Multiset Random Context Grammars, Checkers, and Transducers

Matteo Cavaliere[1,3], Rudolf Freund[2], Marion Oswald[2],
Dragoş Sburlan[3,4]

[1] Microsoft Research, University of Trento
Centre for Computational and Systems Biology
Trento, Italy
matteo.cavaliere@msr-unitn.unitn.it
[2] Faculty of Informatics
Vienna University of Technology
Favoritenstr. 9–11, A–1040 Vienna, Austria
{rudi,marion}@emcc.at
[3] Department of Computer Science and Artificial Intelligence
University of Sevilla,
Av. Reina Mercedes, 41012, Sevilla, Spain
[4] Department of Informatics and Numerical Methods
Ovidius University of Constantza,
124 Mamaia Bd., Constantza, Romania
dsburlan@univ-ovidius.ro

Summary. We introduce a general model of random context multiset grammars as well as the concept of multiset random context checkers and transducers. Our main results show how recursively enumerable sets of finite multisets can be generated using these models of computing; corresponding results for antiport P systems are established, too.

1 Introduction

The basic concepts of regulated rewriting can be found in [5], and especially the concept of random context grammars is investigated there in detail for the string case. In the emergent field of P systems we mostly deal with multisets, hence several generating and accepting devices, and among them various models of P systems, were investigated in several articles in [1]. Various interesting models of grammars for generating multisets were considered in [13], multiset automata were investigated in [3]. For the basic variants of P systems (introduced as membrane systems in [15]) investigated so far we refer the reader to [16] for a comprehensive overview as well as to [19] for the actual state of research. We assume the reader to be familiar with the original definitions and explanations given there for these models.

In this paper, we introduce a general model of random context grammars of arbitrary type based on a partial order relation for the objects the grammar is working on as well as the concept of random context checkers and random context transducers; we show how recursively enumerable sets of finite multisets can be generated using these models of computing. In the string case, we obtain the classic variant of random context grammars when using the subword relation as the partial ordering; for multiset grammars, the order relation is taken to be the multiset inclusion. As a natural extension of multiset grammars we consider antiport P systems and show how several results for multiset grammars with arbitrary rules directly carry over to antiport P systems working in the sequential mode in only one membrane. Finally, we discuss some open problems for future research.

2 Definitions

In this section we first recall some well-known definitions from formal language theory; then we define our general model for random context grammars. Moreover, we recall some notions for string grammars and languages in the general setting of this paper and finally we give a short definition of antiport P systems working in the sequential mode.

2.1 Preliminaries

The set of integers is denoted by $\mathbb{Z}$, the set of non-negative integers by $\mathbb{N}$. An *alphabet* V is a finite non-empty set of abstract *symbols*. Given V, the free monoid generated by V under the operation of concatenation is denoted by V^*; the elements of V^* are called strings, and the *empty string* is denoted by λ; $V^* - \{\lambda\}$ is denoted by V^+. Let $\{a_1, \dots, a_n\}$ be an arbitrary alphabet; the number of occurrences of a symbol a_i in x is denoted by $|x|_{a_i}$; the *Parikh vector* associated with x with respect to $a_1, \dots, a_n$ is $\left(|x|_{a_1}, \dots, |x|_{a_n}\right)$. The *Parikh image* of a language L over $\{a_1, \dots, a_n\}$ is the set of all Parikh vectors of strings in L. For a family of languages FL, the family of Parikh images of languages in FL is denoted by $PsFL$.

A (finite) multiset over the (finite) alphabet V, $V = \{a_1, \dots, a_n\}$, is a mapping $f : V \longrightarrow \mathbb{N}$ and represented as $\langle f(a_1), a_1\rangle \dots \langle f(a_n), a_n\rangle$ or as any string x the Parikh vector of which with respect to $a_1, \dots, a_n$ is $(f(a_1), \dots, f(a_n))$. In the following we will not distinguish between a vector $(m_1, \dots, m_n)$, its representation by a multiset $\langle m_1, a_1\rangle \dots \langle m_n, a_n\rangle$ or its representation by a string x with Parikh vector $\left(|x|_{a_1}, \dots, |x|_{a_n}\right) = (m_1, \dots, m_n)$. Fixing the sequence of symbols $a_1, \dots, a_n$ in the alphabet V in advance, the representation of the multiset $\langle m_1, a_1\rangle \dots \langle m_n, a_n\rangle$ as a string $a_1^{m_1} \dots a_n^{m_n}$ is unique. The set of all finite multisets over an alphabet V is denoted by V^o.

For more details of formal language theory we refer to [5] and [17].

2.2 Grammar Schemes and Grammars

In the following, we shall deal with various types of objects and grammars, hence, we first define a general model of a grammar scheme:

A *grammar scheme* G is a construct $(O, O_T, P, \Longrightarrow_G)$, where:

- O is the set of *objects*;
- $O_T \subseteq O$ is the set of *terminal* objects;
- P is a finite set of *productions*;
- $\Longrightarrow_G \subseteq O \times O$ is the *derivation relation* of G induced by the productions in P.

The derivation relation $\Longrightarrow_G$ is obtained as the union of all $\Longrightarrow_p \subseteq O \times O$, i.e., $\Longrightarrow_G := \cup_{p\in P} \Longrightarrow_p$, where each $\Longrightarrow_p$ is a relation which we assume at least to be recursive. The reflexive and transitive closure of $\Longrightarrow_G$ is denoted by $\overset{*}{\Longrightarrow}_G$.

In the following we shall consider different types of grammar schemes depending on the components of G, especially with respect to different types of productions.

Based on a grammar scheme of a specific type (in the following referred to as grammar scheme of type X), we now define the notion of a (sequential) grammar.

Let $G = (O, O_T, P, \Longrightarrow_G)$ be a grammar scheme of type X. Then the pair (G, w) with $w \in O$ is called a *grammar of type* X, w is the *axiom (start object).*

The *language generated by* (G, w) is the set of all terminal objects (we also assume $v \in O_T$ to be decidable for every $v \in O$) derivable from the axiom, i.e.,

$$L(G, w) = \left\{ v \in O_T \mid w \overset{*}{\Longrightarrow}_G v \right\}.$$

The family of languages generated by grammars of type X is denoted by $\mathcal{L}(X)$.

In many cases, the type X of the grammar scheme allows for the following feature:

A type X of a grammar scheme is called a *type with unit rules* if for every grammar scheme $G = (O, O_T, P, \Longrightarrow_G)$ of type X there exists a grammar scheme $G' = (O, O_T, P \cup P^+, \Longrightarrow_{G'})$ of type X such that:

- $P^+ = \{p^+ \mid p \in P\}$,
- for all $x \in O$, p is applicable to x if and only if p^+ is applicable to x, and
- for all $x \in O$, the application of p^+ to x – in case p^+ is applicable to x – yields x back again.

2.3 General Concepts for Random Context Grammars

In [11], the following general notion of a random context-grammar (scheme) was introduced based on the applicability of rules; we here present this model in a slightly different way in order to make it easier to be compared with the new model introduced afterwards.

