RAIRO-Theor. Inf. Appl. 48 (2014) 23–38
DOI: 10.1051/ita/2013047

Available online at:
www.rairo-ita.org

REACTION AUTOMATA WORKING IN SEQUENTIAL MANNER*,**

Fumiya Okubo[1]

Abstract. Based on the formal framework of reaction systems by Ehrenfeucht and Rozenberg [*Fund. Inform.* **75** (2007) 263–280], reaction automata (RAs) have been introduced by Okubo *et al.* [*Theoret. Comput. Sci.* **429** (2012) 247–257], as language acceptors with multiset rewriting mechanism. In this paper, we continue the investigation of RAs with a focus on the two manners of rule application: maximally parallel and sequential. Considering restrictions on the workspace and the λ-input mode, we introduce the corresponding variants of RAs and investigate their computation powers. In order to explore Turing machines (TMs) that correspond to RAs, we also introduce a new variant of TMs with restricted workspace, called $s(n)$-restricted TMs. The main results include the following: (i) for a language L and a function $s(n)$, L is accepted by an $s(n)$-bounded RA with λ-input mode in sequential manner if and only if L is accepted by a $\log s(n)$-bounded one-way TM; (ii) if a language L is accepted by a linear-bounded RA in sequential manner, then L is also accepted by a P automaton [Csuhaj−Varju and Vaszil, vol. 2597 of *Lect. Notes Comput. Sci.* Springer (2003) 219–233.] in sequential manner; (iii) the class of languages accepted by linear-bounded RAs in maximally parallel manner is incomparable to the class of languages accepted by RAs in sequential manner.

Mathematics Subject Classification. 68Q05, 68Q45.

Keywords and phrases. Models of biochemical reactions, sequential reaction automata, space complexity, Turing machines.

* *This paper is a revised and extended version of a paper that was presented at NCMA 2012 [14].*

** *The work of F. Okubo was in part supported by Grants-in-Aid for Young Scientists (B) No. 24700304, Japan Society for the Promotion of Science and by Grants-in-Aid for JSPS Fellows No.* $25 \cdot 3528$*, Japan Society for the Promotion of Science.*

[1] Graduate School of Education, Waseda University, 1-6-1 Nishiwaseda, Shinjuku-ku, Tokyo 169-8050, Japan. f.okubo@akane.waseda.jp

Article published by EDP Sciences

© EDP Sciences 2014

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press

1. Introduction

In recent years, Ehrenfeucht and Rozenberg have introduced a formal model, called reaction systems [5], for investigating the functioning of the living cell, based on the idea that the functioning is decided by interactions between biochemical reactions, where two basic components (reactants and inhibitors) play a key role as a regulation mechanism in controlling interactions. In [6], it is shown that reaction systems provide a formal framework suited for investigating in an abstract level the way of emergence and evolution of biochemical events and modules. Recent papers continue the investigation of reaction systems in various topics motivated by biological and theoretical considerations, such as the issue of times for creating compounds [7], combinatorial properties of functions defined by reaction systems [8, 9, 18], probabilistic and quantum variants of reaction systems [11].

In the theory of reaction systems, a biochemical reaction is formulated as a triple $a = (R_a, I_a, P_a)$, where R_a is the set of molecules called reactants, I_a is the set of molecules called inhibitors, and P_a is the set of molecules called products. Let T be a set of molecules, and the result of applying a reaction a to T, denoted by $\mathrm{res}_a(T)$, is given by P_a if a is enabled by T (*i.e.*, if R_a is included in T and I_a is disjoint with T). Otherwise, the result is empty. Thus, $\mathrm{res}_a(T) = P_a$ if a is enabled on T, and $\mathrm{res}_a(T) = \emptyset$ otherwise. The result of applying a reaction a is extended to the set of reactions A, denoted by $\mathrm{res}_A(T)$, and an interactive process consisting of a sequence of $\mathrm{res}_A(T)$'s is properly introduced and investigated.

