Small Universal Antiport P Systems and Universal Multiset Grammars

Rudolf Freund, Marion Oswald

Faculty of Informatics
Vienna University of Technology
Favoritenstr. 9–11, A–1040 Vienna, Austria
{rudi,marion}@emcc.at

Summary. Based on the construction of a universal register machine (see [7]) we construct a universal antiport P system working with 31 rules in the maximally parallel mode in one membrane, and a universal antiport P system with forbidden context working with 16 rules in the sequential derivation mode in one membrane for computing any partial recursive function on the set of natural numbers. For accepting/generating any arbitrary recursively enumerable set of natural numbers we need 31/33 and 16/18 rules, respectively. As a consequence of the result for antiport P systems with forbidden context we immediately infer similar results for forbidden random context multiset grammars with arbitrary rules.

1 Introduction

The most interesting variant of communication P systems uses symport/antiport rules (see [10]). The application of an antiport rule x/y assigned to a membrane exchanges the multiset described by x inside the membrane with the multiset y outside the membrane. In antiport P systems with forbidden context, the application of an antiport rule $(x/y, z)$ with forbidden random context z is only possible, if z is not a sub-multiset of the multiset of objects inside the membrane. (A generalized variant of this model was already investigated in [5].)

We consider antiport P systems and antiport P systems with forbidden context and construct universal systems for these variants of P systems based on the construction of a universal register machine (see [7]); the universal antiport P systems constructed in this paper work in the maximally parallel mode and need 31/31/33 antiport rules in one membrane for computing any partial recursive function on the set of natural numbers/accepting/generating any arbitrary recursively enumerable set of natural numbers, and the universal antiport P systems with forbidden context work in the sequential derivation mode and need only 16/16/18 rules in one membrane for these purposes.

As shown in [8], multiset grammars with arbitrary multiset productions cannot generate all recursively enumerable sets of natural numbers. An antiport rule with forbidden context $(x/y, z)$ can be interpreted as a multiset production with forbidden context $(x \to y, z)$ which is only applicable if the forbidden multiset z is not a sub-multiset of the multiset of objects constituting the current sentential form. Hence, as a consequence of the results obtained for antiport P systems with forbidden context, we immediately obtain similar results for forbidden random context multiset grammars over a one-letter alphabet, e.g., interpreting the multiset a^n as the natural number n we can say that any recursively enumerable set of natural numbers can be generated by a forbidden context multiset grammar with at most 23 multiset productions with forbidden context.

2 Definitions

For well-known notions and basic results from the theory of formal languages, the reader is referred to [4] and [13]. We only give some basic notations first: λ denotes the empty word as well as the empty multiset; by $\mathbb{N}$ we denote the set of non-negative integers (i.e., natural numbers). In the following, two sets of natural numbers are considered to be equal if they only differ at most by 0. By NRE we denote the set of all recursively enumerable sets of natural numbers, and by $cNRE$ we denote the set of all partial recursive functions $f : \mathbb{N} \to \mathbb{N}$.

2.1 Multiset Grammars

In this paper we consider several types of multiset grammars (various interesting models of grammars for generating multisets of objects were considered in [8], and multiset automata were investigated in [3]). In fact, we could also define the corresponding types of grammars generating strings which then are interpreted as multisets.

In the following, a multiset of objects will simply be represented by any string containing exactly the same number of each object as the multiset.

A *multiset grammar* is a construct

$$G = (V, T, S, P)$$

with V being a set of *symbols* (or *objects*), $T \subset V$ is the set of *terminal symbols (terminal objects)*, S is the *initial multiset* over V and P is a (finite) set of *multiset productions* of the form $u \to v$ with u and v being multisets over V. The application of the production $u \to v$ to a multiset x has the effect of replacing the multiset u contained in x by the multiset v. We consider all derivations starting from the multiset S and using productions from P and finally yielding a terminal multiset (i.e., a multiset only consisting of objects from T); the set of terminal multisets generated in that way is denoted by $L(G)$.

A *forbidden random context multiset grammar* is a construct

$$G = (V, T, S, R)$$

with the rules in R being of the form $(u \rightarrow v, z)$ where $u \rightarrow v$ is an arbitrary multiset production of weight $\max\{|u|, |v|\}$ (where $|u|$ denotes the size of the multiset u, i.e., the total number of objects in u) and z is a (finite) multiset over V; the application of the rule $(u \rightarrow v, z)$ to a multiset x has the effect of replacing the sub-multiset u from x by the multiset v, but only if z *(forbidden context)* does not occur as a sub-multiset in x. Again we consider all terminal derivations starting from S and using rules from R, and we denote the set of terminal multisets generated in that way by $L(G)$.