A *random context grammar scheme* (based on the applicability of productions) G_{RC_P} of type X is a construct

$$\left(G, P', \Longrightarrow_{G_{RC_P}}\right),$$

where:

- $G = (O, O_T, P, \Longrightarrow_G)$ is a grammar scheme of type X;
- P' is a set of rules of the form (q, R, Q) where $q \in P$, $R \cup Q \subseteq P$;
- $\Longrightarrow_{G_{RC_P}}$ is the derivation relation assigned to G_{RC} such that for any $x, y \in O$, $x \Longrightarrow_{G_{RC_P}} y$ if and only if for some rule $(q, R, Q) \in P'$, $x \Longrightarrow_G y$ by q and, moreover, all productions from R are applicable to x as well as no production from Q is applicable to x.

For short, we shall speak of an X-$RC_P\,(m,n)$ grammar scheme if for any $(q, R, Q) \in P'$ the set R contains at most m rules and the set Q contains at most n rules.

Based on a partial order relation on the objects in O, we now are able to define a different new notion for a random context grammar (scheme).

A *random context grammar scheme* G_{RC_O} of type X (based on an order relation on the objects) is a construct

$$\left(G, \sqsubseteq, P', \Longrightarrow_{G_{RC_O}}\right),$$

where:

- $G = (O, O_T, P, \Longrightarrow_G)$ is a grammar scheme of type X;
- $(O, \sqsubseteq)$ is a partially ordered set;
- P' is a set of rules of the form (q, R, Q) where $q \in P$, $R \cup Q \subseteq O$;
- $\Longrightarrow_{G_{RC_O}}$ is the derivation relation assigned to G_{RC} such that for any $x, y \in O$, $x \Longrightarrow_{G_{RC_O}} y$ if and only if for some rule $(q, R, Q) \in P'$, $x \Longrightarrow_G y$ by q and, moreover, $r \sqsubseteq x$ holds true for all $r \in R$ as well as $s \sqsubseteq x$ does not hold true for any $s \in Q$.

For short, we shall speak of an X-$RC_O\,(m,n)$ grammar scheme if for any $(q, R, Q) \in P'$ the set R contains at most m objects and the set Q contains at most n objects.

Now let $(O, \sqsubseteq)$ be a partially ordered set such that O contains a unique minimal element e with $e \sqsubseteq x$ for all $x \in O$. Moreover, we call an object $x \in O$ k-minimal if and only if any maximal chain between e and x contains at most $k+1$ objects from O (observe that e is the only object which is 0-minimal). Then we shall call G_{RC} an X-$RC_O\,(m/k, n/l)$ grammar scheme if and only if it is an X-$RC_O\,(m,n)$ grammar scheme and, moreover, for any $(q, R, Q) \in P'$, all objects in the set R are k-minimal and all objects in the set Q are l-minimal.

In both cases, if any of the parameters m, n (and k, l) is not to be specified, we replace it by $*$. Again in both variants, a *random context grammar* is a pair $(G_{RC_P}, w) / (G_{RC_O}, w)$ where $w \in O$ is the *axiom*. A random context grammar (scheme) is called a *grammar (scheme) with permitting context* if $Q = \emptyset$ for every $(q, R, Q) \in P'$ (i.e., $n = 0$) and a *grammar (scheme) with forbidden context* if $R = \emptyset$ for every $(q, R, Q) \in P'$ (i.e., $m = 0$). The families of languages generated by random context grammars of the types Y as defined above are denoted by $\mathcal{L}(Y)$.

2.4 String grammars

A *string grammar scheme* (i.e., a grammar scheme of type “string”) usually is defined as a construct (N, T, P) where N is the alphabet of *non-terminal symbols*, T is the set of *terminal* symbols, $N \cap T = \emptyset$, P is a finite set of *productions* of the form $u \to v$ with $u \in V^+$, and $v \in V^*$, where $V := N \cup T$.

In the general notion as defined above, a string grammar scheme G would be represented as

$$\left((V \cup T)^*, T^*, P, \Longrightarrow_G\right),$$

where the derivation relation for $u \to v \in P$ is defined as usual by $xuy \Longrightarrow_{u \to v} xvy$ for all $x, y \in V^*$, thus yielding the well-known derivation relation $\Longrightarrow_G$ for the string grammar scheme G. A *string grammar* then is a pair (G, S) where $S \in V - T$ is the start symbol.

As special types of string grammars we consider string grammars with arbitrary productions, context-free productions of the form $A \to v$ with $A \in N$ and $v \in V^*$, λ-free context-free productions of the form $A \to v$ with $A \in N$ and $v \in V^+$, and (right-)regular productions of the form $A \to v$ with $A \in N$ and $v \in TN \cup T$; the corresponding types of grammars are denoted by RE, CF, $CF_{-\lambda}$, and REG, thus yielding the families of languages $\mathcal{L}(RE)$, i.e., the family of recursively enumerable languages, as well as $\mathcal{L}(CF)$, $\mathcal{L}(CF_{-\lambda})$, and $\mathcal{L}(REG)$, i.e., the families of context-free, λ-free context-free, and regular languages, respectively. Observe that the types RE, CF, and $CF_{-\lambda}$ are types with unit rules (of the form $w \to w$ for $w \to v \in P$), whereas the type REG (in the definition given above) is not a type with unit rules (therefore, we often allow regular productions to be of the general form $A \to v$ with $A \in N$ and $v \in T^*V \cup T^*$).

A *matrix grammar* (with appearance checking) is a construct

$$G = (N, T, P, M, F, S),$$

where (N, T, P) is a string grammar scheme of type X, $S \in N$ is the start symbol, M is a finite set of sequences of the form $(x_1 \to y_1, \ldots, x_n \to y_n)$, $n \geq 1$, of productions of type X over $N \cup T$ in P (with $x_i \in (N \cup T)^+$, $y_i \in (N \cup T)^*$, in all cases), and $F \subseteq P$.

For $w, z \in (N \cup T)^*$ we write $w \Longrightarrow z$ if there are a matrix $(x_1 \to y_1, \ldots, x_n \to y_n)$ in M and strings $w_i \in (N \cup T)^*$, $1 \leq i \leq n+1$, such that $w = w_1$, $z = w_{n+1}$, and, for all $1 \leq i \leq n$, either

(1) $w_i = w_i' x_i w_i'', w_{i+1} = w_i' y_i w_i''$, for some $w_i', w_i'' \in (N \cup T)^*$, or
(2) $w_i = w_{i+1}$, $x_i \to y_i$ is not applicable to w_i, and $x_i \to y_i \in F$.

The language generated by G is defined by $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$, where $\Longrightarrow^*$ is the reflexive and transitive closure of the relation $\Longrightarrow$. The family of languages of this form is denoted by $\mathcal{L}\,(X\text{-}MAT_{ac})$. If the set F is empty, then the grammar is said to be without appearance checking; the corresponding family of languages is denoted by $\mathcal{L}\,(X\text{-}MAT)$. As is well-known, $\mathcal{L}\,(X\text{-}MAT) \subsetneqq \mathcal{L}\,(CF\text{-}MAT_{ac}) = \mathcal{L}\,(RE)$.

In the proofs given in the next section we will use a well-known normal form for matrix grammars of type CF (context-free matrix grammar): a matrix grammar $G = (N, T, P, M, F, S)$ is said to be in *binary normal form* if $N = N_1 \cup N_2 \cup \{S, \#\}$, with these three sets being mutually disjoint, and the matrices in M in one of the following forms:

1. $(S \to XA)$, with $X \in N_1$, $A \in N_2$,
2. $(X \to Y, A \to x)$, with $X, Y \in N_1$, $A \in N_2$, $x \in (N_2 \cup T)^*$, $|x| \leq 2$,
3. $(X \to Y, A \to \#)$, with $X, Y \in N_1$, $A \in N_2$,
4. $(X \to \lambda, A \to x)$, with $X \in N_1$, $A \in N_2$, and $x \in T^*$, $|x| \leq 2$.