Inspired by the notion of reaction systems, reaction automata have been introduced in [15] as computing devices, and it has been shown that they are computationally universal by proving that all recursively enumerable languages are accepted by reaction automata. In [16], the investigation with reaction automata is continued with a focus on the formal language theoretic properties of space-bounded classes of reaction automata. Specifically, it is shown that all context-sensitive languages are accepted by exponential space bounded reaction automata. The notion of reaction automata may be regarded as an extension of reaction systems in the sense that reactants and inhibitors are employed as regulation in reaction automata, however they deal with multisets rather than usual sets as reaction systems do. Reaction automata are introduced as multiset rewriting devices that accept languages over an alphabet, where this feature is realized by a simple idea of feeding one symbol of an input string at each step of computation, to the device from the environment. In this sense, reaction automata may also be regarded as simplified variants of P automata introduced by Csuhaj−Varjú and Vaszil in [4] with no membrane structure.

In this paper, we continue the investigation of reaction automata with a focus on the way of rule application. We consider not only maximally parallel manner employed in [15, 16], but also sequential manner as the way of rule application, and compare the computational powers of reaction automata and their space-bounded variants in the above two manners. Reaction automata with λ-moves (introduced in [16]) in sequential manner are also investigated. Further, we explore a

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press

Turing machine corresponding to reaction automata, and introduce a new variant of Turing machines with restricted workspace, called $s(n)$-restricted Turing machines, which is a relaxation of the notion of restricted $s(n)$ space bounded Turing machines introduced in [2]. The idea for these restrictions is that the workspace used in the computation is provided depending on the number of symbols actually read from the input.

This paper is organized as follows. After preparing the basic notions and notations from formal language theory in Section 2.1, we formally describe the notion of reaction automata (RAs) and the classes of languages $\mathcal{RA}_X$, $\mathcal{RA}_X^\lambda$ with $X \in \{sq, mp\}$ in Section 2.2. We also introduce the space-bounded variants of reaction automata and the classes of languages $\mathcal{LRA}_{sq}$ accepted by them. In Section 2.3, Turing machines and their several variants, *e.g.*, $s(n)$-restricted Turing machines, are introduced. Then, in Section 3, we establish (i) the relation between space-bounded reaction automata in sequential manner and space-bounded one-way Turing machine; (ii) the relation between $\mathcal{LRA}_{sq}$ and the class of languages accepted by P automata in sequential manner, with the help of logarithmic-restricted Turing machines; (iii) the relation between $\mathcal{RA}_{sq}$ and $\mathcal{LRA}_{mp}$. Finally, concluding remarks as well as future research topics are briefly discussed in Section 4.

2. Preliminaries

2.1. Basic definitions

We assume that the reader is familiar with the basic notions of formal language theory. For unexplained details, refer to [12]. Let V be a finite alphabet. For a set $U \subseteq V$, the cardinality of U is denoted by $|U|$. The set of all finite-length strings over V is denoted by V^*. The empty string is denoted by λ. For a string $x \in V^*$, $|x|$ denotes the length of x, while for a symbol $a \in V$ we denote by $|x|_a$ the number of occurrences of a in x. For $x \in V^*$, define the r-tuple of natural numbers $\psi(w) = (|x|_{a_1}, |x|_{a_2}, \ldots, |x|_{a_r})$. w^R is the reversal of w, that is, $(a_1 a_2 \ldots a_n)^R = a_n \ldots a_2 a_1$. Further, for a string $x = a_1 a_2 \ldots a_n \in V^*$, $\hat{x}$ denotes the hat version of x, *i.e.*, $\hat{x} = \hat{a}_1 \hat{a}_2 \ldots \hat{a}_n$, where each $\hat{a}_i$ is in an alphabet $\hat{V} = \{\hat{a} \mid a \in V\}$ such that $V \cap \hat{V} = \emptyset$.

A morphism $h : V^* \to U^*$ such that $h(a) \in U$ for all $a \in V$ is called a coding, and it is a weak coding if $h(a) \in U \cup \{\lambda\}$ for all $a \in V$. A weak coding is a projection if $h(a) \in \{a, \lambda\}$ for each $a \in V$. The notion of a projection is extended to language L as $h(L) = \{h(w) \mid w \in L\}$. Further, for a class of language $\mathcal{L}$, $PR(\mathcal{L}) = \{h(L) \mid h : \text{projection}, L \in \mathcal{L}\}$.