The set of multisets $L(G)$ generated by a (forbidden random context) multiset grammar over a one-letter alphabet can be seen as the corresponding set of natural numbers, and we denote the sets of natural numbers generated by multiset grammars with arbitrary multiset productions and forbidden random context multiset grammars with arbitrary multiset productions by $N(arb)$ and $NfRC(arb)$, respectively. It is well known (e.g., see [8]) that $N(arb) \subsetneqq NRE$.

The sets of natural numbers accepted/generated by forbidden random context multiset grammars with at most s symbols, at most n arbitrary multiset productions of weight at most k and with the size of the forbidden multisets of at most l are denoted by $aNO_sfRC_l(arb_k)_n$ and $gNO_sfRC_l(arb_k)_n$, respectively; the corresponding variants of forbidden context multiset grammars computing partial recursive functions $f : \mathbb{N} \rightarrow \mathbb{N}$ are denoted by $cNO_sfRC_l(arb_k)_n$.

2.2 Antiport P Systems

The reader is supposed to be familiar with basic elements of membrane computing, e.g., from [2] and [11]; comprehensive information can be found on the P systems web page `http://psystems.disco.unimib.it`.

An (extended) antiport P system (of degree $m \geq 1$) is a construct

$$\Pi = (O, T, \mu, w_1, \ldots, w_m, R_1, \ldots, R_m),$$

where:

- O is the alphabet of *objects*,
- T is the alphabet of *terminal objects,*
- μ is the *membrane structure* (it is assumed that we have m membranes, labeled with $1, 2, \ldots, m$, the skin membrane usually being labeled with 1),
- w_i, $1 \leq i \leq m$, are strings over O representing the *initial* multiset of *objects* present in the membranes of the system,
- R_i, $1 \leq i \leq m$, are finite sets of *antiport rules* of the form x/y, for some $x, y \in O^*$, associated with membrane i.

An antiport rule of the form $x/y \in R_i$ means moving the objects specified by x from membrane i to the surrounding membrane j (to the environment, if $i = 1$), at the same time moving the objects specified by y in the opposite direction. (The rules with one of x, y being empty are, in fact, *symport rules*, but in the following we do not explicitly consider this distinction, as it is not relevant for what follows.) The weight of an antiport rule x/y is defined as $\max \{|x|, |y|\}$. We assume the environment to contain all objects in an unbounded number.

The computation starts with the multisets specified by $w_1, \ldots, w_m$ in the m membranes; in each time unit, the rules assigned to each membrane are used in a maximally parallel way, i.e., we choose a multiset of rules at each membrane in such a way that, after identifying objects inside and outside the corresponding membranes to be affected by the selected multiset of rules, no objects remain to be subject to any additional rule at any membrane. The computation is successful if and only if it halts; depending on the function of the system, the input and the output may be encoded by different terminal symbols in different membranes, the input then being added in the initial configuration as the corresponding number of respective symbols in the designated membranes.

The set of all natural numbers accepted/generated in this way by the system Π is denoted by $aN(\Pi)$ and $gN(\Pi)$, respectively. The families of sets $aN(\Pi)/gN(\Pi)$ of natural numbers computed as above by systems with at most m membranes, s symbols, and n rules of weight k are denoted by $aNO_sP_m(anti_k)_n$ and $gNO_sP_m(anti_k)_n$, respectively. The family of functions on the set of natural numbers computed as above by antiport P systems with at most m membranes, s symbols, and n rules of weight (at most) k is denoted by $cNO_sP_m(anti_k)_n$.

An (extended) antiport P system with forbidden context is a construct

$$\Pi_f = (O, T, \mu, w_1, \ldots, w_m, R'_1, \ldots, R'_m),$$

where $O, T, \mu, w_1, \ldots, w_m$ are defined as above and R'_i, $1 \leq i \leq m$, are finite sets of *antiport rules with forbidden context* of the form $(x/y, z)$, for some $x, y, z \in O^*$, associated with membrane i. In this case, the application of the antiport rule x/y is only possible, if z is not a sub-multiset of the multiset of objects inside the membrane. (For a generalized variant of this model see [5].)

A computation of Π_f is performed in a similar way as described for antiport systems, but with the difference that now the rules assigned to the membranes are used in a sequential way, i.e., in each time unit only one antiport rule with forbidden context is applied.

We denote the families of sets $aN(\Pi)/gN(\Pi)$ of natural numbers accepted/generated by such systems with at most m membranes, s symbols, n rules of weight at most k, and the forbidden multisets of size at most l by $aNO_sf_lP_m(anti_k)_n$ and $gNO_sf_lP_m(anti_k)_n$, respectively. The family of functions on the set of natural numbers computed as above by antiport P systems with forbidden context with at most m membranes, s symbols, n rules of weight at most k, and the forbidden multisets of size at most l is denoted by $cNO_sf_lP_m(anti_k)_n$.