Moreover, there is only one matrix of type 1 (that is why we usually write it in the form $(S \to X_0 A_0)$, in order to fix the symbols X, A present in it), and F consists exactly of all rules $A \to \#$ appearing in matrices of type 3; $\#$ is a trap-symbol, because once introduced, it is never removed. A matrix of type 4 is used only once, in the last step of a derivation.

For each context-free matrix grammar (with or without appearance checking) there is an equivalent matrix grammar in the binary normal form (e.g., see [5]). In fact, in the next section we shall use a slightly modified version of this binary normal form called *f-binary normal form* in [9], i.e., instead of the final matrices of type 4 there is only one single final matrix of the form $(f \to \lambda)$, for some special symbol $f \in N_1$.

2.5 Multiset Grammars

For an introduction to multiset grammars, we refer the reader to [13]. In our general notation, a *multiset grammar scheme* G_m is of the form $(O, O_T, P, \Longrightarrow_m)$, where:

- O is the set of all finite multisets over a finite alphabet V, denoted by V^o,
- O_T is the set of all finite multisets over an alphabet $T \subseteq V$, denoted by T^o, and
- P is a set of multiset productions yielding a derivation relation $\Longrightarrow_m$ on the 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, which yields the well-known derivation relation $\Longrightarrow_m$.

A *multiset grammar* then is a pair (G_m, w) where $w \in V^o$ is the axiom (the initial multiset). We consider all derivations starting from the multiset w 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 $L(G_m)$.

As special types of multiset grammars we consider multiset grammars with arbitrary productions, context-free productions of the form $A \to v$ with $A \in V$ and $v \in V^o$, and regular productions of the form $A \to v$ with $A \in N$ and $v \in T^*V \cup T^*$; the corresponding types X of multiset grammars are denoted by $mARB$, mCF, and $mREG$, thus yielding the families of languages $\mathcal{L}(X)$. As it was shown in [13],

$$\begin{aligned} Ps(\mathcal{L}(REG)) &= \mathcal{L}(mREG) = \mathcal{L}(mCF) = Ps(\mathcal{L}(CF)) \\ &\subsetneqq \mathcal{L}(mARB) \\ &\subsetneqq Ps(\mathcal{L}(RE)) = Ps(\mathcal{L}(MAT_{ac})), \end{aligned}$$

i.e., even with arbitrary multiset productions we need some control mechanism to get $Ps(\mathcal{L}(RE))$.

A random context multiset grammar G_{RC} with respect to the new model introduced in this paper now is a construct of the form

$$(G_m, \sqsubseteq_m, P', \Longrightarrow_{G_{RC}}),$$

where:

- $G_m = (V^o, T^o, P, \Longrightarrow_m)$ is a multiset grammar scheme;
- $(O, \sqsubseteq_m)$ is a partially ordered set with $\sqsubseteq_m$ being the multiset inclusion (observe that the unique minimal element for the multiset order relation is the empty multiset);
- P' is a set of rules of the form (q, R, Q) such that $q \in P$ is a multiset processing rule on multisets over V and R, Q are sets of multisets over V;
- $\Longrightarrow_{G_{RC}}$ is the derivation relation assigned to G_{RC} such that for any $x, y \in O$, $x \Longrightarrow_{G_{RC}} y$ if and only if for some rule $(q, R, Q) \in P'$, $x \Longrightarrow_m y$ by the multiset production q and, moreover, all $r \in R$ are sub-multisets of x as well as no $s \in Q$ is a sub-multiset of x.

Now let us consider a very well-known example: Let

$$L = \left\{ a^{2^n} \mid n \geq 0 \right\}.$$

Consider $V = \{A, B, S, X, Y, Z\}$, $T = \{a\}$, and the set of context-free (multiset) rules (i.e., of type mCF)

$$\begin{aligned} P = \{ & S \to XA, X \to Z, A \to a, Z \to \lambda, \\ & A \to BB, X \to Y, B \to a, Y \to X \} \end{aligned}$$

as well as the set of random context multiset rules

$$\begin{aligned} P' = \{ & (S \to XA, \emptyset, \emptyset), (X \to Z, \emptyset, \{B\}), \\ & \{A \to a, \emptyset, \{X, Y\}\}, (Z \to \lambda, \emptyset, \{A\}), \\ & (A \to BB, \emptyset, \{Y, Z\}), (X \to Y, \emptyset, \{A\}), \\ & (B \to a, \emptyset, \{X, Z\}), (Y \to X, \emptyset, \{B\})\}. \end{aligned}$$

Then obviously the $mCF\text{-}RC_O(0, 2/1)$ grammar

$$G_{RC} = ((V^o, \{a\}^o, P, \Longrightarrow_m), \sqsubseteq_m, P', \Longrightarrow_{G_{RC}})$$

yields $Ps(L)$. In order to get an $mCF\text{-}RC_O(1/1, 1/1)$ grammar G'_{RC} we have to replace the rule $(A \to BB, \emptyset, \{Y, Z\})$ by the rule $(A \to BB, \{X\}, \emptyset)$ and the rule $(B \to a, \emptyset, \{X, Z\})$ by the rule $(B \to a, \{Y\}, \emptyset)$ thus obtaining P'' instead of P'. Moreover, as mCF is a type with unit rules, we may replace every symbol H in a set of symbols in the random context grammars above by the corresponding unit rules $H \to H$; adding all these unit rules to P thus getting the set of rules $\tilde{P}$, we obtain the corresponding $mCF\text{-}RC_P$ grammars

$$\tilde{G}_{RC} = \left(\left(V^o, \{a\}^o, \tilde{P}, \Longrightarrow_m\right), \sqsubseteq_m, \tilde{P}', \Longrightarrow_{G_{RC}}\right)$$

and

$$\tilde{G}_{RC} = \left(\left(V^o, \{a\}^o, \tilde{P}, \Longrightarrow_m\right), \sqsubseteq_m, \tilde{P}'', \Longrightarrow_{G_{RC}}\right).$$

Hence, $Ps(L)$ is a multiset language in all the families $\mathcal{L}(mCF\text{-}RC_O(0, 2/1))$, $\mathcal{L}(mCF\text{-}RC_O(1/1, 1/1))$, $\mathcal{L}(mCF\text{-}R_PC(0, 2))$, and $\mathcal{L}(mCF\text{-}RC_P(1, 1))$.

In [3], a relation between $mCF_{-\lambda}\text{-}RC_O(*/1, */1)$ multiset grammars and linear bounded automata was established where $CF_{-\lambda}$ indicates that no erasing productions of the form $H \to \lambda$ are allowed. In the following, we shall restrict ourselves to show the computational completeness of several models introduced in this paper.

2.6 Antiport P Systems

The reader is supposed to be familiar with basic elements of membrane computing, e.g., from [16]; comprehensive information can be found on the P systems web page `http://psystems.disco.unimib.it`.