We use the basic notations and definitions regarding multisets that follow [1,13]. A multiset over an alphabet V is a mapping $\mu : V \to \mathbf{N}$, where $\mathbf{N}$ is the set of non-negative integers and for each $a \in V$, $\mu(a)$ represents the number of occurrences of a in the multiset μ. The set of all multisets over V is denoted by $V^\#$, including the empty multiset denoted by μ_λ, where $\mu_\lambda(a) = 0$ for all $a \in V$. We can represent the multiset μ by any permutation of the string $w = a_1^{\mu(a_1)} \ldots a_n^{\mu(a_n)}$. Conversely,

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press

with any string $x \in V^*$ one can associate the multiset $\mu_x : V \to \mathbf{N}$ defined by $\mu_x(a) = |x|_a$ for each $a \in V$. In this sense, we often identify a multiset μ with its string representation w_μ or any permutation of w_μ. Note that the string representation of μ_λ is λ, *i.e.*, $w_{\mu_\lambda} = \lambda$. A usual set $U \subseteq V$ is regarded as a multiset μ_U such that $\mu_U(a) = 1$ if a is in U and $\mu_U(a) = 0$ otherwise. In particular, for each symbol $a \in V$, a multiset $\mu_{\{a\}}$ is often denoted by a itself.

For two multisets μ_1, μ_2 over V, we define one relation and four operations as follows:

- Inclusion : $\mu_1 \subseteq \mu_2$ iff $\mu_1(a) \leq \mu_2(a)$, for each $a \in V$;
- Sum : $(\mu_1 + \mu_2)(a) = \mu_1(a) + \mu_2(a)$, for each $a \in V$;
- Union : $(\mu_1 \cup \mu_2)(a) = \max\{\mu_1(a), \mu_2(a)\}$, for each $a \in V$;
- Intersection : $(\mu_1 \cap \mu_2)(a) = \min\{\mu_1(a), \mu_2(a)\}$, for each $a \in V$;
- Difference : $(\mu_1 - \mu_2)(a) = \mu_1(a) - \mu_2(a)$, for each $a \in V$ (for the case $\mu_2 \subseteq \mu_1$).

The sum for a family of multisets $M = \{\mu_i\}_{i \in I}$ is denoted by $\sum_{i \in I} \mu_i$. The union for a family of multisets $M = \{\mu_i\}_{i \in I}$ is also denoted by $\bigcup_{i \in I} \mu_i$. For a multiset μ and $n \in \mathbf{N}$, μ_n is defined by $\mu_n(a) = n \cdot \mu(a)$ for each $a \in V$. The weight of a multiset μ is $|\mu| = \sum_{a \in V} \mu(a)$.

2.2. Reaction automata

Inspired by the works of reaction systems, we have introduced the notion of reaction automata in [15] by extending sets in each reaction to multisets. Here, we start by recalling basic notions concerning reaction automata.

Definition 2.1. For a set S, a *reaction* in S is a 3-tuple $\mathbf{a} = (R_\mathbf{a}, I_\mathbf{a}, P_\mathbf{a})$ of finite multisets, such that $R_\mathbf{a}, P_\mathbf{a} \in S^\#$, $I_\mathbf{a} \subseteq \mu_S$ and $R_\mathbf{a} \cap I_\mathbf{a} = \emptyset$.

The multisets $R_\mathbf{a}$, $I_\mathbf{a}$ and $P_\mathbf{a}$ are called the *reactant* of $\mathbf{a}$, the *inhibitor* of $\mathbf{a}$ and the *product* of $\mathbf{a}$, respectively. These notations are extended to a multiset of reactions as follows: For a set of reactions A and a multiset α over A,

$$R_\alpha = \sum_{\mathbf{a} \in A} R_\mathbf{a}^{\alpha(\mathbf{a})}, \ I_\alpha = \bigcup_{\mathbf{a} \subseteq \alpha} I_\mathbf{a}, \ P_\alpha = \sum_{\mathbf{a} \in A} P_\mathbf{a}^{\alpha(\mathbf{a})}.$$

In this paper, we consider two ways for applying reactions, *i.e.*, sequential manner and maximally parallel manner, while only the latter manner is concerned in the previous papers.

Definition 2.2. Let A be a set of reactions in S and $\alpha \in A^\#$ be a multiset of reactions over A. Then, for a finite multiset $T \in S^\#$, we say that:

(1) α is *enabled by* T if $R_\alpha \subseteq T$ and $I_\alpha \cap T = \emptyset$.
(2) α is *enabled by* T *in sequential manner* if α is enabled by T with $|\alpha| = 1$.
(3) α is *enabled by* T *in maximally parallel manner* if there is no $\beta \in A^\#$ such that $\alpha \subset \beta$, and α and β are enabled by T.