3 A Universal Register Machine

In [7], several variants of universal register machines were exploited. The main interesting variant for the results presented in this paper is shown in Figure 1.

Register machines usually consist of several registers where natural numbers can be stored and a finite control for storing the (deterministic) program and the current state. The instructions (used in discrete time steps) are simple operations on the registers changing their contents and also the state. The simplest operations on the registers are increment and decrement as well as zero-test. In the diagram of the universal register machine $URM32$ in Figure 1, only these operations are used: the zero-test on register i is indicated by a rhomboid inclosing the encryption Ri, and in the case that the contents of register i is zero, the next operation is the one to be reached with the arc labeled by z; the decrement operation is depicted by a rectangle with the encryption RiP, and the decrement operation by a rectangle with the encryption RiM (as the decrement operation RiM is always preceded by the corresponding zero-test, it can always be carried out). The states are depicted directly at the corresponding operations; q_1 is the initial state, and the state where the $URM32$ stops is indicated by $STOP$ in Figure 1.

The universal register machine $URM32$ uses a very sophisticated number-theoretic encoding of the enumeration of a specific variant of register machines. The code of the register machine to be simulated is put into register 1, the input number into register 2 (where also the output will be computed). The instructions are decoded in the *Instruction reader* part and the *Decoder* part (which essentially performs a division by three), and these instructions work on the registers 0, 2, and 3 (as we know, e.g., see [9], three registers are sufficient to simulate any other register machine). Thus, $URM32$ can compute any partial recursive function $f : \mathbb{N} \rightarrow \mathbb{N}$ (with input and output number in register 2) in the same way as the register machine encoded by the number in register 1 computes f. For the following construction of antiport P systems it is important to note that $URM32$ only stops when having finished the simulation of the register machine encoded in register 1 with the input in register 2, but enters an infinite computation otherwise.

4 A Universal Antiport P System Working in the Maximally Parallel Derivation Mode

Based on the universal register machine $URM32$ described in the previous section, we now construct universal antiport P systems computing any partial recursive function $f : \mathbb{N} \rightarrow \mathbb{N}$ as well as universal antiport P systems accepting/generating arbitrary recursively enumerable sets of natural numbers.

