, *DNA Computing, 9th International Workshop on DNA Based Computers, DNA9, Madison, WI, USA, June 1-3, 2003, revised Papers*, volume 2943 of *Lecture Notes in Computer Science*, pages 48–53. Springer, 2003.
8. Gh. Păun. Computing with membranes. *Journal of Computer and System Sciences*, 61:108–143, 1998.
9. Gh. Păun. DNA computing based on splicing: universality results. *Theoretical Computer Science*, 231(2):275–296, 2000.
10. Gh. Păun and M. J. Pérez-Jiménez. Towards bridging two cell-inspired models: P systems and R systems. *Theoretical Computer Science*, 429:258–264, 2012.
11. Gh. Păun, G. Rozenberg, and A. Salomaa. Computing by splicing. *Theoretical Computer Science*, 168(2):321–336, 1996.
12. Gh. Păun, G. Rozenberg, and A. Salomaa. *DNA Computing - New Computing Paradigms*. Texts in Theoretical Computer Science. An EATCS Series. Springer, 1998.
13. Gh. Păun, G. Rozenberg, and A. Salomaa. *The Oxford Handbook of Membrane Computing*. Oxford University Press, Inc., New York, NY, USA, 2010.
14. D. Sburlan. Further results on P systems with promoters/inhibitors. *International Journal of Foundations of Computer Science*, 17(1):205–221, 2006.
15. The P Systems Website. `http://ppage.psystems.eu`.