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

$$\Pi = (V, T, \mu, w_1, \ldots, w_d, R_1, \ldots, R_d, i_0),$$

where:

- V is the alphabet of *objects*,
- T is the alphabet of *terminal objects,*
- μ is the *membrane structure* (it is assumed that we have d membranes, labeled with $1, 2, \ldots, d$, the skin membrane usually being labeled with 1),

- w_i, $1 \leq i \leq d$, are strings over V representing the *initial* multiset of *objects* present in the membranes of the system,
- R_i, $1 \leq i \leq d$, are finite sets of *antiport rules* of the form x/y, for some $x, y \in V^*$, associated with membrane i,
- i_0 is the output membrane.

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 here, 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.

In this paper, we consider antiport P systems working in the sequential derivation mode (e.g., see [7]), i.e., a computation starts with the multisets specified by $w_1, \ldots, w_d$ in the d membranes, and in each time unit, we choose a rule assigned to some membrane in such a way that we can identify objects inside and outside the corresponding membrane to be affected by the selected antiport rule. The computation is successful if and only if it halts; the output of a halting computation consists of the different terminal symbols in the designated output membrane i_0 at the end of the computation.

The set of all multisets generated in this way by the system Π is denoted by $L(\Pi)$. The families of multisets $L(\Pi)$ generated as above by systems with at most d membranes and rules of weight at most g are denoted by $\mathcal{L}(P_d(anti_g))$. When any of these parameters d, g is not bounded, it is replaced by $*$.

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

$$\Pi_f = (V, T, \mu, w_1, \ldots, w_d, R_1', \ldots, R_d'),$$

where $V, T, \mu, w_1, \ldots, w_d$ are defined as above and R_i', $1 \leq i \leq d$, are finite sets of *antiport rules with forbidden context* of the form $(x/y, z)$, for some $x, y, z \in V^*$, 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 [10]; for variants of P systems with permitting and forbidden contexts also see [18].) A computation of Π_f is performed in a similar way as described for antiport systems. The families of sets $L(\Pi)$ of multisets computed by such systems with at most d membranes, rules of weight at most g, and the forbidden multiset of length at most l are denoted by $\mathcal{L}(f_l P_d(anti_g))$. When any of these parameters d, g, l is not bounded, it is replaced by $*$.

3 Results for RC (Multiset) Grammars

In this section we prove some results for random context grammars working on multisets or on strings and finally show a corresponding result for antiport P systems:

Theorem 1 $$\begin{aligned} Ps\left(\mathcal{L}\left(RE\right)\right) &= \mathcal{L}\left(mCF\text{-}RC_O\left(1/1,1/1\right)\right) \\ &= \mathcal{L}\left(mCF\text{-}RC_P\left(1,1\right)\right). \end{aligned}$$

Proof. The theorem is proved by showing the two inclusions

$$\mathcal{L}\left(mCF\text{-}RC_O\left(1/1,1/1\right)\right) \subseteq \mathcal{L}\left(mCF\text{-}RC_P\left(1,1\right)\right)$$

and

$$Ps\left(\mathcal{L}\left(RE\right)\right) \subseteq \mathcal{L}\left(msCF\text{-}R_OC\left(1/1,1/1\right)\right).$$

Let us first show the inclusion

$$\mathcal{L}\left(mCF\text{-}RC_O\left(1/1,1/1\right)\right) \subseteq \mathcal{L}\left(mCF\text{-}RC_P\left(1,1\right)\right).$$

Consider a *random context multiset grammar* (based on the multiset order relation) G_{RC} of type $mCF\text{-}RC_O\left(1/1,1/1\right)$

$$\left(G, P', \Longrightarrow_{G_{RC}}, w\right),$$

where $G = \left(O, O_T, P, \Longrightarrow_G\right)$ is a grammar scheme of type mCF; then we define the *random context grammar* G'_{RC} (based on the applicability of productions) of type X

$$\left(G', \sqsubseteq, P'', \Longrightarrow_{G'_{RC}}, w\right)$$

as follows: By adding all unit rules $H \rightarrow H$ for any symbol $H \in O$ to P we get the set of rules $\tilde{P}$, thus obtaining the corresponding mCF grammar scheme $G' = \left(O, O_T, \tilde{P}, \Longrightarrow_{G'}\right)$. Then we replace every symbol H in a set of symbols R or Q in the productions (q, R, Q) of P' by the corresponding unit rules $H \rightarrow H$; in that way, we obtain the new set of productions P''. From this construction, we immediately infer $L\left(G'_{RC}\right) = L\left(G_{RC}\right)$.

Based on the results established in [13] and [8], for proving the second inclusion,

$$Ps\left(\mathcal{L}\left(RE\right)\right) \subseteq \mathcal{L}\left(mCF\text{-}RC_O\left(1/1,1/1\right)\right),$$

we may consider a recursively enumerable language $L \in RE$ to be given by a matrix grammar in f-binary normal form G_M with $G_M = (N, T, P, M, F, S)$, $N = N_1 \cup N_2 \cup \{S, \#\}$, with these three sets being mutually disjoint, $F \subseteq \{A \rightarrow \# \mid A \in N_2\}$; we assume the matrices in M to be uniquely labeled by elements of a set of labels $Lab \subseteq \mathbb{N}$ such that the start matrix $(S \rightarrow X_0 A_0)$ has label 1 and the final matrix $(f \rightarrow \lambda)$ has the label 0. We then construct an $mCF\text{-}RC_O\left(1/1,1/1\right)$ grammar G_{RC} generating $Ps\left(L\right)$ as follows:

$$\begin{array}{ll}
G_{RC} & = (G_m, \sqsubseteq_m, P', \Longrightarrow_{G_{RC}}, S)\,, \\
G_m & = (V^o, T^o, P'', \Longrightarrow_m)\,, \\
V & = N \cup \{[i,j] \mid i \in Lab, 1 \leq j \leq 3\}\,, \\
P'' & = \{S \to X_0 A_0\} \cup \{f \to \lambda\} \\
 & \cup\ \{X_r \to [r,1]\,, A_r \to w_{r,1}\,[r,2]\,, \\
 & \quad [r,1] \to [r,3]\,, [r,2] \to w_{r,2}\,[r,3]\,, [r,3] \to Y_r \\
 & \quad \mid r : (X_r \to Y_r, A_r \to w_{r,1} w_{r,2}) \in M\} \\
 & \cup\ \{X \to Y \mid r : (X_r \to Y_r, A_r \to \#) \in M\}\,, \\
P' & = \{(S \to X_0 A_0, \emptyset, \emptyset)\} \cup \{(f \to \lambda, \emptyset, \emptyset)\} \\
 & \cup\ \{(X_r \to [r,1]\,, \{A_r\}\,, \emptyset)\,, (A_r \to w_{r,1}\,[r,2]\,, \{[r,1]\}\,, \{[r,2]\})\,, \\
 & \quad ([r,1] \to [r,3]\,, \{[r,2]\}\,, \emptyset)\,, ([r,2] \to w_{r,2}, \{[r,3]\}\,, \emptyset)\,, \\
 & \quad ([r,3] \to Y_r, \emptyset, \{[r,2]\}) \\
 & \quad \mid r : (X_r \to Y_r, A_r \to w_{r,1} w_{r,2}) \in M\} \\
 & \cup\ \{(X_r \to Y_r, \emptyset, \{A_r\}) \mid r : (X_r \to Y_r, A_r \to \#) \in M\}\,.
\end{array}$$

The initial matrix $1 : (S \to X_0 A_0)$ is simply simulated by the production $(S \to X_0 A_0, \emptyset, \emptyset)$, the final matrix $0 : (f \to \lambda)$ by the production $(f \to \lambda, \emptyset, \emptyset)$.

