Computer Science Journal of Moldova, vol.**23**, no.3(69), 2015

Small P Systems with Catalysts or Anti-Matter Simulating Generalized Register Machines and Generalized Counter Automata

Artiom Alhazov Rudolf Freund Petr Sosík

Abstract

In this paper we focus on two weak forms of cooperation in P systems, namely, catalytic rules and matter/anti-matter annihilation rules. These variants of P systems both are computationally complete, while the corresponding rule complexity turns out to be of special interest. For establishing considerably small universal P systems in both cases, we found two suitable tools: generalized register machines and generalized counter automata. Depending on the features used in the different variants, we construct several small universal P systems.

1 Introduction

Membrane systems with symbol objects are a theoretical framework of parallel distributed multiset processing, for example, see [12, 13, 14]. While non-cooperative P systems are known to characterize the regular languages, in case of unrestricted (even binary) cooperation, showing computational completeness is straightforward, for example, by simulating register machines. Hence, since many years researchers have been interested in even weaker forms of cooperation.

A catalytic rule is a non-cooperative rule with an additional catalyst on both the left side and the right side of the rule. Essentially, a catalyst only inhibits parallelism of rules where it is indicated. The question whether catalytic P systems are computationally complete (without priorities or other additional features) has been open for a number of

years, being finally answered positively, moreover, even showing that two catalysts suffice (or three for the purely catalytic systems), see [7].

In the variant with anti-matter objects, in addition to non-cooperative rules, specific cooperative erasing is allowed, namely, of two objects related by a bijection "object-antiobject". Anti-matter in P systems is a rather recent direction, for instance, see [1].

Small universal P systems have been investigated for a number of years. The smallest ones are those with string objects and splicing rules where even five rules suffice, see [5]. In the case of symbol objects, if full cooperation is allowed, then 23 rules suffice, see [6], and only 16 are needed if in addition inhibitors are allowed, see [8].

In this paper, we give an overview on small universal P systems using anti-matter or catalysts as in [4] and we even improve the results established there for (purely) catalytic P systems, based on recent results obtained in [3] as well as in [16] and [17].

2 Definitions

We assume the reader to be familiar with the basic notions and concepts from formal language theory, for example, see textbooks as [15]; for the area of P systems we refer to [12, 13, 14] and to [18] for actual news.

For an alphabet V, by V^* we denote the free monoid generated by V under the operation of concatenation, i.e., containing all possible strings over V. The *empty string* is denoted by λ.

In this paper we will not distinguish between a multiset, its string representation (having as many occurrences of every symbol as its multiplicity in the multiset, the order in the string being irrelevant), and a vector of multiplicities (assuming that the order of enumeration of symbols from V is fixed). We mention that $\prod$ represents the concatenation of an ordered list of strings, and if these strings represent multisets, this corresponds to their union.

2.1 Register Machines

Register machines are well-known universal devices for computing (generating or accepting) sets of (vectors of) natural numbers.

Definition 1 *A* register machine *is a construct* $M = (m, B, l_0, l_h, P)$ *where*

- m *is the number of registers,*
- P *is the set of instructions bijectively labeled by elements of* B*,*
- $l_0 \in B$ *is the initial label, and*
- $l_h \in B$ *is the final label.*

The instructions of M *can be of the following forms:*

- $p : (\mathtt{ADD}(r), q, s)$*, with* $p \in B \setminus \{l_h\}$*,* $q, s \in B$*,* $1 \leq r \leq m$*.*
  Increase the value of register r *by one, and non-deterministically jump to instruction* q *or* s*.*
- $p : (\mathtt{SUB}(r), q, s)$*, with* $p \in B \setminus \{l_h\}$*,* $q, s \in B$*,* $1 \leq r \leq m$*.*
  If the value of register r *is not zero, then decrease the value of register* r *by one (*decrement *case) and jump to instruction* q*, otherwise jump to instruction* s *(*zero-test *case).*
- $l_h : \mathtt{HALT}$*.*
  Stop the execution of the register machine.

A configuration *of a register machine is described by the contents of each register and by the value of the current label, which indicates the next instruction to be executed.* M *is called* deterministic *if all the ADD-instructions are of the form* $p : (\mathtt{ADD}(r), q)$*.*

In the accepting case, a computation starts with the input of a k-vector of natural numbers in its first k registers and by executing the first instruction of P (labeled with l_0); it terminates with reaching the HALT-instruction. Without loss of generality, we may assume all registers to be empty at the end of the computation.

A register machine M_U is called *universal*, if, given the code of an arbitrary register machine M, M_U can simulate the computations of M on any given input. We speak of *strong universality*, if both input and output are given directly as numbers, whereas *weak universality* means that both input and output are encoded by a recursive function f, e.g., $f(n) = 2^n$; we also consider *weak-strong universality* with encoded input, but unencoded output.

2.2 P Systems

In this paper, we will only consider membrane systems with the simplest membrane structure $\mu = [\ ]_1$, i.e., with even omitting μ, we consider a (catalytic) *P system* as a construct $\Pi = (O, C, w_1, R_1)$ where O is the alphabet of *objects*, $C \subseteq O$ is the set of *catalysts*, w_1 the multiset of objects present in the skin region at the beginning of a computation, and R_1 is a finite set of *evolution rules*, associated with the skin region. In this paper we only use the *maximally parallel derivation mode*, i.e., in each derivation step we apply a non-extendable multiset of rules.

If a rule $u \to v$ has at least two objects in u, then it is called *cooperative*, otherwise it is called *non-cooperative*. In *catalytic P systems* we use non-cooperative as well as *catalytic rules*, which are of the form $ca \to cv$ where c is a special object called *catalyst*, which never evolves (this restriction can be relaxed), but it just assists object a to evolve to the multiset v. In a *purely catalytic P system* we only allow catalytic rules. If we allow catalysts to switch between different states, we speak of *multi-stable catalysts*.

In P systems with *anti-matter* objects, each object a also has an anti-matter object $\bar{a}$ in O and, in addition to non-cooperative and catalytic rules, matter/anti-matter annihilation rules $a\bar{a} \to \lambda$ are allowed, for instance, see [1].

In P systems with *toxic* objects, specific symbols are specified as being toxic; a computation can only be continued by a non-extendable multiset of rules which does not leave any toxic object idle. For more details about toxic P systems, for example, see [2].

3 Small Universal Register Machines

The universal register machines with the smallest known number of instructions are those constructed by I. Korec in [10]. For the standard instruction set (ADD-instructions and SUB-instructions, not counting the halting one), these are the strongly universal machine U_{22} and the weakly universal machine U_{20}, see Figure 1.