Theorem 1. *$cNO_{34}P_1(anti_6)_{31} = cNRE$.*

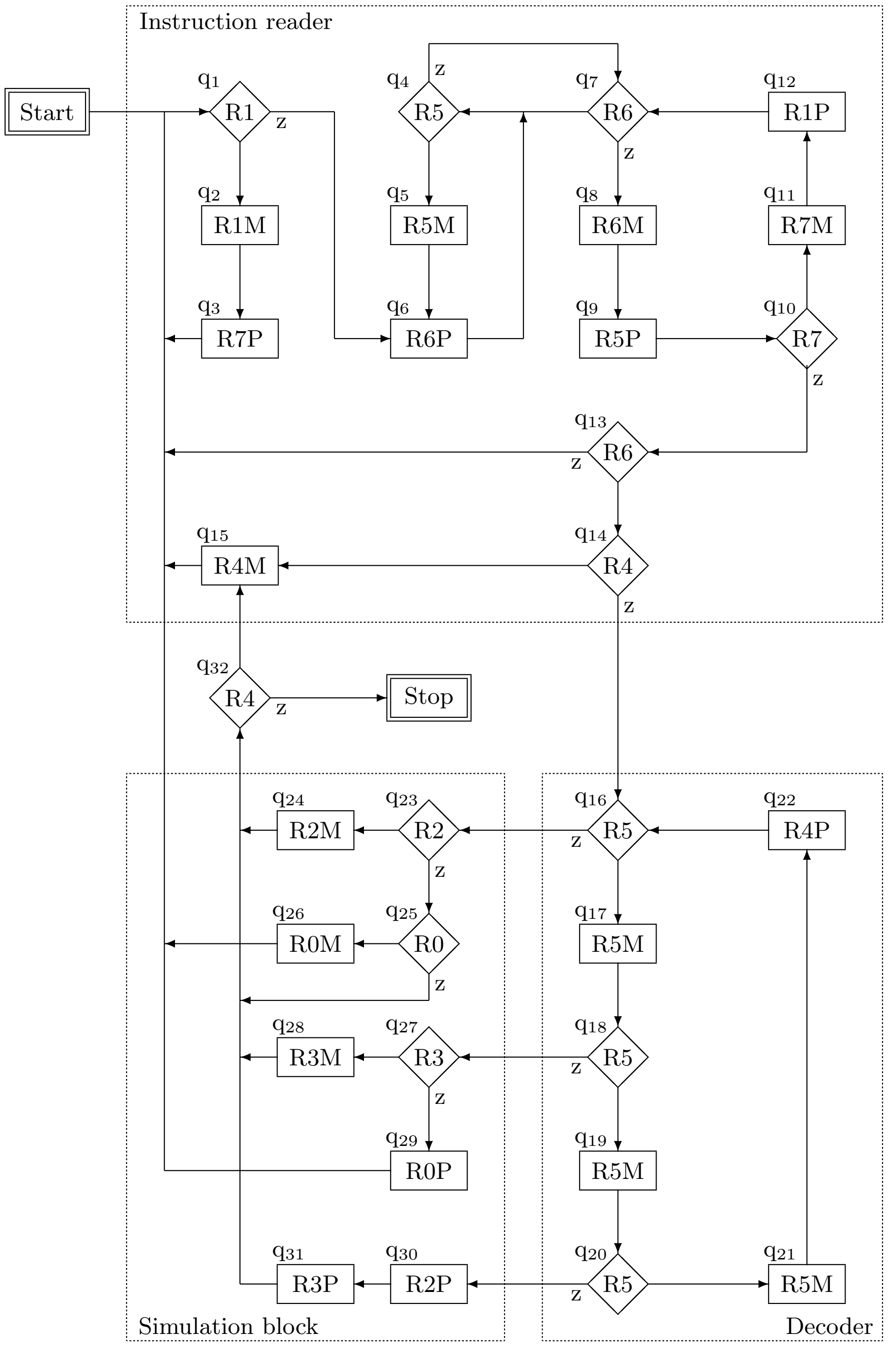

Fig. 1. The universal register machine *URM*32

Proof. We simulate the computation of the partial recursive function $f : \mathbb{N} \to \mathbb{N}$ computed by the register machine with code m on the natural number n by the antiport P system

$$\Pi = (O, \{R_2\}, [_1\ ]_1, q_1 X C_1 R_1^m R_2^n, R)$$

in the following way: The initial multiset $q_1 X C_1 R_1^m R_2^n$ represents the code m by the corresponding number of symbols R_1 and the input number by the corresponding number of symbols R_2; in general, the contents of register i is represented by the corresponding number of symbols R_i. In a halting computation, the result of the computation is represented by the number of terminal symbols R_2. In sum, we use the objects in

$$\begin{aligned} O = {} & \{X, X'\} \cup \{C_i, C_i' \mid 0 \leq i \leq 7\} \cup \{R_i \mid 0 \leq i \leq 7\} \\ & \cup \{q_j \mid j \in \{1, 4, 7, 10, 14, 18, 23, 25, 27, 32\}\} \end{aligned}$$

and the following antiport rules in R:

1. X/X'
2. $C_0 R_0 / C_0'$
3. $C_1 R_1 / C_1'$
4. $C_2 R_2 / C_2'$
5. $C_3 R_3 / C_3'$
6. $C_4 R_4 / C_4'$
7. $C_5 R_5 / C_5'$
8. $C_6 R_6 / C_6'$
9. $C_7 R_7 / C_7'$
10. $q_1 X' C_1 / q_4 R_6 C_5 X$
11. $q_1 X' C_1 / q_1 R_7 X$
12. $q_4 X' C_5' / q_4 R_6 C_5 X$
13. $q_4 X' C_5 / q_7 X C_6$
14. $q_7 X' C_6' / q_{10} R_5 C_7 C_6$
15. $q_7 X' C_6 / q_4 X C_5$
16. $q_{10} X' C_7' C_6' / q_{10} X R_1 R_3 C_7 C_6$
17. $q_{10} X' C_7' C_6' / q_4 X R_1 C_5$
18. $q_{10} X' C_7 C_6 / q_1 X C_1$
19. $q_{10} X' C_7 C_6' / q_{14} X C_4 C_5$
20. $q_{14} X' C_4' C_5' / q_1 X R_5 C_1$
21. $q_{14} X' C_4 C_5' / q_{18} X {C_5}^3$
22. $q_{18} X' {C'_5}^3 / q_{18} X R_4 {C_5}^3$
23. $q_{18} X' {C_5'}^2 C_5 / q_{23} X R_4 C_2$
24. $q_{18} X' C_5' {C_5}^2 / q_{32} X R_2 R_3 C_4$
25. $q_{18} X' {C_5}^3 / q_{27} X C_3$
26. $q_{23} X' C_2' / q_{32} X C_4$
27. $q_{23} X' C_2 / q_{25} X C_0$
28. $q_{25} X' C_0' / q_1 X C_1$
29. $q_{25} X' C_0 / q_{32} X C_4$
30. $q_{27} X' C_3' / q_{32} X C_4$
31. $q_{27} X' C_3 / q_1 X R_0 C_1$
32. $q_{32} X' C_4' / q_1 X C_1$