For simulating a matrix $r : (X_r \to Y_r, A_r \to \#) \in M$ used in the appearance checking mode we use the production $(X_r \to Y_r, \emptyset, \{A_r\})$.

For simulating a matrix $r : (X_r \to Y_r, A_r \to w_{r,1} w_{r,2}) \in M$ we have to apply the productions
$(X_r \to [r,1]\,, \{A_r\}\,, \emptyset)$,
$(A_r \to w_{r,1}\,[r,2]\,, \{[r,1]\}\,, \{[r,2]\})$,
$([r,1] \to [r,3]\,, \{[r,2]\}\,, \emptyset)$,
$([r,2] \to w_{r,2}, \{[r,3]\}\,, \emptyset)$, and
$([r,3] \to Y_r, \emptyset, \{[r,2]\})$
in exactly this sequence in order to get rid of the newly introduced variables of the form $[r,j]$, $r \in Lab$, $1 \leq j \leq 3$; the production $(A_r \to w_{r,1}\,[r,2]\,, \{[r,1]\}\,, \{[r,2]\})$ is the only one where we use both permitting and forbidden context, but in that way we can guarantee that exactly one variable A is affected. In sum, it is easy to see that $L\,(G_{RC}) = Ps\,(L)$. □

The following results for strings are well known (e.g., see [5]); they can be proved by using similar constructions as those elaborated in the proof of Theorem 1:

Theorem 2 $\mathcal{L}\,(RE) = \mathcal{L}\,(CF\text{-}RC_O\,(1/1,1/1)) = \mathcal{L}\,(CF\text{-}RC_P\,(1,1))\,.$

We now turn back to multiset processing and consider arbitrary multiset productions; for unary alphabets, the following result of computational completeness is also shown in [12]:

Theorem 3 $Ps\,(\mathcal{L}\,(RE)) = \mathcal{L}\,(mARB\text{-}RC_O\,(0,1/1)) = \mathcal{L}\,(mARB\text{-}RC_P\,(0,1))\,.$

Proof. We again start with a context-free matrix grammar in f-binary normal form. A matrix $r : (X_r \to Y_r, A_r \to w_r)$ now can easily be simulated by the production $(X_r A_r \to Y_r w_r, \emptyset, \emptyset)$ and a matrix $(X_r \to Y_r, A_r \to \#)$ by the production $(X_r \to Y_r, \emptyset, \{A_r\})$ or $(X_r \to Y_r, \emptyset, \{A_r \to A_r\})$; all these forms of productions in the random context multiset grammars fulfill the required conditions. The remaining details of the construction are obvious and therefore omitted. □

As already discussed in [12], multiset grammars with arbitrary multiset productions nicely correspond to antiport P systems working in the sequential mode in only one membrane; hence, from the preceding theorem we immediately infer the following result:

Corollary 4 $\mathcal{L}(f_1 P_1(anti_2)) = Ps(\mathcal{L}(RE))$.

Proof. In order to fulfill the required conditions, a matrix $r : (X_r \to Y_r, A_r \to w_r)$ with $w_r = w_{r,1} w_{r,2}$, $w_{r,1}, w_{r,2} \in N_2 \cup T \cup \{\lambda\}$, now has to be simulated by the two antiport rules with forbidden context $(X_r A_r / (r,1)\, w_{r,1}, \lambda)$ and $((r,1) / Y_r w_{r,2}, \lambda)$ and a matrix $r : (X_r \to Y_r, A_r \to \#)$ by the antiport rule with forbidden context $(X_r / Y_r, A_r)$. The remaining details of the construction are similar to those in the preceding proofs and therefore omitted. □

4 RC Checkers and RC Transducers

Let $(O, \sqsubseteq)$ be a partially ordered set. Then a *random context checker* (an *RC checker*) over $(O, \sqsubseteq)$ is of the form (R, Q) with $R \cup Q \subseteq O$ and R, Q both being finite sets. Using RC checkers in a generating grammar means that we have to check whether at least one checker is consistent with the current configuration before continuing the computation. The idea of RC checkers reminds us of the checkers used in Darwinian P systems (see [4] and [6]) where finite multiset automata are used as checkers.

A *grammar scheme G_{RCC} with random context checkers* of type X (based on an order relation on the objects) is a construct

$$(G, \sqsubseteq, H, \Longrightarrow_{G_{RCC}}),$$

where:

- $G = (O, O_T, P, \Longrightarrow_G)$ is a grammar scheme of type X;
- $(O, \sqsubseteq)$ is a partially ordered set;
- H is a set of RC checkers over $(O, \sqsubseteq)$;
- $\Longrightarrow_{G_{RCC}}$ is the derivation relation assigned to G_{RCC} such that for any $x, y \in O$, $x \Longrightarrow_{G_{RCC}} y$ if and only if for some rule $q \in P$, $x \Longrightarrow_G y$ by q and, moreover, there exists a checker $(R, Q) \in H$ such that $r \sqsubseteq x$ holds true for all $r \in R$ as well as $s \sqsubseteq x$ does not hold true for any $s \in Q$ (i.e., x is consistent with one of the checkers in H).

For short, we shall speak of an X-$RCC\,(m,n)$ grammar scheme if for any $(R,Q)\in H$ the set R contains at most m objects and the set Q contains at most n objects. Moreover, we shall call G_{RCC} an X-$RCC\,(m/k,n/l)$ grammar scheme if and only if it is an X-$RCC\,(m,n)$ grammar scheme and for any $(R,Q)\in H$, all objects in the set R are k-minimal and all objects in the set Q are l-minimal. If any of the parameters k,l,m,n is not to be specified, we replace it by $*$. A *grammar with random context checkers (an RCC grammar)* is a pair (G_{RCC},w) where $w\in O$ is the *axiom.* The corresponding families of languages generated by RCC grammars of types X are denoted by $\mathcal{L}\,(X\text{-}RCC\,(m/k,n/l))$.

The notion of RC checkers can easily be extended to P systems (in some analogy to Darwinian P systems, see [4] and [6]) where a checker for the whole system consists of a checker for each membrane region:

An *(extended) antiport P system with RC checkers* is a construct

$$\Pi_{RCC}=(V,T,\mu,w_1,\dots,w_d,R_1,\dots,R_d,H)\,,$$

where $V,T,\mu,w_1,\dots,w_d,R_1,\dots,R_d$ are defined as for antiport P systems and H is a finite set of membrane checkers, where each *membrane checker* is of the form $(h_1,\dots,h_d)$ with h_i, $1\le i\le m$, being RC checkers over $(V^o,\sqsubseteq_m)$. In this case, the application of the antiport rule x/y is only possible if for some membrane checker $(h_1,\dots,h_d)\in H$ the contents of each region i is consistent with the RC checker h_i. The families of sets $L\,(\Pi)$ of multisets computed by such systems with at most m membranes and rules of weight at most g such that for any (R,Q) occurring in H the set R contains at most m objects and the set Q contains at most n objects and all objects in the set R are k-minimal and all objects in the set Q are l-minimal, are denoted by $\mathcal{L}\,(RCC\,(m/k,n/l)\,P_d\,(anti_g))$. When any of these parameters d,g,k,l,m,n is not bounded, it is replaced by $*$.