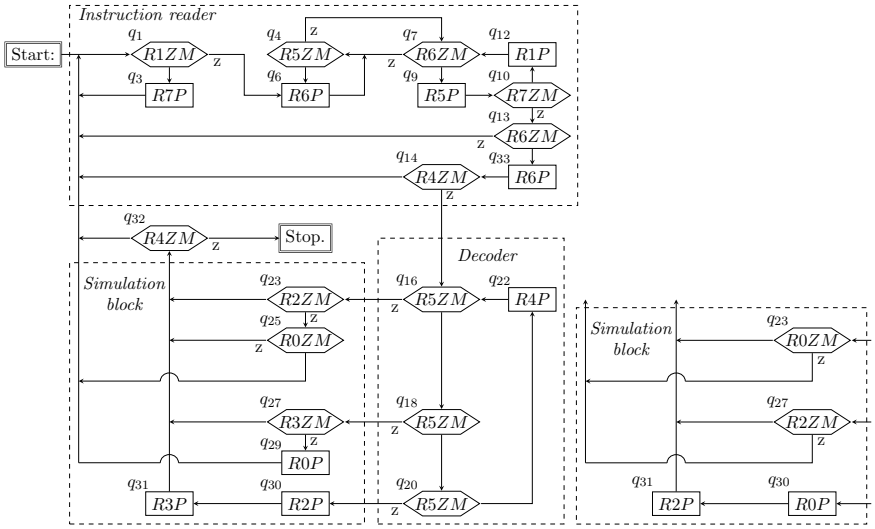

Figure 1. The strongly universal register machine U_{22} (left) and the simulation block of the weakly universal register machine U_{20} (right).

3.1 Generalized Register Machines

We often observe that the most efficient (in terms of rule complexity) simulations of register machines by P systems do not use separate rules for ADD-instructions, but perform them as a part of the rules simulating SUB-instructions. Hence, we recall from [4] the following generalization of register machines, as a tool for such simulations.

The model of *generalized register machines* has only instructions of one type except the halt instruction, i.e., generalized SUB-instructions

of the form $j : (\texttt{SUB}(r), A_-(j)k, A_0(j)l)$ where $j, k, l \in B$ are instruction labels and $A_-(j), A_0(j)$ are (possibly empty) strings of increment commands (sub-instructions) $\texttt{ADD}(j')$. Clearly, a standard register machine (with ADD-instructions and SUB-instructions) can be obtained from a generalized one, simply by introducing intermediate states, see [4] for additional remarks.

3.2 With Multiple Registers

Below we present the (rules for the) strongly universal register machine U_{22} of Korec, see [10] and Figure 1, left, in the form of a generalized register machine:

$q_1 : (\texttt{SUB}(1), \texttt{ADD}(7)q_1, \texttt{ADD}(6)q_4)$, $q_{16} : (\texttt{SUB}(5), q_{18}, q_{23})$,
$q_4 : (\texttt{SUB}(5), \texttt{ADD}(6)q_4, q_7)$, $q_{18} : (\texttt{SUB}(5), q_{20}, q_{27})$,
$q_7 : (\texttt{SUB}(6), \texttt{ADD}(5)q_{10}, q_4)$, $q_{23} : (\texttt{SUB}(2), q_{32}, q_{25})$,
$q_{10} : (\texttt{SUB}(7), \texttt{ADD}(1)q_7, q_{13})$, $q_{25} : (\texttt{SUB}(0), q_1, q_{32})$,
$q_{13} : (\texttt{SUB}(6), \texttt{ADD}(6)q_{14}, q_1)$, $q_{27} : (\texttt{SUB}(3), q_{32}, \texttt{ADD}(0)q_1)$,
$q_{14} : (\texttt{SUB}(4), q_1, q_{16})$, $q_{32} : (\texttt{SUB}(4), q_1, q_h)$,
$q_{20} : (\texttt{SUB}(5), \texttt{ADD}(4)q_{16}, \texttt{ADD}(2)\texttt{ADD}(3)q_{32})$.

In the generalized register machine form of the weakly universal register machine U_{20} of Korec, see [10], q_{25} is no longer present, and instructions q_{20}, q_{23} and q_{27} are different, see Figure 1, right, and register 3 is not needed any more:

$q_{20} : (\texttt{SUB}(5), \texttt{ADD}(4)q_{16})$, $q_{23} : (\texttt{SUB}(0), q_{32}, q_1)$, $q_{27} : (\texttt{SUB}(2), q_{32}, q_1)$.

Remark 1 *Sometimes, also for technical reasons, we want to produce the output in a register which only has increment instructions associated to it, and have all other registers empty in the end. Unfortunately, these technical details are not fulfilled by the (strongly or weakly) universal register machines constructed by Korec in [10]: the result is obtained in register* 0*, a register allowing for SUB-instructions, and, due to the specific features of the register machines simulated by the universal Korec machines, (only) the registers* 1 *and* 6 *are not empty. Therefore, the last instruction* q_{32} *can be replaced by the following ones, with*

register 8 *being the new output register; we can omit the right column and already take* q_{35} *as the halting state if "cleaning" is not needed:*

$$q_{32} : (\texttt{SUB}(4), q_1, q_{34}), \qquad q_{35} : (\texttt{SUB}(1), q_{35}, q_{36}),$$
$$q_{34} : (\texttt{SUB}(0), \texttt{ADD}(8)q_{34}, q_{35}), \qquad q_{36} : (\texttt{SUB}(6), q_{36}, q_h).$$

3.3 With Two Decrementable Registers

In this subsection we discuss how to reduce the number of registers to two, possibly not counting an extra increment-only register. It is well known, e.g., see [11], that the computations of any m-register machine can be simulated by a 2-register machine, via exponential encoding. Indeed, if we take the first m prime numbers p_i, $1 \leq i \leq m$, the values x_i of the registers i, $1 \leq i \leq m$, can be encoded in any of the first two registers as the single number ${p_1}^{x_1} \dots {p_m}^{x_m}$. Then, $\texttt{ADD}(r)$ is simulated by multiplying the value of the first register by p_r, and $\texttt{SUB}(r)$ is simulated by trying to divide the value of the first register by p_r; if the division is successful, the decrement transition is made, and otherwise, the value is restored, and the zero-test transition is made.

In the following, we analyze the simulation blocks mentioned above and represent the obtained 2-register machine in the generalized register machine form, following the constructions given in [11]:
– Instruction $j : (\texttt{ADD}(r), k)$ is simulated by the two generalized SUB-instructions $j : (\texttt{SUB}(1), (\texttt{ADD}(2))^{p_r}\ j, j')$ and $j' : (\texttt{SUB}(2), \texttt{ADD}(1)j', k)$.
– Instruction $j : (\texttt{SUB}(r), k, l)$ is simulated by the $p_r + 2$ generalized SUB-instructions

$$j : (\texttt{SUB}(1), j_1, j'),$$
$$j_n : (\texttt{SUB}(1), j_{n+1}, (\texttt{ADD}(1))^n\ j''), \text{ for } 1 \leq n \leq p_r - 2,$$
$$j_n : (\texttt{SUB}(1), \texttt{ADD}(2)j, (\texttt{ADD}(1))^n\ j''), \text{ for } n = p_r - 1,$$
$$j' : (\texttt{SUB}(2), \texttt{ADD}(1)j', k),$$
$$j'' : (\texttt{SUB}(2), (\texttt{ADD}(1))^n\ j'', l), \text{ for } n = p_r.$$

In the course of the analysis of the number of uses of decrements, the assignment of prime numbers to registers was chosen for U_{22}: The conditional decrement of register 5 happens 4 times, the conditional decrements of registers 4 and 6 happen twice each, and the conditional

decrement of any other register happens once. Register 5 is represented by powers of 2, registers 6 and 4 by powers of 3 and 5 as well as registers 0, 1, 2, 7, and 3 by powers of 7, 11, 13, 17, and 19, respectively. We remark that we use a smaller prime for $R6$ than for $R4$ because the former is incremented twice and the latter is incremented only once in the underlying Korec machine, which, compared to the opposite choice leads to saving two ADD-instructions, which might be an interesting feature in another context, although in the present paper we are not concerned about that. We used the largest of the first 8 primes for $R3$ because $R3$, besides being one of the least-used registers here, is no longer used in the weakly universal Korec machine U_{20} considered next. Moreover, a smaller prime is used for $R0$ than for $R1$ and $R2$, because $R0$ is also involved in the decoding phase discussed below.