The symbols X and X' work as a trigger indicating an even and odd number of steps performed so far. In each odd step, a checker symbol C_i tests whether register i is zero or not – in case it is non-zero, a corresponding symbol R_i is eliminated and the checker is turned to its primed version C_i'. These rules are used in the odd steps of a computation in Π together with the antiport rule X/X', whereas the rules involving state symbols q_j can only be used in the even steps together with the symbol X' in rules which bring back the symbol X.

For example, the subprogram of Figure 1 including the states q_1, q_2, and q_3 with the exit to the state q_4 thus is easily simulated by the two antiport rules

10. $q_1 X' C_1 / q_4 R_6 C_5 X$
11. $q_1 X' C_1 / q_1 R_7 X$

together with the antiport rules

$$\begin{array}{l}1.\ X/X' \\ 3.\ C_1R_1/C_1'\end{array}$$

using the maximally parallel derivation mode. In a similar way, the other instructions of $URM32$ are simulated by the antiport rules in Π using the maximally parallel derivation mode.

Sometimes, an analysis of the function of specific parts of the diagram in Figure 1 allows for even shorter simulations, e.g., consider the loop from q_{10} to q_{10} via q_7 simulated by

$$16.\ q_{10}X'C_7'C_6'/q_{10}XR_1R_3C_7C_6$$

and its "exits" via the rules 17 to 19. Moreover, as the division by three is the main function of the "Decoder part", in the lower part some shortcuts are possible, but we do not go into more details here. Only one more observation has to be explained, because so far we have 32 and not only 31 rules: As register 2 and register 3 both are only used once – in the "Simulation part" – the states q_{23} and q_{27} can be taken to be identical because the two symbols C_2 and C_3 and their primed versions, respectively, identify the correct state; on the other hand, as both decrement operations from the states q_{23} and q_{27} lead to the same instruction, we can even identify the symbols C_2' and C_3'; thus, we get the modified antiport rule $5'.\ C_3R_3/C_2'$ as well as

$$\begin{array}{l}25'.\ q_{18}X'{C_5}^3/q_{23}XC_3 \\ 30'.\ q_{23}X'C_2'/q_{32}XC_4 \\ 31'.\ q_{23}X'C_3/q_1XR_0C_1\end{array}$$

As the rules $30'$ and 26 now are identical, finally we only have 31 rules in R and only 34 objects in O. These observations complete the proof. □

In the literature, the result of a halting computation often consists of the number of all symbols in the designated output membrane without specifying the set of terminal symbols in the antiport P system. A thorough analysis of the universal register machine $URM32$ shows that when it halts not only register 2 as the output register, but also register 6 and register 1 (still containing the code of the register machine to be simulated) may be non-empty. (Due to the features of the 3-register machine as constructed in [9] and simulated by $URM32$ in registers 0, 2, and 3, at the end of a computation registers 0 and 3 are empty; observe that the emptiness of these registers cannot be inferred from the program of $URM32$.) Therefore, to eliminate all non-terminal symbols, we would have to add the following rules (using the additional symbols q_{33} and q_{34}):

$$\begin{array}{l}32.\ q_{32}X'C_4/q_{33}XC_6 \\ 33.\ q_{33}X'C_6'/q_{33}XC_6 \\ 34.\ q_{33}X'C_6/q_{34}XC_1 \\ 35.\ q_{34}X'C_1'/q_{34}XC_1 \\ 36.\ q_{34}X'C_1/\lambda\end{array}$$

Corollary 1. $aNO_{34}P_1(anti_6)_{31} = NRE$.

Proof. The construction of the preceding theorem already yields the desired result, as, given $L \in NRE$, we just have to simulate a register machine computing the partial recursive function $f : \mathbb{N} \to \mathbb{N}$ with $f(n) = 0$ for any $n \in L$ and $f(n)$ being undefined otherwise. □

Corollary 2. $gNO_{35}P_1(anti_6)_{33} = NRE$.