Theorem 5 $Ps\,(\mathcal{L}\,(RE))=\mathcal{L}\,(mCF\text{-}RCC\,(1/1,*/1))\,.$

Proof. We start with a context-free matrix grammar in f-binary normal form $G_M=(N,T,P,M,F,S)$, $N=N_1\cup N_2\cup\{S,\#\}$, with these three sets being mutually disjoint, $F\subseteq\{A\to\#\mid A\in N_2\}$; we assume the matrices in M to be uniquely labeled by elements of a set of labels $Lab\subseteq\mathbb{N}$ such that the start matrix $(S\to X_0A_0)$ has label 1 and the final matrix $(f\to\lambda)$ has the label 0. We then construct an mCF-$RCC\,(1/1,*/1)$ grammar G_{RCC} generating $Ps\,(L)$ as follows:

$$\begin{aligned}
G_{RCC}&=(G_m,\sqsubseteq_m,H,\Longrightarrow_{G_{RCC}},S)\,,\\
G_m&=(V^o,T^o,P'',\Longrightarrow_m)\,,\\
V&=N_1\cup\{[i,j]\mid i\in Lab,1\le j\le 4\}\cup N_2\cup\{S,\#\}\,,\\
N'&=N_1\cup\{[i,j]\mid i\in Lab,1\le j\le 4\}\,,\\
P''&=\{S\to X_0A_0\}\cup\{(f\to\lambda)\}\\
&\quad\cup\ \{A_r\to[r,1]\,w_{r,1},X_r\to[r,2]\,,[r,1]\to[r,3]\,w_{r,2},\\
&\qquad[r,2]\to[r,4]\,,[r,4]\to\lambda,[r,3]\to Y_r\\
&\qquad\mid r:(X_r\to Y_r,A_r\to w_{r,1}w_{r,2})\in M\}\\
&\quad\cup\ \{X_r\to[r,1]\,,[r,1]\to Y_r\mid r:(X_r\to Y_r,A_r\to\#)\in M\}\,,
\end{aligned}$$

$$\begin{aligned} H = {} & \{(\emptyset, N' - \{S\})\} \cup \{(\{X\}, N' - \{X\}) \mid X \in N_1\} \\ & \cup \{(\{X_r\}, (N' - \{X_r, [r,1]\})), (\{[r,1]\}, (N' - \{[r,1], [r,2]\})), \\ & \quad (\{[r,2]\}, (N' - \{[r,2], [r,3]\})), (\{[r,3]\}, (N' - \{[r,3], [r,4]\})) \\ & \quad \mid r : (X_r \to Y_r, A_r \to w_{r,1} w_{r,2}) \in M\} \\ & \cup \{(\{[r,1]\}, N' - \{A_r\}) \mid r : (X_r \to Y_r, A_r \to \#) \in M\}. \end{aligned}$$

The RC checker $(\emptyset, N' - \{S\})$ is needed before starting a derivation with applying the initial production $S \to X_0 A_0$. After the simulation of a matrix exactly one of the control symbols from N_1 is present, hence, we need the RC checkers $(\{X\}, N' - \{X\})$ for $X \in N_1$ to continue the derivation. For checking the non-appearance of the symbol A_r (by a matrix of the form $r : (X_r \to Y_r, A_r \to \#)$) we use the intermediate symbol $[r,1]$ together with the RC checker $(\{[r,1]\}, N' - \{A_r\})$; this new symbol $[r,1]$ only appears in this intermediate step.

The simulation of a matrix $r : (X_r \to Y_r, A_r \to w_{r,1} w_{r,2})$ needs a more complex simulation procedure; in the following table, we list the productions in the first column and the corresponding RC checkers used to check the underlying multiset after the application of this rule; the sequence of productions listed in the table below is constructed in such a way that these productions have to be applied in exactly this sequence, otherwise the derivation will be blocked.

production	RC checker
$A_r \to [r,1]\, w_{r,1}$	$(\{X_r\}, (N' - \{X_r, [r,1]\}))$
$X_r \to [r,2]$	$(\{[r,1]\}, (N' - \{[r,1], [r,2]\}))$
$[r,1] \to [r,3]\, w_{r,2}$	$(\{[r,2]\}, (N' - \{[r,2], [r,3]\}))$
$[r,2] \to [r,4]$	$(\{[r,3]\}, (N' - \{[r,3], [r,4]\}))$
$[r,4] \to \lambda$	$(\{[r,3]\}, (N' - \{[r,3]\}))$
$[r,3] \to Y_r$	$(\{Y_r\}, N' - \{Y_r\})$

The RC checkers in the table above are of the form $(M', N' - M)$ where $M' \subsetneqq M$ and M contains the symbols from N' currently occurring in the multiset. If the production $A_r \to [r,1]\, w_{r,1}$ is applied more than once, then after the application of $X_r \to [r,2]$ we can apply the production $[r,1] \to [r,3]\, w_{r,2}$ only once, because the resulting multiset is not valid for any of the RC checkers; the RC checker $(\{[r,2]\}, (N' - \{[r,2], [r,3]\}))$ can only be used if originally the production $A_r \to [r,1]\, w_{r,1}$ has been applied only once. On the other hand, this production has to be applied, because otherwise we cannot apply $X_r \to [r,2]$. As the RC checker $(\{[r,3]\}, (N' - \{[r,3]\}))$ is subsumed by the RC checker $(\{[r,3]\}, (N' - \{[r,3], [r,4]\}))$, we need not take it into H. Finally we would like to point out that the permitting contexts in our construction are needed to guarantee that the production $A_r \to [r,1]\, w_{r,1}$ has been applied before X_r evolves to Y_r.

In sum, G_{RCC} fulfills the required complexity parameters, and obviously we have $L(G_{RCC}) = Ps(L)$, which observations complete the proof. □

Adding the final RC checker $(\emptyset, N' \cup N_2)$ in the proof elaborated above, we could also consider a slightly different variant of a grammar with RC checkers where we check the validity of the object *after* the application of a production. The same observation holds true for the next results, too.

Theorem 6
$$\begin{aligned} Ps(\mathcal{L}(RE)) &= \mathcal{L}(mARB\text{-}RCC(1/1, 1/1)) \\ &= \mathcal{L}(mARB\text{-}RCC(0, */1)). \end{aligned}$$

Proof. Once more we start with a context-free matrix grammar in f-binary normal form, with the matrices uniquely labelled by elements of some finite set $Lab \subseteq \mathbb{N}$. Moreover, for each label $r \in Lab$ we introduce a new variable $[r]$. A matrix $r : (X_r \to Y_r, A_r \to w_r)$ then can be simulated by using the production $X_r A_r \to Y_r w_r$ together with the RC checker $(\{X_r\}, \emptyset)$, and a matrix $r : (X_r \to Y_r, A_r \to \#)$ by using the productions $X_r \to [r]$ and $[r] \to Y_r$ together with the RC checkers $(\{X_r\}, \emptyset)$ and $(\{[r]\}, \{A_r\})$, respectively; all these forms of productions in the $mARB\text{-}RCC$ grammar fulfill the required conditions, i.e., we have shown $Ps(\mathcal{L}(RE)) = \mathcal{L}(mARB\text{-}RCC(1/1, 1/1))$. For proving $Ps(\mathcal{L}(RE)) = \mathcal{L}(mARB\text{-}RCC(0, */1))$, let $N_2' = N_2 \cup \{[r] \mid r \in Lab\}$; then we use the same productions as before, but we replace the RC checker $(\{X_r\}, \emptyset)$ by $(\emptyset, N_2' - \{X_r\})$ and $(\{[r]\}, \{A_r\})$ by $(\emptyset, (N_2' - \{X_r\}) \cup \{A_r\})$. The remaining details of the construction are rather obvious and therefore omitted. □