In [4] the rule complexity of this reduction was improved as follows. It was noted that the recopying for increment and zero-test usually can be avoided by assigning different "master registers" to different states. The word "usually" means whenever the master register is changed after increment and zero-test, but is not changed after a decrement.

Hence, now the following allocation of master registers is chosen:

Register 1: q_1, q_6, q_9, q_{12}, q_{33}, q_{18}, q_{22}, q_{27}.

Register 2: q_3, q_4, q_7, q_{10}, q_{13}, q_{14}, q_{16}, q_{20}, q_{23}, q_{25}, q_{32}, q_h.

Another observation that we use to save even more instructions is the following: if an increment instruction has a unique entry point which is a zero-test, then such an increment can be embedded into the zero-test without using additional instructions. Clearly, the same transformation can be applied to multiple consecutive increments. The states q_{29}, q_{30}, and q_{31} do not appear in the register allocations above because we have embedded them into the zero-tests of q_{27} and q_{20}.

We proceed with evaluating the instruction complexity of the obtained generalized register machine by states. With the register allocation given above, recopying has been skipped for all transitions except $q_{13} \to q_1$ and $q_{23} \to q_{32}$. The table below shows the numbers of generalized register machine instructions associated with each generalized register machine instruction, and the numbers of generalized register machine instructions associated with the states q_{13} and q_{23} are

underlined. The necessity of at least two recopyings can be argued by inspecting the cycles $q_1 - q_6 - q_4 - q_7 - q_9 - q_{10} - q_{13} - q_1$ and $q_{23} - q_{25} - q_{32} - q_{23}$; these cycles do not have common nodes, and each cycle needs at least one recopying to have the value in the original register. This minimality has been further confirmed by computer search in the space of possible allocations of registers to states, furthermore showing the uniqueness of the optimal allocation modulo the symmetric assignment.

state	q_1	q_3	q_4	q_6	q_7	q_9	q_{10}	q_{12}	q_{13}	q_{33}
instructions	12	1	3	1	4	1	18	1	$\underline{5}$	1
state	q_{14}	q_{16}	q_{18}	q_{20}	q_{22}	q_{23}	q_{25}	q_{27}	q_{32}	q_h
instructions	6	3	3	3	1	$\underline{15}$	8	20	6	0

This gives a total of 112 instructions for a weakly universal generalized 2-register machine. To obtain weak-strong universality, the result has to be decoded into a third increment-only register, which means repeated division of the encoding by 7 with incrementing the new register in each cycle, iterated until a remainder is obtained. In fact, this means adding the following generalized register machine instructions:

$$
\begin{aligned}
&q_h : (\texttt{SUB}(2), h_1, h_7),\\
&h_i : (\texttt{SUB}(2), h_{i+1}, h_8), 1 \leq i \leq 5,\\
&h_6 : (\texttt{SUB}(2), \texttt{ADD}(1)\texttt{ADD}(3)q_h, h_8),\\
&h_7 : (\texttt{SUB}(1), \texttt{ADD}(2)h_7, q_h)
\end{aligned}
$$

with h_8 being the new halting state. The computation ends up with empty register 2, but still some "garbage" in register 1, which can be erased by taking an additional rule $h_8 : (\texttt{SUB}(1), h_8, h_9)$ and h_9 as the new halting state instead. Hence, in total this additional part costs 8 instructions for the decoding, plus an extra instruction to erase the rest of the encoding, i.e., the instruction labeled by h_8, resulting in the overall value for the generalized register machine instructions of 121 (with "cleaning") and 120 (without "cleaning"), respectively.

Yet for weak universality, several states and rules can be saved by simulating the weakly universal machine U_{20} of Korec, see [10], instead

of the strongly universal register machine U_{22}. The weakly universal register machine U_{20} does not use register 3, so we no longer need to carry out division by 19. The difference is only in the simulation block, so only instructions associated with states q_{23} and q_{27} are affected, as well as q_{30} and q_{31} work on different registers, which does not affect the number of generalized instructions, and instructions q_{25} and q_{29} are no longer present. Like in case of the strongly universal register machine, we embed the instructions q_{30} and q_{31} into the preceding zero-test of q_{20}.

We leave the same assignment of prime numbers to the registers and the same allocation of the main register, except that we reallocate q_{23} to register 1 and that we no longer have q_{25}. This leads to skipping recopying for all transitions except for $q_{13} \to q_1$ and $q_{16} \to q_{23}$; again, the associated numbers of instructions are underlined in the table below. The necessity of at least two recopyings can be argued by inspecting the cycles $q_1 - q_6 - q_4 - q_7 - q_9 - q_{10} - q_{13} - q_1$ and $q_{16} - q_{18} - q_{27} - q_{32} - q_{23} - q_{16}$; these cycles have no common nodes, and each cycle needs at least one recopying to have the value in the original register. This minimality has been further confirmed by computer search in the space of possible allocations of registers to states, furthermore again showing the uniqueness of the optimal allocation modulo the symmetric assignment for the weakly universal generalized register machine with embedded increments.

We have the following adjustment on the number of generalized instructions; the numbers to the left of each arrow are replaced by the numbers to the right of that arrow.

state	q_{16}	q_{23}	q_{25}	q_{27}
instructions	$3 \to \underline{4}$	$\underline{15} \to 8$	$8 \to 0$	$20 \to 14$

After having saved 20 instructions in this way, only $112 - 20 = 92$ generalized instructions remain.

3.4 Generalized Counter Automata

Generalized counter automata (GCAs for short) were introduced in [1] and also used in [4] and [3] with slightly different restrictions because

of how they then were simulated by the corresponding P systems. The reason to consider a generalization of counter automata is that sometimes the simulation costs (measured in the number of rules in the description of a P system) of a composite instruction is the same as that (or almost the same, but anyway less than the sum of those) for simulating an elementary instruction.

For a register machine $M = (m, B, l_0, l_h, P)$ consider the more general type of instructions $i : (q, M_-, N, M_+, q')$ where $q, q' \in Q$ are states, $N \subseteq R$ is a set of registers, and M_-, M_+ are multisets of registers. Such a register machine applies instruction i as follows: first, multiset M_- is subtracted from the register values (i.e., for each register $j \in R$, $M_-(j)$ is subtracted from the contents of register j; if at least one resulting value would be negative, the machine is blocked without producing any result); second, the subset N of registers is checked to be zero (if at least one of them is found to be non-zero, the machine is blocked without producing any result); third, the multiset M_+ is added to the register values (i.e., for each register $j \in R$, $M_+(j)$ is added to the contents of register j), and finally the state changes to q'.