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press

(4) By $En_A^{sq}(T)$ and $En_A^{mp}(T)$, we denote the sets of all multisets of reactions $\alpha \in A^{\#}$ which are enabled by T in sequential manner and in maximally parallel manner, respectively.
(5) The *results of* A *on* T, denoted by $Res_A^X(T)$ with $X \in \{sq, mp\}$, is defined as follows:

$$\mathrm{Res}_A^X(T) = \{T - R_\alpha + P_\alpha \mid \alpha \in En_A^X(T)\}.$$

We note that $\mathrm{Res}_A^X(T) = \{T\}$ if $En_A^X(T) = \emptyset$. Thus, if no multiset of reactions $\alpha \in A^{\#}$ is enabled by T, then T remains unchanged.

We are now in a position to introduce the notion of reaction automata.

Definition 2.3. A *reaction automaton* (RA) $\mathcal{A}$ is a 5-tuple $\mathcal{A} = (S, \Sigma, A, D_0, f)$, where

- S is a finite set, called the *background set of* $\mathcal{A}$;
- $\Sigma(\subseteq S)$ is called the *input alphabet of* $\mathcal{A}$;
- A is a finite set of reactions in S;
- $D_0 \in S^{\#}$ is an *initial multiset*;
- $f \in S$ is a special symbol which indicates the final state.

Definition 2.4. Let $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be an RA, $w = a_1 \dots a_n \in \Sigma^*$ and $X \in \{sq, mp\}$. An *interactive process in* $\mathcal{A}$ *with input* w in X manner is an infinite sequence $\pi = D_0, \dots, D_i, \dots$, where

$$\begin{cases} D_{i+1} \in \mathrm{Res}_A^X(a_{i+1} + D_i) & (\text{for } 0 \le i \le n-1), \text{ and} \\ D_{i+1} \in \mathrm{Res}_A^X(D_i) & (\text{for all } i \ge n). \end{cases}$$

In order to represent an interactive process π, we also use the "arrow notation" for $\pi : D_0 \to^{a_1} D_1 \to^{a_2} D_2 \to^{a_3} \dots \to^{a_{n-1}} D_{n-1} \to^{a_n} D_n \to D_{n+1} \to \dots$ By $IP_X(\mathcal{A}, w)$ we denote the set of all interactive processes in $\mathcal{A}$ with input w in X manner.

Recall that in [16], if it is allowed that $a_i = \lambda$ for some several $1 \le i \le n$, for an input string $w = a_1 \dots a_n$, an interactive process is said to be with λ-input mode. By $IP_X^\lambda(\mathcal{A}, w)$ we denote the set of all interactive processes in $\mathcal{A}$ with λ-input mode in X manner for the input w.

For an interactive process π in $\mathcal{A}$ with input w, if $En_A^X(D_m) = \emptyset$ for some $m \ge |w|$, then we have that $\mathrm{Res}_A(D_m) = \{D_m\}$ and $D_m = D_{m+1} = \dots$ In this case, considering the smallest m, we say that π *converges on* D_m (at the mth step). If an interactive process π converges on D_m, then D_m is called the *converging state* of π and each D_i of π is omitted for $i \ge m + 1$.

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press

Definition 2.5. Let $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be an RA and $X = \{sq, mp\}$. Then, the set of accepting interactive processes is defined as follows:

$$AIP_X(\mathcal{A}, w) = \{\pi \in IP_X(\mathcal{A}, w) \mid \pi \text{ converges on } D_m \text{ at the } m\text{th step for some } m \geq |w| \text{ and } f \subseteq D_m\},$$
$$AIP_X^{\lambda}(\mathcal{A}, w) = \{\pi \in IP_X^{\lambda}(\mathcal{A}, w) \mid \pi \text{ converges on } D_m \text{ at the } m\text{th step for some } m \geq |w| \text{ and } f \subseteq D_m\}.$$

The *language accepted by* $\mathcal{A}$ is defined as follows:

$$L_X(\mathcal{A}) = \{w \in \Sigma^* \mid AIP_X(\mathcal{A}, w) \neq \emptyset\},$$
$$L_X^{\lambda}(\mathcal{A}) = \{w \in \Sigma^* \mid AIP_X^{\lambda}(\mathcal{A}, w) \neq \emptyset\}.$$