Proof. We add the following two rules (using the additional symbol q_0) for generating an arbitrary number n in register 2:

$$0.\ (q_0/R_2q_0, \lambda)$$
$$0'.\ (q_0/q_1, \lambda)$$

Now we start with $q_0XC_1R_1^m$, apply rule 0 for n times and once rule $0'$, and then we just simulate the acceptance of the number n in register 2 as in the preceding corollary. □

5 A Small Universal Antiport P System with Forbidden Context Working in the Sequential Derivation Mode

Based on the universal register machine $URM32$, in this section we construct universal antiport P systems with forbidden context computing any partial recursive function $f : \mathbb{N} \to \mathbb{N}$ as well as universal antiport P systems with forbidden context accepting/generating arbitrary recursively enumerable sets of natural numbers.

Theorem 2. $cNO_{12}f_5P_1(anti_4)_{16} = cNRE$.

Proof. We simulate the computation of the partial recursive function $f : \mathbb{N} \to \mathbb{N}$ computed by the register machine with code m on the natural number n by the antiport P system with forbidden context

$$\Pi = (O, \{R_2\}, [_1\ ]_1, q_1R_1^mR_2^n, R),$$

in the following way: The initial multiset $q_1R_1^mR_2^n$ represents the code m by the corresponding number of symbols R_1 and the input number by the corresponding number of symbols R_2; in general, the contents of register i is represented by the corresponding number of symbols R_i. In a halting computation, the result of the computation is represented by the number of terminal symbols R_2. In sum, we use the objects in

$$O = \{R_i \mid 0 \leq i \leq 7\} \cup \{q_j \mid j \in \{1, 4, 10, 18\}\}$$

and the following antiport rules in R:

1. $(q_1R_1/q_1R_7, \lambda)$
2. $(q_1/q_4R_6, R_1)$
3. $(q_4R_5/q_4R_6, \lambda)$
4. $(q_4R_6/q_{10}R_5, R_5)$
5. $(q_{10}R_7R_6/q_{10}R_1R_5, \lambda)$
6. $(q_{10}R_7/q_4R_1, R_6)$
7. $(q_{10}/q_1, R_6R_7)$
8. $(q_{10}R_6R_4/q_1, R_7)$
9. $(q_{10}R_6R_5/q_{18}, R_7R_4)$
10. $\left(q_{18}{R_5}^3/R_4q_{18}, \lambda\right)$
11. $(q_{18}/R_0q_1, R_5R_3)$
12. $\left(q_{18}{R_5}^2R_0/q_1, {R_5}^3R_2\right)$
13. $\left(q_{18}{R_5}^2R_2/q_1, {R_5}^3\right)$
14. $\left(q_{18}{R_5}^2/q_1, {R_5}^3R_2R_0\right)$
15. $(q_{18}R_3R_4/q_1, R_5)$
16. $\left(q_{18}R_5R_4/q_1R_2R_3, {R_5}^2\right)$

As we can check for the occurrence of a multiset and not only of one symbol, we now can do several zero tests with one rule; especially the “Simulation block” can be “checked through” directly from state q_{18}; observe that the increment operation at state q_{22} and the decrement operation at state q_{15} neutralize each other, hence, in the rules 12 to 14 we can omit R_4 on both sides of the antiport rule. Halting computations end up with a multiset containing q_{18} with none of the rules 10 to 16 being applicable anymore. □

Corollary 3. $aNO_{12}f_5P_1(anti_4)_{16} = NRE$.

Proof. Given $L \in NRE$, we just simulate – according to the construction given in the preceding theorem, a register machine computing the partial recursive function $f : \mathbb{N} \to \mathbb{N}$ with $f(n) = 0$ for any $n \in L$ and $f(n)$ being undefined otherwise. □

Corollary 4. $gNO_{13}f_5P_1(anti_4)_{18} = NRE$.

Proof. We add the following two rules (using the additional symbol q_0) for generating an arbitrary number n in register 2:

0. $(q_0/R_2q_0, \lambda)$

0′. $(q_0/q_1, \lambda)$

Now we start with $q_0R_1^m$, apply rule 0 for n times and once rule 0′, and then we just simulate the acceptance of the number n in register 2 as in the preceding corollary. □

5.1 Tuning the complexity parameters

As is well known, e.g., see [6], antiport rules of weight 2 in one membrane are sufficient for generating any recursively enumerable set of natural numbers. Hence, in the following theorem, we bound the weights of the antiport rules by two, but also allow the forbidden context multiset to check for the non-occurrence of a single object only.

Theorem 3. $cNO_{22}f_1P_1(anti_2)_{26} = cNRE$.

Proof. Again, we simulate the universal register machine $URM32$; due to the restrictions on the weights of the rules and on the size of the forbidden contexts, more rules than in the proof of Theorem 2 are needed, because we have to introduce some intermediate states.