The work of such a register machine, now also called a *generalized counter automaton* and written $M = (m, B, q_1, q_h, P)$, consists of derivation steps applying instructions, chosen in a non-deterministic way, associated with the current state. The computation starts in the initial state q_1, and we say that it halts if the final state q_h has been reached (which replaces the condition of reaching the final HALT-instruction labeled by l_h).

We start by presenting the small universal antiport P systems with inhibitors from [8]; let us call it GCA 1.

$1 : (q_1, \langle 1\rangle, \{\}, \langle 7\rangle, q_1)$,
$2 : (q_1, \langle\rangle, \{1\}, \langle 6\rangle, q_4)$,
$3 : (q_4, \langle 5\rangle, \{\}, \langle 6\rangle, q_4)$,
$4 : (q_4, \langle 6\rangle, \{5\}, \langle 5\rangle, q_{10})$,
$5 : (q_{10}, \langle 7, 6\rangle, \{\}, \langle 1, 5\rangle, q_{10})$,
$6 : (q_{10}, \langle 7\rangle, \{6\}, \langle 1\rangle, q_4)$,
$7 : (q_{10}, \langle\rangle, \{6, 7\}, \langle\rangle, q_1)$,
$8 : (q_{10}, \langle 6, 4\rangle, \{7\}, \langle\rangle, q_1)$,
$9 : (q_{10}, \langle 6, 5\rangle, \{7, 4\}, \langle\rangle, q_{18})$,
$10 : (q_{18}), \langle 5^3\rangle, \{\}, \langle 4\rangle, q_{18})$,
$11 : (q_{18}, \langle\rangle, \{5, 3\}, \langle 0\rangle, q_1)$,
$12 : (q_{18}, \langle 5^2, 0\rangle, \{5, 2\}, \langle\rangle, q_1)$,
$13 : (q_{18}, \langle 5^2, 2\rangle, \{5\}, \langle\rangle, q_1)$,
$14 : (q_{18}, \langle 5^2\rangle, \{5, 2, 0\}, \langle\rangle, q_1)$
$15 : (q_{18}, \langle 3, 4\rangle, \{5\}, \langle\rangle, q_1)$,
$16 : (q_{18}, \langle 5, 4\rangle, \{5\}, \langle 2, 3\rangle, q_1)$.

We now present a few variations of GCA 1, which have more instructions, but satisfy certain requirements that make them more suitable for a simulation by specific P systems.

For the variant to be simulated by anti-matter P systems, we require that for any instruction, M_- does not overlap with M_+; note that this condition is already fulfilled by GCA 1. Moreover, we note that, as it will be shown later, if M_- does not overlap with N, then the simulation (in terms of the number of instructions) is more efficient, but otherwise the simulation is still more efficient than in the case of splitting such an instruction into two instructions and simulating these two. Another requirement, due to the technicalities of the simulation, is that the halting must be in a state with no associated instructions (unlike in GCA 1, which halts in q_{18} if no instruction is applicable, its straightforward simulation would non-deterministically choose an instruction to simulate and fail, entering an infinite loop). The solution is to replace the last two rules with the following ones; let us call this resulting automaton GCA 2:

$$15 : (q_{18}, \langle 3\rangle, \{5\}, \langle\rangle, q_{32}), \qquad 16 : (q_{18}, \langle 5\rangle, \{5\}, \langle 2, 3\rangle, q_{32}),$$
$$17 : (q_{32}, \langle 4\rangle, \{\}, \langle\rangle, q_1), \qquad 18 : (q_{32}, \langle\rangle, \{4\}, \langle\rangle, q_h).$$

For the simulation with many catalysts, we need different requirements. However, also in this case we need that the GCA halts in a state with no associated instructions, so we take GCA 2 as the basis. While it no longer matters whether M_- and N are disjoint, we require that M_+ does not overlap with either M_- or N. To fulfill this condition, we take GCA 2 and replace instruction 4 by new instructions 4 and $4'$ below. Let us call the result GCA 3. Moreover, for technical reasons, we have to produce the output in a register that only has increment instructions associated to it, and have all other registers empty in the end, hence, instruction 18 is replaced by instructions 18–21 below. Let us call the result GCA 4.

$$4 : (q_4, \langle 6\rangle, \{5\}, \langle\rangle, q_{4'}) \qquad 4' : (q_{4'}, \langle\rangle, \{\}, \langle 5\rangle, q_{10}),$$
$$18 : (q_{32}, \langle 0\rangle, \{4\}, \langle 8\rangle, q_{32}), \qquad 20 : (q_{32}, \langle 6\rangle, \{4\}, \langle\rangle, q_{32}),$$
$$19 : (q_{32}, \langle 1\rangle, \{4\}, \langle\rangle, q_{32}), \qquad 21 : (q_{32}, \langle\rangle, \{0, 1, 4, 6\}, \langle\rangle, q_h).$$

Finally, for a simulation with multiple catalysts (in fact, 8), the setting is more restricted. A coupling function f_c is considered, which is a bijective mapping from the set of registers to the same set, without a fixed point. Not only is M_- forbidden to contain more than one copy of the same register, but we need all the sets $supp(M_-)$, $f_c(supp(M_-)$ and N to be disjoint. After having carefully inspected the Korec machines and the resulting GCAs from [4], we decided to use the following coupling function f_c:

r:	0	1	2	3	4	5	6	7
$f_c(r)$	6	5	7	4	3	1	0	2

While we keep instructions 1–9 identical to the ones listed above, the rest of instructions is presented below. We call the result GCA 5 (in [3] such a variant of counter automata is called "weakly generalized").

$10 : (q_{18}, \langle 5\rangle, \{\}, \langle\rangle, q_{20})$,
$10' : (q_{20}, \langle 5\rangle, \{\}, \langle 4\rangle, q_{16})$,
$10'' : (q_{16}, \langle 5\rangle, \{\}, \langle\rangle, q_{18})$,
$11 : (q_{18}, \langle\rangle, \{3, 5\}, \langle 0\rangle, q_1)$,
$12 : (q_{16}, \langle 0\rangle, \{2, 5\}, \langle\rangle, q_1)$,
$13 : (q_{16}, \langle 2\rangle, \{5\}, \langle\rangle, q_{32})$,
$14 : (q_{16}, \langle\rangle, \{0, 2, 5\}, \langle\rangle, q_{32})$,
$15 : (q_{18}, \langle 3\rangle, \{5\}, \langle\rangle, q_{32})$,
$16 : (q_{20}, \langle\rangle, \{5\}, \langle 2, 3\rangle, q_{32})$,
$17 : (q_{32}, \langle 4\rangle, \{\}, \langle\rangle, q_1)$,
$18 : (q_{32}, \langle\rangle, \{4\}, \langle\rangle, q_h)$.

As in the case of multiple catalysts, the input must be moved to an increment-only register, but for technical reasons the other registers do not have to be cleaned by instructions of the GCA. Hence, we replace instruction 18 by the instructions below, with λ being the new final state, and we call the result GCA 6.

$$18 : (q_{32}, \langle 0\rangle, \{4\}, \langle 8\rangle, q_{32}), \qquad 18' : (q_{32}, \langle\rangle, \{0, 4\}, \langle\rangle, \lambda).$$

4 Antimatter

We now consider P systems with matter/anti-matter annihilation rules, see [1].