Let $\mathcal{A}$ be an RA. Motivated by the notion of a workspace for a phrase-structure grammar [17], we define: for $w \in L_X(\mathcal{A})$ with $n = |w|$, and for π in $AIP_X(\mathcal{A}, w)$,

$$WS(w, \pi) = \max\{|D_i| \mid D_i \text{ appears in } \pi\ \}.$$

Further, the *workspace of* $\mathcal{A}$ *for* w is defined as:

$$WS(w, \mathcal{A}) = \min\{WS(w, \pi) \mid \pi \in AIP_X(\mathcal{A}, w)\ \}.$$

Definition 2.6. Let s be a function defined on $\mathbf{N}$ and $X = \{sq, mp\}$.

(1) An RA $\mathcal{A}$ is $s(n)$-*bounded* if for any $w \in L_X(\mathcal{A})$ with $n = |w|$, $WS(w, \mathcal{A})$ is bounded by $s(n)$.
(2) If a function $s(n)$ is a constant k (linear, exponential), then $\mathcal{A}$ is termed constant-bounded (resp. linear-bounded, exponential-bounded).
(3) The class of languages accepted by constant-bounded RAs (linear-bounded, exponential-bounded, arbitrary RAs) in X manner is denoted by $\mathcal{CRA}_X$ (resp. $\mathcal{LRA}_X$, $\mathcal{ERA}_X$, $\mathcal{RA}_X$).
(4) The class of languages accepted by constant-bounded RAs (linear-bounded, exponential-bounded, arbitrary RAs) with λ-input mode in X manner is denoted by $\mathcal{CRA}_X^{\lambda}$ (resp. $\mathcal{LRA}_X^{\lambda}$, $\mathcal{ERA}_X^{\lambda}$, $\mathcal{RA}_X^{\lambda}$).

From the definition, it obviously holds that $\mathcal{REG} = \mathcal{CRA}_{sq}$. For reaction automata and their space-bounded subclasses, the following results have been shown in [15, 16].

Proposition 2.7 [15, 16]**.** *The following inclusions hold*:

(1) $\mathcal{REG} = \mathcal{CRA}_{mp} \subset \mathcal{LRA}_{mp} \subset \mathcal{ERA}_{mp} \subset \mathcal{RA}_{mp} = \mathcal{RE}$.
(2) $\mathcal{LIN}$ $(\mathcal{CF})$ *and* $\mathcal{LRA}_{mp}$ *are incomparable.*
(3) $\mathcal{RE} = PR(\mathcal{LRA}_{mp})$.
(4) $\mathcal{CS} = \mathcal{ERA}_{mp}$.

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press

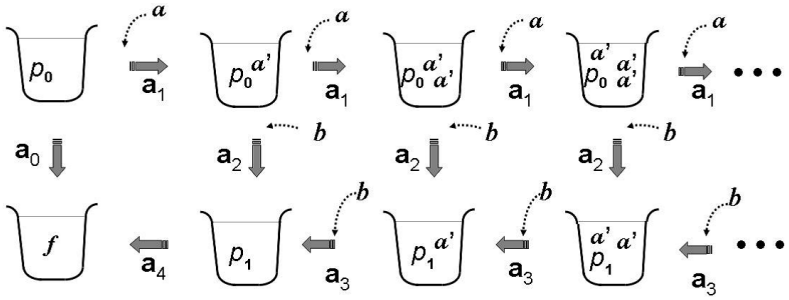


FIGURE 1. A graphic illustration of interactive processes for accepting strings in the language $L = \{a^n b^n \mid n \geq 0\}$ in terms of a reaction automaton $\mathcal{A}$.