In that way we obtain the antiport P system with forbidden context

$$\Pi = (O', \{R_2\}, [_1\ ]_1, q_1 R_1^m R_2^n, R')$$

with the objects in

$$\begin{aligned} O' = &\{R_i \mid 0 \leq i \leq 7\} \\ &\cup \{q_j \mid j \in \{1,4,7,10,13,14,16,18,20,23,25,27,30,32\}\} \end{aligned}$$

and the following antiport rules in R':

1. $(q_1R_1/q_1R_7, \lambda)$
2. $(q_1/q_4R_6, R_1)$
3. $(q_4R_5/q_4R_6, \lambda)$
4. $(q_4/q_7, R_5)$
5. $(q_7R_6/q_{10}R_5, \lambda)$
6. $(q_7/q_4, R_6)$
7. $(q_{10}R_7/q_7R_1, \lambda)$
8. $(q_{10}/q_{13}, R_7)$
9. $(q_{13}R_6/q_{14}R_6, \lambda)$
10. $(q_{13}/q_1, R_6)$
11. $(q_{14}R_4/q_1, \lambda)$
12. $(q_{14}/q_{16}, R_4)$
13. $(q_{16}R_5/q_{18}, \lambda)$
14. $(q_{16}/q_{23}, R_5)$
15. $(q_{18}R_5/q_{20}, \lambda)$
16. $(q_{18}/q_{27}, R_5)$
17. $(q_{20}R_5/q_{16}R_4, \lambda)$
18. $(q_{20}/q_{30}R_2, R_5)$
19. $(q_{23}R_2/q_{32}, \lambda)$
20. $(q_{23}/q_{25}, R_2)$
21. $(q_{25}R_0/q_1, \lambda)$
22. $(q_{25}/q_{32}, R_0)$
23. $(q_{27}R_3/q_{32}, \lambda)$
24. $(q_{27}/q_1R_0, R_3)$
25. $(q_{30}/q_{32}R_3, \lambda)$
26. $(q_{32}R_4/q_1, \lambda)$

As the forbidden context multisets contain only one symbol, in this case we only check for the appearance of this symbol in the underlying multiset which in fact is a less powerful control mechanism than checking for the non-appearance of a finite multiset. □

As in the previous cases, we immediately get the following corollaries:

Corollary 5. $aNO_{22}f_1P_1(anti_2)_{26} = aNRE.$

Corollary 6. $gNO_{23}f_1P_1(anti_2)_{28} = NRE.$

6 Universal Multiset Grammars

The results obtained for antiport P systems with forbidden context allow us to derive similar results for forbidden random context multiset grammars with arbitrary multiset productions, because the antiport rule with forbidden context $(x/y, z)$ can be interpreted as the multiset production with forbidden random context $(x \rightarrow y, z)$.

Theorem 4. $cNO_{14}fRC_5(arb_4)_{21} = cNRE.$

Proof. We can interpret the antiport P system with forbidden context

$$\Pi = (O, \{R_2\}, [_1\ ]_1, q_1R_1^mR_2^n, R)$$

as the forbidden context multiset grammar with arbitrary multiset productions

$$G = (O, \{R_2\}, q_1R_1^mR_2^n, R')$$

with $R' = \{(x \rightarrow y, z) \mid (x/y, z) \in R\}$. As now we have to derive terminal multisets, we in any case have to eliminate the non-terminal symbols occurring in the final multisets of halting computations in Π. As already mentioned above, a thorough analysis of the universal register machine $URM32$ shows that we have to introduce rules for eliminating an arbitrary number of symbols R_1 and R_6 as well as the state symbol q_{18}. Yet, instead of just adding rules accomplishing these tasks, we only take the rules 1 to 14 from R into R' and then add the following rules instead of the last two rules having the state symbol q_{18} and R_4 on the left-hand side of the multiset production:

15. $(q_{18}R_3 \rightarrow q_{32}, R_5)$
16. $\left(q_{18}R_5 \rightarrow q_{32}R_2R_3, R_5{}^2\right)$
17. $(q_{32}R_4 \rightarrow q_1, \lambda)$
18. $(q_{32} \rightarrow q_{33}, R_4)$
19. $(q_{33}R_6 \rightarrow q_{33}, \lambda)$
20. $(q_{33}R_1 \rightarrow q_{33}, \lambda)$
21. $(q_{33} \rightarrow \lambda, R_1R_6)$

Hence, in total, we have 21 rules and the set of objects

$$O'' = \{R_i \mid 0 \leq i \leq 7\} \cup \{q_j \mid j \in \{1, 4, 10, 18, 32, 33\}\}.$$

In sum, we have obtained the forbidden random context multiset grammar

$$G'' = (O'', \{R_2\}, q_1 R_1^m R_2^n, R'')$$

with R'' containing the 21 rules as constructed above and therefore fulfilling all the desired constraints. □

If we were not allowing initial multisets but only a start symbol from $V - T$ in the multiset grammar, then we would have to add the start symbol q_0 and the start rule

$$0'. (q_0 \to R_1^m q_1, \lambda)$$

which would not allow us to bound the size of the multiset productions.