Theorem 1 *(see [1]) There exist small universal P systems with non-cooperative rules and matter/anti-matter annihilation rules – with 9 annihilation rules and, in total,* **53** *rules in the accepting case,* 59 *rules in the generating case, and* 57 *rules in the computing case.*

Table 1. A small universal P system with anti-matter.

$$
\begin{aligned}
\Pi &= \left(O,\ [\ ]_1\ , q_1, R_1, 1, 1\right) \text{ where} \\
O &= \{l_2, l_4, l_6, l_7, l_8, l_9, l_{11}, l_{12}, l'_{12}, l_{13}, l'_{13}, l_{14}, l'_{14}, l_{15}, l_{16}, l'_{16}, l_{18}\} \\
&\cup \{q_1, q_4, q_{10}, q_{18}, q_{32}, q_h\} \cup \{a, a^- \mid a \in \{a_j \mid 0 \le j \le 7\} \cup \{\#\}\}
\end{aligned}
$$

and R_1 contains the following rules:

$$
\begin{array}{lll}
q_1 \to q_1 a_1{}^- a_7, & & \\
q_1 \to l_2 a_1{}^-, & l_2 \to q_4 \# a_6, & \\
q_4 \to q_4 a_5{}^- a_6, & & \\
q_4 \to l_4 a_5{}^-, & l_4 \to q_{10} \# a_6{}^- a_5, & \\
q_{10} \to q_{10} a_7{}^- a_6{}^- a_1 a_5, & & \\
q_{10} \to l_6 a_6{}^-, & l_6 \to q_4 \# a_7{}^- a_1, & \\
q_{10} \to l_7 a_6{}^- a_7{}^-, & l_7 \to q_1 \#\#, & \\
q_{10} \to l_8 a_7{}^-, & l_8 \to q_1 \# a_6{}^- a_4{}^-, & \\
q_{10} \to l_9 a_7{}^- a_4{}^-, & l_9 \to q_{18} \#\# a_6{}^- a_5{}^-, & \\
q_{18} \to q_{18} a_5{}^- a_5{}^- a_5{}^- a_4, & & \\
q_{18} \to l_{11} a_5{}^- a_3{}^-, & l_{11} \to q_1 \#\# a_0, & \\
q_{18} \to l_{12} a_5{}^- a_5{}^- a_0{}^-, & l_{12} \to l'_7 a_5{}^- a_2{}^-, & \\
q_{18} \to l_{13} a_5{}^- a_5{}^- a_2{}^-, & l_{13} \to l'_{13} a_5{}^-, & l'_{13} \to q_1 \#, \\
q_{18} \to l_{14} a_5{}^- a_5{}^-, & l_{14} \to l'_{14} a_5{}^- a_2{}^- a_0{}^-, & l'_{14} \to q_1 \#\#\#, \\
q_{18} \to l_{15} a_5{}^-, & l_{15} \to q_{32} \# a_3{}^-, & \\
q_{18} \to l_{16} a_5{}^-, & l_{16} \to l'_{16} a_5{}^-, & l'_{16} \to q_{32} \# a_2 a_3, \\
q_{32} \to q_1 a_4{}^-, & & \\
q_{32} \to l_{18} a_4{}^-, & l_{18} \to q_h \#, & \\
\#^- \to \#^4, & \# \to \#^4, & (\#\#^- \to \lambda), \\
a_r{}^- \to \#^-, & (a_r a_r{}^- \to \lambda), & 0 \le r \le 7.
\end{array}
$$

For a generalized counter automaton $M = (m, B, q_1, q_h, P)$, let

$$k = 1 + \max_{i:(q,M_-,N,M_+,q')\in P}(|M_-|, |N|).$$

Common for different instructions of M, we consider the following rules:

$$\#^- \to \#^k,\ \# \to \#^k,\ \#\#^- \to \lambda,\ a_r \to \#^-,\ a_r a_r^- \to \lambda,\ r \in R.$$

We recall the main construction block: the simulation of instruction $i : (q, M_-, N, M_+, q') \in P$. First we consider the case when M_- and N have no common elements, and moreover, we also assume that M_- does not overlap with M_+.

$$q \to l_i \prod_{r\in N} {a_r}^-,\ l_i \to q'(\prod_{r\in N} \#)(\prod_{r\in M_-} {a_r}^-) \prod_{r\in M_+} a_r.$$

If the zero-test set N is empty, then the first step is a simple renaming and can be combined with the second step, yielding one rule

$$q \to q'(\prod_{r\in M_-} {a_r}^-) \prod_{r\in M_+} a_r.$$

Clearly, if M_- and N overlap, such an instruction can be broken down into two subsequent instructions of the generalized counter automaton. However, a more efficient solution with only three rules exists:

$$q \to l_i \prod_{r\in M_-} {a_r}^-,\ l_i \to l_i' \prod_{r\in N} {a_r}^-,\ l_i' \to q(\prod_{r\in N} \#) \prod_{r\in M_+} a_r.$$

The accepting case is shown by the construction in Table 1, simulating GCA 2.

5 One Catalyst and One Multi-Stable Catalyst

A conditional decrement is performed by letting the multi-stable catalyst try to remove one register object, the states of the catalyst being associated to the registers. In the next step, the "program object" verifies whether the state of the multi-stable object was changed, and the proper transition is modeled. Based on this idea, a few universal P systems have been constructed in [4], depending on whether the strong Korec machine, the weak one, or the one reduced to two working registers is simulated, whether the output is decoded, and whether the feature of toxic objects is used, see the upper part of Table 4.

6 Multiple Catalysts

In this section, we do not limit the number of catalysts, but aim at a small number of rules, based on the results recently established in [3].

Theorem 2 *(see [3]) There exists a small universal catalytic P system with 8 catalysts and* **98** *rules. Using toxic objects, the number of rules can be reduced to 89.*

Besides the rules associated to the instructions, we use rules

$$\{\# \to \#\} \cup \{c_r o_r \to c_r d_r,\ c_r d_r \to c_r,\ c_r e_r \to c_r \#,\\ c_{f_c(r)} e_r \to c_{f_c(r)},\ d_r \to \# \mid 0 \le r \le 7\}.$$

For a general instruction j of wGCA, $j : (q_i, M_-, N, M_+, q_k)$ it suffices to have the following three rules:

$q_i \to p_j E_{M_-} D_{m,M_-,N}$, $p_j \to p_j D'_{m,M_-}$, $p_j \to q_k D_m O_{M_+}$ where

$$\begin{aligned} D_{m,M_-,N} &= \prod_{i \in [1..m] \setminus (supp(M_-) \cup N)} d_i, \\ D'_{m,M_-} &= \prod_{i \in [1..m] \setminus \{r, c(r) \mid r \in M_-\}} d_i, \\ E_{M_-} &= \prod_{r \in M_-} e_r, \text{ and} \\ O_{M_+} &= \prod_{r \in M_+} o_r. \end{aligned}$$

The construction given in Table 2 was used for the proof, simulating GCA 6 (for conciseness, the multiset of objects d_r, $0 \le r \le 7$, $r \notin M$, is denoted by $d(M)$, and we omit the braces denoting M).