Example 2.8. Let us consider a reaction automaton $\mathcal{A} = (S, \Sigma, A, D_0, f)$ defined as follows:

$$
\begin{aligned}
&S = \{p_0, p_1, a, b, a', f\} \text{ with } \Sigma = \{a, b\},\\
&A = \{\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4\}, \text{ where}\\
&\qquad \mathbf{a}_0 = (p_0, aba', f), \ \ \mathbf{a}_1 = (p_0 a, b, p_0 a'), \ \mathbf{a}_2 = (p_0 a' b, \emptyset, p_1),\\
&\qquad \mathbf{a}_3 = (p_1 a' b, a, p_1), \ \mathbf{a}_4 = (p_1, aba', f),\\
&D_0 = p_0.
\end{aligned}
$$

Figure 1 illustrates the whole view of possible interactive processes in $\mathcal{A}$ with inputs $a^n b^n$ for $n \geq 0$. Let $w = aaabbb \in \Sigma^*$ be the input string and consider an interactive process π in sequential manner such that

$$\pi : p_0 \to^a p_0 a' \to^a p_0 a'^2 \to^a p_0 a'^3 \to^b p_1 a'^2 \to^b p_1 a' \to^b p_1 \to f.$$

It can be easily seen that $\pi \in IP_{sq}(\mathcal{A}, w)$ and $w \in L_{sq}(\mathcal{A})$. We may see that $L_{sq}(\mathcal{A}) = \{a^n b^n \mid n \geq 0\}$ which is a context-free language.

We note the following two remarks: (i) this interactive process can be also performed by $\mathcal{A}$ in maximally parallel manner, *i.e.* $\pi \in IP_{mp}(\mathcal{A}, w)$. Moreover, it holds that $L_{mp}(\mathcal{A}) = \{a^n b^n \mid n \geq 0\}$. (ii) The workspace of $\mathcal{A}$ may be bounded by a linear function regarding the length of an input string. Hence, it holds that $\{a^n b^n \mid n \geq 0\} \in \mathcal{LRA}_{sq} \cap \mathcal{LRA}_{mp}$.

2.3. TURING MACHINES AND VARIANTS

In [2], in order to look for a Turing machine corresponding to P automata, a variant of a Turing machine restricted on the workspace, called a restricted $s(n)$ space bounded Turing machine, is introduced. Here, we consider the relaxation of that restriction.

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press

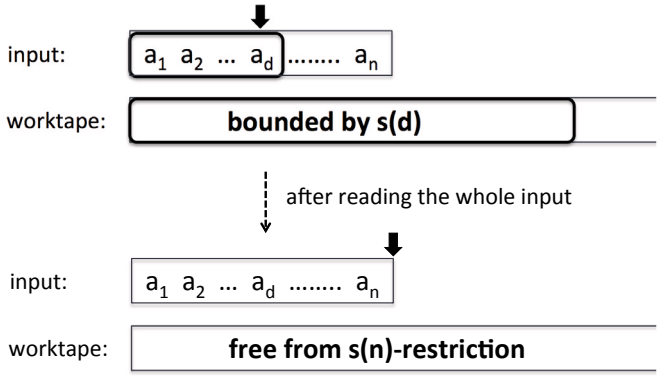


FIGURE 2. The notion of an $s(n)$-restricted Turing machine.

Definition 2.9. A one-way nondeterministic Turing machine is $s(n)$-*restricted* if for every accepted input of length n, there is an accepting computation where the number of cells on the worktape *before reading through the whole input* is bounded by $s(d)$, where d is the number of input tape cells already read.

The difference between "$s(n)$-restricted" and "restricted $s(n)$ space bounded (in [2])" is that for the case "$s(n)$-restricted", a *different restriction* is imposed on the workspace *after reading the whole input*. We say that a one-way nondeterministic Turing machine M is LOG-restricted, LIN-restricted or NON-restricted if M is logarithmic-restricted, linear function-restricted or not restricted, respectively.

Definition 2.10. Let $\mathcal{X}, \mathcal{Y} \in \{LOG, LIN, NON\}$. $\mathcal{L}_1(\mathcal{X}, \mathcal{Y})$ denotes the class of languages accepted by $\mathcal{X}$-restricted $\mathcal{Y}$-space-bounded one-way nondeterministic Turing machines.

Note that (i) $\mathcal{L}_1(NON, \mathcal{Y})$ is the class of language accepted by $\mathcal{Y}$-space-bounded (in usual sense in space complexity theory) one-way nondeterministic Turing machines, (ii) the class of language accepted by restricted $\mathcal{X}$ space bounded one-way nondeterministic Turing machines (defined in [2]) is equivalent to $\mathcal{L}_1(\mathcal{X}, \mathcal{X})$.

We introduce a notation about instantaneous descriptions (IDs) of offline Turing machines.