As already discussed in the preceding section, multiset grammars using arbitrary multiset productions can be interpreted as antiport P systems with one membrane; each membrane checker then simply consists of only one RC checker that checks the contents of the skin region:

Corollary 7
$$\begin{aligned} Ps(\mathcal{L}(RE)) &= \mathcal{L}(RCC(1/1, 1/1)\, P_1(anti_2)) \\ &= \mathcal{L}(RCC(0, */1)\, P_1(anti_2)). \end{aligned}$$

Proof. We can use constructions quite similar to those elaborated in the preceding proof; again start with a context-free matrix grammar in f-binary normal form. The only difference is that a matrix $r : (X_r \to Y_r, A_r \to w_{r,1} w_{r,2})$ now has to be simulated by using the antiport rules $X_r / [r]\, w_{r,1}$ and $[r] / Y_r w_{r,2}$ together with the RC checkers $(\{X_r\}, \emptyset)$ and $(\{[r]\}, \emptyset)$ or $(\emptyset, N_2' - \{X_r\})$ and $(\emptyset, N_2' - \{[r]\})$, respectively. The remaining details of the construction again are obvious and therefore omitted. □

We now extend the idea of RC checkers to RC transducers having in mind some results already elaborated, for example, in [2], with the main idea to observe the steps in a derivation of a grammar and to take the collection of the outputs of checking the current object before each derivation step as the result of the derivation – instead of the result of the derivation itself.

Let $(O, \sqsubseteq)$ be a partially ordered set and $(O', +, e, \sqsubseteq')$ be a partially ordered semi-group with the operation $+$ and the unit element e. Then a *random context transducer* (an *RC transducer*) over $((O, \sqsubseteq), (O', +, e, \sqsubseteq'))$ is of the form (R, Q, t)

with $R \cup Q \subseteq O$, $t \in O'$, and R, Q both being finite sets, i.e., (R, Q) is an RC checker. Applying RC transducers in a grammar now means that if for an RC transducer (R, Q, t) the checker (R, Q) is consistent with the current object, then we may choose it and generate the output t. For example, we may take the commutative semi-group $(O', +, e, \sqsubseteq') = (V^o, \cup_m, \lambda, \sqsubseteq_m)$ where $\cup_m$ denotes the union (addition) of multisets and λ denotes the empty multiset or the non-commutative semi-group $(O', +, e, \sqsubseteq') = (V^*, \circ, \lambda, \sqsubseteq_s)$ where $\circ$ denotes the concatenation of strings, λ is the empty string, and $\sqsubseteq_s$ denotes the substring relation. We now may use RC transducers to generate subsets of O' by collecting the outputs during a computation of the corresponding RCC grammar obtained by using the RC transducers as RC checkers; formally, we give the following definitions:

A *grammar scheme* G_{RCT} *with random context transducers* of type X (based on an order relation on the objects) is a construct

$$(G, \sqsubseteq, (O', +, e, \sqsubseteq'), H, \Longrightarrow_{G_{RCT}}),$$

where:

- $G = (O, O_T, P, \Longrightarrow_G)$ is a grammar scheme of type X;
- $(O, \sqsubseteq)$ is a partially ordered set;
- $(O', +, e, \sqsubseteq')$ is a partially ordered semi-group with unit element e;
- H is a set of RC transducers over $((O, \sqsubseteq), (O', +, e))$;
- $\Longrightarrow_{G_{RCT}}$ is the derivation relation assigned to G_{RCT} such that for any $x, y \in O$, $x \Longrightarrow_{G_{RCT}} y$ if and only if for some rule $q \in P$, $x \Longrightarrow_G y$ by q and, moreover, there exists a transducer $(R, Q, t) \in H$ such that $r \sqsubseteq x$ holds true for all $r \in R$ as well as $s \sqsubseteq x$ does not hold true for any $s \in Q$ (i.e., x is consistent with the checker in one of the transducers in H).

As the result of a successful derivation, i.e., a derivation yielding a terminal object from O_T, we do not take this terminal object, yet instead, if (R_1, Q_1, t_1), $\dots$, (R_j, Q_j, t_j) is a sequence of RC transducers that can be used during the derivation, we take $t_1 + \cdots + t_j$ as a result of this derivation. For example, if $(O', +, e, \sqsubseteq') = (V^o, \cup_m, \lambda, \sqsubseteq_m)$, then the output consists of the multiset obtained by collecting all outputs during the derivation or, if $(O', +, e, \sqsubseteq') = (V^*, \circ, \lambda, \sqsubseteq_s)$, then we consider the sequence of output strings as the string generated as a result of the derivation. The union of all these elements $t_1 + \cdots + t_j$ yields the language $L(G_{RCT}) \subseteq O'$.

For $(O', +, e, \sqsubseteq') = (V^o, \cup_m, \lambda, \sqsubseteq_m)$, the results of terminal derivations in G_{RCT} yield the multiset language $L_m(G_{RCT})$ consisting of all multisets being the union of all multisets produced by the RC transducers during a derivation yielding a terminal object. On the other hand, for $(O', +, e, \sqsubseteq') = (V^*, \circ, \lambda, \sqsubseteq_s)$, we obtain the string language $L_s(G_{RCT})$ consisting of all strings generated as the sequence of strings produced by the RC transducers during a derivation yielding a terminal object. As any string of Σ^* can be interpreted as a multiset over Σ, in the case $O' = \Sigma^*$ we may consider both $L_s(G_{RCT})$ and $L_m(G_{RCT})$.

We then shall speak of an X-$RCT\,(m,n,s)$ grammar scheme if for any $(R,Q,t) \in H$ the set R contains at most m objects, the set Q contains at most n objects and t is s-minimal in $(O', \sqsubseteq')$. Moreover, we shall call G_{RCT} an X-$RCT\,(m/k,n/l,s)$ grammar scheme if and only if it is an X-$RCT\,(m,n,s)$ grammar scheme and for any $(R,Q) \in H$, all objects in the set R are k-minimal and all objects in the set Q are l-minimal. If any of the parameters k,l,m,n,s is not to be specified, we replace it by $*$. A *grammar with random context transducers (an RCT grammar)* is a pair (G_{RCT}, w) where $w \in O$ is the *axiom*. The corresponding families of multiset and string languages generated by RCT grammars of types X are denoted by $\mathcal{L}_m\,(X\text{-}RCT\,(m/k,n/l,s))$ and $\mathcal{L}_s\,(X\text{-}RCT\,(m/k,n/l,s))$, respectively.