In addition to w_1, to the initial configuration we add the number of symbols o_1 corresponding with the code of the machine to be simulated and the number of symbols o_0 corresponding with the input number to

Table 2. A universal catalytic P system with 8 catalysts.

$$
\begin{aligned}
\Pi &= (O, \Sigma, C = \{c_r \mid 0 \le r \le 7\}, \mu = [\ ]_1, w_1, R_1, f = 1),\\
O &= \{o_r, d_r, e_r \mid 0 \le r \le 7\} \cup \{\#, p_{10'}, p_{10''}, p_{18'}, o_8\}\\
&\cup \{p'_j \mid j \in \{1,3,4,5,6,8,9,10,10',10'',12,13,15,17,18'\}\}\\
&\cup \{p_j \mid 1 \le j \le 18\} \cup \{q_1, q_4, q_{10}, q_{16}, q_{18}, q_{20}, q_{32}\},\\
R_1 &= R \cup \{\# \to \#\} \cup \{c_r o_r \to c_r d_r,\ c_r d_r \to c_r,\ c_r e_r \to c_r \#,\\
&\qquad c_{f_c(r)} e_r \to c_{f_c(r)},\ d_r \to \# \mid 0 \le r \le 7\},\\
w_1 &= q_1 d(),
\end{aligned}
$$

and the rules from the set R are listed below:

$$
\begin{array}{lll}
q_1 \to p_1 e_1 d(1), & p_1 \to p'_1 d(1,5), & p'_1 \to q_1 d() o_7,\\
q_1 \to p_2 d(1), & p_2 \to q_4 d() o_6, & \\
q_4 \to p_3 e_5 d(5), & p_3 \to p'_3 d(1,5), & p'_3 \to q_4 d() o_6,\\
q_4 \to p_4 e_6 d(5,6), & p_4 \to p'_4 d(0,6), & p'_4 \to q_{10} d() o_5,\\
q_{10} \to p_5 e_6 e_7 d(6,7), & p_5 \to p'_5 d(0,2,6,7), & p'_5 \to q_{10} d() o_1 o_5,\\
q_{10} \to p_6 e_7 d(6,7), & p_6 \to p'_6 d(2,7), & p'_6 \to q_4 d() o_1,\\
q_{10} \to p_7 d_{6,7}, & p_7 \to q_1 d(), & \\
q_{10} \to p_8 e_4 e_6 d(4,6,7), & p_8 \to p'_8 d(0,3,4,6), & p'_8 \to q_1 d(),\\
q_{10} \to p_9 e_5 e_6 d(4,5,6,7), & p_9 \to p'_9 d(0,1,5,6), & p'_9 \to q_{18} d(),\\
q_{18} \to p_{10} e_5 d(5), & p_{10} \to p'_{10} d(1,5), & p'_{10} \to q_{20} d(),\\
q_{20} \to p_{10'} e_5 d(5), & p_{10'} \to p'_{10'} d(1,5), & p'_{10'} \to q_{16} d(),\\
q_{16} \to p_{10''} e_5 d(5), & p_{10''} \to p'_{10''} d(1,5), & p'_{10''} \to q_{18} d(),\\
q_{18} \to p_{11} d(3,5), & p_{11} \to q_1 d() o_0, & \\
q_{16} \to p_{12} e_0 d(0,2,5), & p_{12} \to p'_{12} d(0,6), & p'_{12} \to q_1 d(),\\
q_{16} \to p_{13} e_2 d(2,5), & p_{13} \to p'_{13} d(2,7), & p'_{13} \to q_{32} d(),\\
q_{16} \to p_{14} d(0,2,5), & p_{14} \to q_{32} d(), & \\
q_{18} \to p_{15} e_3 d(3,5), & p_{15} \to p'_{15} d(3,4), & p'_{15} \to q_{32} d(),\\
q_{20} \to p_{16} d(5), & p_{16} \to q_{32} d() o_2 o_3, & \\
q_{32} \to p_{17} e_4 d(4), & p_{17} \to p'_{17} d(3,4), & p'_{17} \to q_1 d(),\\
q_{32} \to p_{18} e_0 d(0,4), & p_{18} \to p'_{18} d(0,6), & p'_{18} \to q_{32} d() o_8,\\
q_{32} \to p_{18'} d(0,4), & p_{18'} \to d(). &
\end{array}
$$

this machine; the result of the simulation is represented by the number of symbols o_8 in the final configuration.

Going to the extreme with the number of catalysts, we can even obtain a real-time simulation of register machines, see [4], and obtain a universal P system with even less rules.

Theorem 3 *(see [4]) There exists a small universal purely catalytic P system with 21 catalysts and* **74** *rules. Using toxic objects, the number of rules can be reduced to 64.*

Let S be a finite multiset and $S' \subseteq S$, and let $e_S(S')$ be a string representing the multiset $S \setminus S'$. We note that we will use $e_S(\lambda)$ (as λ denotes the empty multiset) for representing the multiset S itself. We define the multiset S and the corresponding mapping e_S by

$$e_S(\lambda) = d_{0,-} \cdots\ d_{4,-}\ d_{5,-}\ d_{5,-}\ d_{5,-}\ d_{6,-}\ d_{7,-}\ d_{0,0}\ \cdots\ d_{7,0},$$

and for a finite multiset L, by $g(L)$ we denote a string representing a multiset consisting of objects o_r for each occurrence of r in L. Besides the rules associated to instructions, the following rules are used:

$$\begin{aligned} R &= \{c_{r,-}o_r \to c_{r,-},\ c_{r,-}d \to c_{r,-}\# \mid r \in \{0,\cdots,7\}\} \\ &\cup \{c_{r,0}o_r \to c_{r,0}\#,\ c_{r,0}d_{r,0} \to c_{r,0},\ c_{r,-}d_{r,-} \to c_{r,-}, \\ &\quad c_\# d_{r,-} \to c_\#\# \mid r \in \{0,\cdots,7\}\} \\ &\cup \{c_d d' \to c_d,\ c_d d \to c_d,\ c_\# d' \to c_\#\#,\ c_\#\# \to c_\#\#\}. \end{aligned}$$

If we take S to be the finite multiset over $\{d_{r,-}\ ,\ d_{r,0}\ \mid\ 1 \leq r \leq m\}$ such that, for $1 \leq r \leq m$, $S(d_{r,0}) = 1$ if $r \in \bigcup_{j:(q,M_-,N,M_+,q')\in P} N$ and $S(d_{r,0}) = 0$ otherwise, as well as $S(d_{r,-}) = \max\{M_-(r) \mid j : (q, M_-, N, M_+, q') \in P\}$, then the simulation of an instruction $j : (q, M_-, N, M_+, q')$ is initiated by the catalytic rule

$$c_p q \to c_p q' d' e_S(\langle {d_{r,-}}^{M_-(r)} \mid r \in supp(M_-)\rangle \cup \langle d_{r,0} \mid r \in N\rangle) g(M_+).$$

The construction for the proof simulating GCA 4 is given in Table 3.

Table 3. A universal purely catalytic P system with 21 catalysts.