Definition 2.11. For an offline Turing machine $M = (Q, \Sigma, \Gamma, \delta, p_0, F)$ and an input string w, an ID can be expressed by $(w_1 q w_2, x_1 q x_2)$, where $q \in Q$ is the current state, $w_1 w_2 \in \Sigma^*$ is the input string, $x_1 x_2 \in \Gamma^*$ is the content of the worktape of M, and the heads of M point to the first symbols of w_2 and x_2.

By $ID(M, w)$, we denotes the set of all sequences of the IDs which express computations of M with the input w.

A multicounter machine is a variant of a Turing machine with a one-way read only input tape and several counters. It is known that a two-counter machine is

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press

equivalent to a Turing machine as a language accepting device [10,12]. A k-counter machine is represented by a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$, where, Q is a set of states, Σ is an alphabet of inputs, q_0 is an initial state, F is a set of final states, and δ is a mapping from $Q \times (\Sigma \cup \{\lambda\}) \times \{0,1\}^k$ into $Q \times \{0,+1\} \times \{-1,0,+1\}^k$. A configuration of M on an input w is given by a $(k+3)$-tuple $(q, w, i, c_1, \ldots, c_k)$, where M is in state q with the input head reading the ith symbol of w, and $c_1, c_2, \ldots, c_k$ stored in the k counters. Write $(q, w, i, c_1, \ldots, c_k) \Rightarrow (p, w, i+d, c_1 + d_1, \ldots, c_k + d_k)$ if a is the ith symbol of w and $\delta(q, a, h(c_1), \ldots, h(c_k))$ contains $(p, d, d_1, \ldots, d_k)$, where $h(c_j) = 0$ if $c_j = 0$ and $h(c_j) = 1$ if $c_j \neq 0$. The reflexive-transitive closure of $\Rightarrow$ is written by $\Rightarrow^*$. A language accepted by M is defined as

$$L(M) = \{w \in \Sigma^* \mid (q_0, w, 1, 0, \ldots, 0) \Rightarrow^* (f, w, i, c_1, \ldots, c_k),\ f \in F\}.$$

In [10,12], it is proved that every recursively enumerable language is accepted by a 3-counter machine (and a 2-counter machine), where a 2-stack machine acts as an intermediary machine. In the proof, for $r-1$ tape symbols used by the stack machine, the contents of the 2 stacks $X_1 \ldots X_i$ and $Y_1 \ldots Y_j$ are regarded as an integer in base r. They are simulated by the 2 counters as $X_i r^{i-1} + X_{i-1} r^{i-2} + \ldots + X_1$ and $Y_j r^{j-1} + Y_{j-1} r^{j-2} + \ldots + Y_1$. The third counter is used to adjust the other 2 counters. It can be observed that a 2-stack machine using $s(n)$ workspace is simulated by a 3-counter machine where the values of counters are bounded by $r^{s(n)}$.

On the other hand, from the proof of Theorem 8.13 in [12], it obviously holds that any $s(n)$-bounded 1-way Turing machine can be simulated by a 2-stack machine using $s(n)$ workspace.

These facts imply the following proposition.

Proposition 2.12. *If a language L is accepted by a $\log s(n)$-bounded one-way TM, then L is accepted by a 3-counter machine where the values of counters are bounded by $s(n)$.*

3. Main results

Theorem 3.1. *A language L is accepted by an $s(n)$-bounded RA with λ-input mode in sequential manner if and only if L is accepted by a $\log s(n)$-bounded one-way TM.*

Proof. ("if" part) For a k-counter machine $M = (Q, \Sigma, \delta, q_0, F)$, construct an RA $\mathcal{A}_M = (S, \Sigma, A, q_0, f')$, where $S = Q \cup \{f', e_1, \ldots, e_k\}$ and A is defined as follows:

(1) for any $f \in F$, $(f, \emptyset, f')$ is in A;
(2) If $(p, d, d_1, \ldots, d_k)$ is contained in $\delta(q, a, h(c_1), \ldots, h(c_k))$ of M, then

$$(qas_1 \ldots s_n,\ t_1 \ldots t_m f,\ ps_1 \ldots s_n + u_1 \ldots u_r - v_1 \ldots v_l) \in A,$$

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press

where

- $\{e_i \in S \,|\, 1 \leq i \leq k,\ h(c_i) = 1\} = \{s_1, \ldots, s_n\}$;
- $\{e_i \in S \,|\, 1 \leq i \leq k,\ h(c_i) = 0\} = \{t_1, \ldots, t_m\}$;
- $\{e_i \in S \,|\, 1 \leq i \leq k,\ d_i = +1\} = \{u_1, \ldots, u_r\}$;
- $\{e_i \in S \,|\, 1 \leq i \leq k,\ d_i = -1\} = \{v_1, \ldots, v_l\}$.