For antiport P systems, each membrane checker yields an output, i.e., we define an *(extended) antiport P system with RC transducers* as a construct

$$\Pi_{RCT} = (V, T, \Sigma, \mu, w_1, \ldots, w_d, R_1, \ldots, R_d, H)\,,$$

where $V, T, \mu, w_1, \ldots, w_d, R_1, \ldots, R_d$ are defined as for antiport P systems, Σ is the output alphabet, and H is a finite set of membrane transducers, where each *membrane transducer* is of the form $(h_1, \ldots, h_d, t)$ with $t \in \Sigma^*$ and h_i, $1 \leq i \leq m$, being RC checkers over $(V^o, \sqsubseteq_m)$. With the parameters already explained above and with s denoting the maximal length of t in a membrane transducer $(h_1, \ldots, h_d, t)$, the resulting families of sets of multisets and of strings are denoted by $\mathcal{L}_m\,(RCT\,(m/k,n/l,s)\,P_d\,(anti_g))$ and $\mathcal{L}_s\,(RCT\,(m/k,n/l,s)\,P_d\,(anti_g))$.

Theorem 8 $Ps\,(\mathcal{L}\,(RE)) = \mathcal{L}_m\,(mCF\text{-}RCT\,(1/1,*/1,1,1))\,.$

Proof. The result directly follows from the proofs of Theorem 5: all the productions there are of the form $X \to Yu$ with $X \in N_1 \cup N_2 \cup \{S\} \cup N_1'$, $Y \in N_1 \cup N_1' \cup \{\lambda\}$ and $u \in N_1 \cup T \cup \{\lambda\}$. Hence, we immediately obtain the RCT grammar from the RCC grammar constructed there by replacing the RC checker (R,Q) checking the object after the application of the rule $X \to Yu$ as listed in the table in the proof of Theorem 5 by the corresponding RC transducer (R,Q,u). □

Corollary 9 $\mathcal{L}\,(RE) = \mathcal{L}_s\,(mCF\text{-}RCT\,(1/1,*/1,1,1))\,.$

Proof. The result directly follows from the proof of Theorem 5 and of Theorem 8 as well as of the constructions given for matrix grammars in the proofs of [8] which show that for every recursively enumerable language we can construct a matrix grammar with appearance checking that generates the symbols of a terminal string in the correct sequence. □

The following results are obvious from the constructions elaborated in the proofs above, hence, we omit their proofs:

Corollary 10 $Ps\,(\mathcal{L}\,(RE)) = \mathcal{L}_m\,(mARB\text{-}RCT\,(1/1,1/1,1))$
$= \mathcal{L}_m\,(mARB\text{-}RCT\,(0,*/1,1))\,.$

Corollary 11 $\mathcal{L}(RE) = \mathcal{L}_s(mARB\text{-}RCT(1/1, 1/1, 1))$
$= \mathcal{L}_s(mARB\text{-}RCC(0, */1, 1))$.

Corollary 12 $Ps(\mathcal{L}(RE)) = \mathcal{L}_m(RCT(1/1, 1/1, 1)\, P_1(anti_2))$
$= \mathcal{L}_m(RCT(0, */1, 1)\, P_1(anti_2))$.

Corollary 13 $\mathcal{L}(RE) = \mathcal{L}_s(RCT(1/1, 1/1, 1)\, P_1(anti_2))$
$= \mathcal{L}_s(RCT(0, */1, 1)\, P_1(anti_2))$.

5 Conclusion

We have introduced a new general model for random context grammars based on a partial order relation on the objects the grammars deal with. Based on the idea of checking the current object to be smaller than some given ones, but not to be smaller than some other ones, we also introduced grammars with random context checkers as well as grammars yielding as an output the multiset or sequence of objects generated by random context transducers during a computation. These ideas and notions were carried over to antiport P systems as well. For all these models of generating sets of multisets or strings we established computational completeness results, i.e., we proved that – even with some quite restricted bounds on the complexity of these generating devices – we obtain $Ps(\mathcal{L}(RE))$ or $\mathcal{L}(RE)$, respectively.

On the other hand, we have not considered variants of the models introduced in this paper which yield language classes below $Ps(\mathcal{L}(RE))$ or $\mathcal{L}(RE)$, respectively; for example, $\mathcal{L}(mCF\text{-}RC_O(0,*)) \subsetneqq Ps(\mathcal{L}(RE))$, because we know (e.g., see [5]) that $\mathcal{L}(CF\text{-}RC_O(0,*)) \subsetneqq \mathcal{L}(RE)$. We think that especially the families $\mathcal{L}(msCF\text{-}RC_O(0,*/k))$ and $\mathcal{L}(msCF\text{-}RC_O(*/k,0))$, especially for $k = 1, 2$, should be investigated/characterized.

The idea of RC checkers and RC transducers can be carried over to many other models of computation, especially to other models of P systems or tissue P systems (e.g., see [14]). Hence, there seems to be a broad area for future research based on the ideas and notions introduced in his paper.

Acknowledgements

The work of Marion Oswald is supported by FWF-project T225-N04.

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. M. Cavalliere: *Evolution, Communication, Observation: From Biology to Membrane Computing and Back.* PhD thesis, University of Sevilla, Spain, 2006.

3. E. Csuhaj-Varjú, C. Martín-Vide, V. Mitrana: Multiset automata. In [1], 69–84.
4. E. Csuhaj-Varjú, C. Martín-Vide, Gh. Păun, A. Salomaa: From Watson-Crick L-systems to Darwinian P systems. *Natural Computing*, 2 (2003), 299–318.
5. J. Dassow, Gh. Păun: *Regulated Rewriting in Formal Language Theory.* Springer, Berlin, 1989.
6. J. Dassow, E. Csuhaj-Varjú: On the syntactic complexity of Darwinian membrane systems. In volume II of the present proceedings.
7. R. Freund: Asynchronous P systems. *Proceedings WMC5*, 2004, 12–28.
8. R. Freund, Gh. Păun: On the number of nonterminal symbols in graph-controlled, programmed and matrix grammars. In *Machines, Computations, and Universality. 3rd MCU* (M. Margenstern, Y. Rogozhin, eds.), LNCS 2055, Springer, 2001, 214–225.
9. R. Freund, Gh. Păun, M. J. Pérez-Jiménez: Tissue P systems with channel states. *Theoretical Computer Science*, 330 (2005), 101–116.
10. R. Freund, M. Oswald: GP systems with forbidding context. *Fundamenta Informaticae*, 49, 1–3 (2002), 81–102.
11. R. Freund, M. Oswald: Modelling grammar systems by tissue P systems working in the sequential mode. In *Proceedings of Grammar Systems Workshop*, Budapest, 2004.
12. R. Freund, M. Oswald: Variants of small universal antiport P systems. In volume II of the present proceedings.
13. M. Kudlek, C. Martín-Vide, Gh. Păun: Toward a formal macroset theory. In [1], 123–134.
14. C. Martín-Vide, J. Pazos, Gh. Păun, A. Rodriguez-Paton: Tissue P systems. *Theoretical Computer Science*, 296, 2 (2003), 295–326.
15. Gh. Păun: Computing with membranes. *Journal of Computer and System Sciences*, 61, 1 (2000), 108–143.
16. Gh. Păun: *Membrane Computing – An Introduction.* Springer, Berlin, 2002.
17. A. Salomaa, G. Rozenberg, eds.: *Handbook of Formal Languages.* Springer, Berlin, 1997.
18. D. Sburlan: *Promoting and Inhibiting Contexts in Membrane Computing.* PhD thesis, University of Sevilla, Spain, 2006.
19. The P systems webpage: `http://psystems.disco.unimib.it`