$\Pi = (O, C, \{o_1, o_2\}, \{o_8\}, w, R)$ where

$O = C \cup \{o_r \mid 0 \leq r \leq 8\} \cup Q \cup \{d_{r,-}, d_{r,0} \mid 0 \leq i \leq 7\} \cup \{d, d', \#\}$,

$C = \{c_{r,-}, c_{r,0} \mid 0 \leq r \leq 7\} \cup \{c_d, c_p, c_\#\}$,

$w = c_{0,-} \cdots c_{4,-}\ c_{5,-}\ c_{5,-}\ c_{5,-}\ c_{6,-}\ c_{7,-}\ c_{0,0}\ \cdots\ c_{7,0}\ \ c_d c_p c_\#$
$dd' p_1 e_S(d_{1,-})$, and the set R consists of the following rules:

$R = \{c_{r,-} o_r \to c_{r,-},\ c_{r,-} d \to c_{r,-}\# \mid r \in \{0, \cdots, 7\}\}$

$\cup\ \{c_{r,0} o_r \to c_{r,0}\#,\ c_{r,0} d_{r,0} \to c_{r,0},\ c_{r,-} d_{r,-} \to c_{r,-},$
$c_\# d_{r,-} \to c_\#\# \mid r \in \{0, \cdots, 7\}\}$

$\cup\ \{c_d d' \to c_d,\ c_d d \to c_d,\ c_\# d' \to c_\#\#,\ c_\#\# \to c_\#\#\}$

$\cup\ \{c_p q_1 \to c_p q_1 d' e_S(d_{1,-}) o_7,\ c_p q_1 \to c_p q_4 d' e_S(d_{1,0}) o_6,$
$c_p q_4 \to c_p q_4 d' e_S(d_{5,-}) o_6,\ c_p q_4 \to c_p q_{4'} d' e_S(d_{6,-}\, d_{5,0}),$
$c_p q_{4'} \to c_p q_{10} d' e_S(\lambda) o_5,$
$c_p q_{10} \to c_p q_{10} d' e_S(d_{6,-}\, d_{7,-}) o_1 o_5,$
$c_p q_{10} \to c_p q_4 d' e_S(d_{7,-}\, d_{6,0}) o_1,$
$c_p q_{10} \to c_p q_1 d' e_S(d_{6,0}\, d_{7,0}),\ c_p q_{10} \to c_p q_1 d' e_S(d_{4,-}\, d_{6,-}\, d_{7,0}),$
$c_p q_{10} \to c_p q_{18} d' e_S(d_{5,-}\, d_{6,-}\, d_{4,0}\, d_{7,0}),$
$c_p q_{18} \to c_p q_{18} d' e_S(d_{5,-}\, d_{5,-}\, d_{5,-}) o_4,$
$c_p q_{18} \to c_p q_1 d' e_S(d_{3,0}\, d_{5,0}) o_0,$
$c_p q_{18} \to c_p q_1 d' e_S(d_{0,-}\, d_{5,-}\, d_{5,-}\, d_{2,0}\, d_{5,0}) o_4,$
$c_p q_{18} \to c_p q_{32} d' e_S(d_{2,-}\, d_{5,-}\, d_{5,-}\, d_{5,0}) o_4,$
$c_p q_{18} \to c_p q_{32} d' e_S(d_{5,-}\, d_{5,-}\, d_{0,0}\, d_{2,0}\, d_{5,0}) o_4,$
$c_p q_{18} \to c_p q_{32} d' e_S(d_{3,-}\, d_{5,0}),$
$c_p q_{18} \to c_p q_{32} d' e_S(d_{5,-}\, d_{5,0}) o_2 o_3, c_p q_{32} \to c_p q_1 d' e_S(d_{4,-}),$
$c_p q_{32} \to c_p q_{32} d' e_S(d_{0,-}\, d_{4,0}) o_8,$
$c_p q_{32} \to c_p q_{32} d' e_S(d_{1,-}\, d_{4,0}),\ c_p q_{32} \to c_p q_{32} d' e_S(d_{6,-}\, d_{4,0}),$
$c_p q_{32} \to c_p e_S(d_{0,0}\, d_{1,0}\, d_{4,0}\, d_{6,0})\}$.

7 Universal P Systems with Two Catalysts

We now take the new simulation from [3]. For a register machine with only two working registers, we need 5 rules per instruction plus 11 rules; cleaning happens by the P system itself at the end of a successful simulation (for example, see [16], for detailed arguments), but recopying of the result to an extra non-decrementable register at the end of the simulation is needed for the case of weak universality. Hence, we obtain a weakly-strongly / weakly universal catalytic P system with two catalysts, having **611/476** rules (improving the result of 1091/848 rules from [4]). Using the feature of toxic objects, the simulation costs are reduced to 5 rules per instruction plus 8, yielding **608/473** rules (improving the result of 726/564 rules from [4]).

8 Universal Purely Catalytic P Systems

It was stated in [3] that the constructions obtained there for cat_m also hold for $pcat_{m+2}$: one catalyst can take care of the states and program symbols, while one more catalyst can perform the trapping rules. Hence, any generalized register machine with m decrementable registers and s generalized SUB-instructions can be simulated by a *purely* catalytic P system with $m+2$ catalysts and $5s+5m+1$ rules. Therefore, the results with 611, 476, 608 and 473 rules for universal catalytic P systems with 2 catalysts also hold for universal purely catalytic P systems with 4 catalysts.

It was shown in [3] that purely catalytic P systems with 9 catalysts are strongly universal with $6 \times 16 + 6 \times 8 + 1 = 145$ rules. Using the formula $6s + 6m + 1$ from [17], simulating the weakly universal generalized register machine we obtain a weakly universal purely catalytic P system with 8 catalysts and $6 \times 15 + 6 \times 7 + 1 = \mathbf{133}$ rules (improving the result of 171 rules from [4]). In a similar approach, consider the weakly-strongly/weakly universal generalized register machine with $m = 2$ decrementable registers and $s = 93/s = 120$ generalized register machine instructions. Again using the formula $6s + 6m + 1$ from the recent paper [17], we obtain a weakly-strongly/weakly universal

purely catalytic P system with 3 catalysts and $6 \times 120 + 6 \times 2 + 1 = \mathbf{733}/6 \times 93 + 6 \times 2 + 1 = \mathbf{571}$ rules, thus improving the previously best known results of 1091/848 rules, respectively.

9 Conclusions

It has been known that only one bi-stable catalyst suffices for computational completeness of P systems (having non-cooperative rules besides the bi-catalytic ones) and that purely catalytic P systems with three catalysts are computationally complete. With two catalysts computational completeness can be obtained if one of them is bi-stable, see [4].

Generalizing counter automata by allowing them to perform multiple operations on multiple registers, a few small generalized counter automata are obtained (from 16 rules to 22 rules), depending on the specific requirements of P systems that would simulate them. Generalized counter automata are a very convenient tool for constructing small universal P systems. For instance, small strongly universal P systems with anti-matter with 9 annihilation rules and, in total, **53** rules in the accepting case, 59 rules in the generating case, and 57 rules in the computing case can be constructed.

By optimizing the reduction of the universal register machines U_{22} and U_{20} to register machines with two working registers, in [4] a strongly universal register machine with 120 instructions and two decrementable registers and a weakly universal register machine with 92 instructions and two registers have been obtained.