The following corollary immediately follows from the constructions given in the proof of the preceding theorem.

Corollary 7. *$aNO_{14}fRC_5(arb_4)_{21} = gNO_{15}fRC_5(arb_4)_{23} = NRE$.*

7 Conclusion

The results obtained in this paper concerning the complexity of antiport P systems and antiport P systems with forbidden context as well as forbidden random context grammars with arbitrary multiset productions are summarized in the following tables:

$$\begin{aligned} cNRE &= cNO_{34}P_1(anti_6)_{31} \\ &= cNO_{12}f_5P_1(anti_4)_{16} \\ &= cNO_{22}f_1P_1(anti_2)_{26} \\ &= cNO_{14}fRC_5(arb_4)_{21} \end{aligned}$$

The table for the complexity of the systems computing partial recursive functions immediately yields the table for the complexity of the accepting systems with exactly the same size of the parameters:

$$\begin{aligned} NRE &= aNO_{34}P_1(anti_6)_{31} \\ &= aNO_{12}f_5P_1(anti_4)_{16} \\ &= aNO_{22}f_1P_1(anti_2)_{26} \\ &= aNO_{14}fRC_5(arb_4)_{21} \end{aligned}$$

For the generating systems, we need some additional symbols and rules for generating a representation of the initial number to be analyzed:

$$\begin{aligned} NRE &= gNO_{35}P_1(anti_6)_{33} \\ &= gNO_{13}f_5P_1(anti_4)_{18} \\ &= gNO_{23}f_1P_1(anti_2)_{28} \\ &= gNO_{15}fRC_5(arb_4)_{23} \end{aligned}$$

Improvements especially with respect to the number of rules, but also with respect to the number of objects as well as with respect to the complexity of the antiport rules themselves remain for future research as well as the challenge to find a variant of P systems allowing for a universal system with less than 16/16/18 rules for computing any partial recursive function on the set of natural numbers/accepting any arbitrary recursively enumerable set of natural numbers/generating any arbitrary recursively enumerable set of natural numbers.

Acknowledgement

The work of Marion Oswald is supported by FWF-project T225-N04. The first author gratefully acknowledges the interesting discussions with Sergey Verlan and Artiom Alhazov during the Brainstorming Week providing the main ideas which the results of Section 4 are based on.

References

1. C.S. Calude, Gh. Păun, G. Rozenberg, A. Salomaa, eds.: *Multiset Processing – Mathematical, Computer Science and Molecular Computing Points of View.* LNCS 2235, Springer, Berlin, 2001.
2. C.S. Calude, Gh. Păun: *Computing with Cells and Atoms.* Taylor & Francis, London, 2001.
3. E. Csuhaj-Varjú, C. Martín-Vide, V. Mitrana: Multiset automata. In [1], 69–84.
4. J. Dassow, Gh. Păun: *Regulated Rewriting in Formal Language Theory.* Springer, Berlin, 1989.
5. R. Freund, M. Oswald: GP systems with forbidding context. *Fundamenta Informaticae*, 49, 1–3 (2002), 81–102.
6. R. Freund, M. Oswald: P Systems with activated/prohibited membrane channels. In [12], 261–268.
7. I. Korec: Small universal register machines. *Theoretical Computer Science*, 168, (1996), 267–301.
8. M. Kudlek, C. Martín-Vide, Gh. Păun: Toward a formal macroset theory. In [1], 123–134.
9. M.L. Minsky: *Computation – Finite and Infinite Machines.* Prentice Hall, Englewood Cliffs, NJ, 1967.
10. A. Păun, Gh. Păun: The power of communication: P systems with symport/antiport. *New Generation Computing*, 20, 3 (2002), 295–306.
11. Gh. Păun: *Membrane Computing: An Introduction.* Springer, Berlin, 2002.
12. Gh. Păun, G. Rozenberg, A. Salomaa, C. Zandron, eds.: *Membrane Computing. International Workshop, WMC-CdeA 2002*, Curtea de Argeş, Romania, August 2002. LNCS 2597, Springer, Berlin, 2003.
13. G. Rozenberg, A. Salomaa, eds.: *Handbook of Formal Languages.* Springer, Berlin, 1997.