The now best known results for catalytic systems are summarized in Table 4. It describes universal (purely or not) catalytic P systems with and without toxic objects where the type of universality ranges from strong over weak-strong to weak. The results in the upper part of the table correspond to one normal catalyst and one m-stable catalyst, $2 \leq m \leq 8$, while the results in the lower part of the table correspond to k catalysts, $2 \leq k \leq 21$. Depending on all these features, the overall number of rules varies from 43, top right, to 733, bottom left.

The new results elaborated in this paper are indicated in boldface. If some entry of a table contains "+", then the reference following it in-

Table 4. Number of rules in universal catalytic P systems. We write "s" for strongly universal P systems, "ws" for weakly-strongly universal P systems and "w" for weakly universal P systems, "tox" for P systems with toxic objects, "cat_k" for k catalysts, $mcat$ for an m-stable catalyst, and "p" indicates P systems without non-cooperative rules.

Feature	s	ws	w	s,tox	ws,tox	w,tox
$p8cat$, $pcat$	$61_{[4]}$			$47_{[4]}$		
$p7cat$, $pcat$			$56_{[4]}$			$43_{[4]}$
$p2cat$, $pcat$		$483_{[4]}$	$371_{[4]}$		$362_{[4]}$	$278_{[4]}$
$pcat_{21}$	$74_{[4]}$					
$pcat_{20}$				$64_{[4]}$		
$pcat_{10}$	$98_{[3]}$			$89_{[3]}$		
cat_8	$98_{[3]}$			$89_{[3]}$		
$pcat_9$	$145_{[3]}$					
$pcat_8$			**$133_{[17]+[4]}$**	$120_{[4]}$		
$pcat_7$						$111_{[4]}$
$pcat_4$		**$611_{+[4]}$**	**$476_{+[4]}$**		**$608_{+[4]}$**	**$473_{+[4]}$**
cat_2		**$611_{+[4]}$**	**$476_{+[4]}$**		**$608_{+[4]}$**	**$473_{+[4]}$**
$pcat_3$		**$733_{[17]+[4]}$**	**$571_{[17]+[4]}$**			
$pcat_2$					$726_{[4]}$	$564_{[4]}$

dicates where the underlying simulating model has been studied, while the reference preceding "+" (if indicated, otherwise we imply the current paper) indicates where the currently best known simulated complexity has been obtained. Three small universal P systems, namely, the one with anti-matter and 53 rules, the catalytic one with 8 catalysts and 98 rules, and the purely catalytic one with 21 catalysts and 74 rules, were chosen to be presented explicitly in Tables 1, 2, and 3.

Acknowledgements The work of the third author was supported by the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).

References

[1] A. Alhazov, B. Aman, R. Freund, Gh. Păun. *Matter and Anti-Matter in Membrane Systems.* In: H. Jürgensen, J. Karhumäki, A. Okhotin (Eds.): 16th International Workshop on Descriptional Complexity of Formal Systems, DCFS 2014, Lecture Notes in Computer Science **8614**, 2014, 65–76.

[2] A. Alhazov, R. Freund. *P Systems with Toxic Objects.* In: M. Gheorghe, G. Rozenberg, A. Salomaa, P. Sosík, C. Zandron: Membrane Computing - 15th International Conference, CMC 2014, Prague, Lecture Notes in Computer Science. **8961**, 2014, 99–125.

[3] A. Alhazov, R. Freund. *Small Catalytic P Systems.* In: M. Dinneen, Ed., Workshop on Membrane Computing, Auckland, 2015, 1–16.

[4] A. Alhazov, R. Freund. *Variants of Small Universal P Systems with Catalysts.* Fundamenta Informaticae. **138**(1-2), 227–250, 2015.

[5] A. Alhazov, Yu. Rogozhin, S. Verlan. *On Small Universal Splicing Systems.* International Journal of Foundations of Computer Science. **23** (7), 2012, 1423–1438.

[6] A. Alhazov, S. Verlan. *Minimization Strategies for Maximally Parallel Multiset Rewriting Systems.* Theoretical Computer Science. **412** (17), 2011, 1581–1591.

[7] R. Freund, L. Kari, M. Oswald, P. Sosík. *Computationally Universal P Systems without Priorities: Two Catalysts Are Sufficient.* Theoretical Computer Science. **330** (2), 2005, 251–266.

[8] R. Freund, M. Oswald. *A Small Universal Antiport P System with Forbidden Context.* In: H. Leung, G. Pighizzini (Eds.): Proceedings of the 8th International Workshop on Descriptional Complexity of Formal Systems, DCFS 2006, Las Cruces, New Mexico, USA, 2006, New Mexico State University, 2006, 259–266.

[9] S. Ivanov, E. Pelz, S. Verlan. *Small Universal Non-deterministic Petri Nets with Inhibitor Arcs.* In: H. Jürgensen, J. Karhumäki, A. Okhotin (Eds.): 16th International Workshop on Descriptional Complexity of Formal Systems, DCFS 2014, Lecture Notes in Computer Science **8614**, 2014, 186–197.

[10] I. Korec. *Small Universal Register Machines.* Theoretical Computer Science. **168**, 1996, 267–301.

[11] M.L. Minsky. *Computation: Finite and Infinite Machines.* Prentice Hall, Englewood Cliffs, New Jersey, USA, 1967.

[12] Gh. Păun. *Computing with Membranes.* Journal of Computer and System Sciences. **61** (1), 108–143 (2000) (and Turku Center for Computer Science – TUCS Report 208, November 1998, `www.tucs.fi`).

[13] Gh. Păun. *Membrane Computing. An Introduction.* Springer, 2002.

[14] Gh. Păun, G. Rozenberg, A. Salomaa. *The Oxford Handbook of Membrane Computing.* Oxford University Press, 2010.

[15] G. Rozenberg, A. Salomaa (Eds.). *Handbook of Formal Languages,* 3 volumes. Springer, 1997.

[16] P. Sosík, M. Langer. *Improved Universality Proof for Catalytic P Systems and a Relation to Non-Semilinear Sets.* In: S. Bensch, R. Freund, F. Otto (Eds.): Sixth Workshop on Non-Classical Models of Automata and Applications (NCMA 2014), books@ocg.at, BAND 304, 2014, 223–233.

[17] P. Sosík, M. Langer. *Small Catalytic P Systems Simulating Register Machines.* Theoretical Computer Science, accepted, 2015.

[18] P systems webpage. http://ppage.psystems.eu.

Artiom Alhazov, Rudolf Freund, Petr Sosík Received November 2, 2015

Artiom Alhazov
Institute of Mathematics and Computer Science
Str. Academiei 5,
Chişinău, MD-2028, Moldova
E–mail: artiom.alhazov@math.md

Rudolf Freund
Faculty of Informatics, TU Wien
Favoritenstraße 9-11, 1040 Wien, Austria
E–mail: rudi@emcc.at

Petr Sosík
Research Institute of the IT4Innovations Centre of Excellence
Faculty of Philosophy and Science, Silesian University in Opava
74601 Opava, Czech Republic
E–mail: petr.sosik@fpf.slu.cz