From the way of construction of $\mathcal{A}_M$, it is easily confirmed that for the input w, there is a configuration $(q, w, i, c_1, \ldots, c_k)$ in M if and only if there is an interactive process $\pi = D_0, \ldots, D_{i'}, \ldots$ in $\mathcal{A}_M$ such that $D_{i'} = qe_1^{c_1} \ldots e_k^{c_k}$ after reading the ith symbol of w. Hence, it holds that $L(M) = L(\mathcal{A}_M)$. Note that if the values of counters of M are bounded by $s(n)$, the workspace of $\mathcal{A}_M$ is also bounded by $s(n)$.

From Proposition 2.12, if a language L is accepted by a $\log s(n)$-bounded one-way TM, then L is accepted by a 3-counter machine where the values of counters are bounded by $s(n)$. Hence, L is accepted by an $s(n)$-bounded RA with λ-input mode in sequential manner.

("only if" part) The proof is almost same to the one of Theorem 8 in [16], however, we show it for the proof of the following corollary.

Let $S = \{s_1, \ldots, s_k\}$ be an ordered alphabet and $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be an RA. Assume that for an input $w = a_1 \ldots a_n$ the workspace of $\mathcal{A}$ is bounded by the function $s(n)$. Then, we shall construct the nondeterministic $(k+2)$-tape Turing machine $M_{\mathcal{A}}$. $M_{\mathcal{A}}$ imitates an interactive process $\pi : D_0, \ldots, D_n, \ldots \in IP^{\lambda}_{sq}(\mathcal{A}, w)$ in the following manner:

1. At first, Tape-1 has the input $w \in \Sigma^*$ and Tape-$(i+1)$ has the number of s_i in D_0 (for $1 \leq i \leq k$) represented by the binary number. Tape-$(k+2)$ is used to count the number of computation step of $M_{\mathcal{A}}$.
2. Let D be the current multiset in π. When $M_{\mathcal{A}}$ reads the symbol s_i in the input, add one to the Tape-$(i+1)$. Then, by checking all tapes except Tape-1, compute an element of $\mathrm{Res}^{sq}_A(s_i + D)$ in the nondeterministic way and rewrite the contents in the tapes. After reading through the input w, $M_{\mathcal{A}}$ computes an element of $\mathrm{Res}^{sq}_A(D)$ in the nondeterministic way and rewrite the contents in the tapes.
3. After reading through the input w, if $\mathrm{Res}^{sq}_A(D) = \{D\}$ and $f \subseteq D$, then $M_{\mathcal{A}}$ accepts w. In the case where (i) $\mathrm{Res}^{sq}_A(D) = \{D\}$ and $f \not\subseteq D$; (ii) $|D|$ exceeds $s(n)$ or (iii) the number of computation step exceeds $c(s(n))^k$ for $k(= |S|)$ and some constant c, $M_{\mathcal{A}}$ rejects w.

Since we use the binary number for counting the number of symbols, the maximum length of each tape to memorize D is $\log_2 s(n)$. In the case where $M_{\mathcal{A}}$ never stops with the input w, there exists a cycle of configurations in the computation. Since the number of all possible Ds during the computation is bounded by $c(s(n))^k$ for k and some constant c, the length of Tape-$(k+2)$ to count the number of steps of computation is bounded by $\log_2 c + k \log_2 s(n)$. Therefore, it holds that $L(M_{\mathcal{A}}) = L^{\lambda}_{sq}(\mathcal{A})$ and the workspace of $M_{\mathcal{A}}$ is bounded by the function regarding $\log_2 s(n)$. □

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press

Corollary 3.2. *The following equations hold:*
(1) $\mathcal{RA}_{sq}^{\lambda} = \mathcal{RE}$.
(2) $\mathcal{ERA}_{sq}^{\lambda} = \mathcal{CS}$.

Corollary 3.3. $PR(\mathcal{RA}_{sq}) = \mathcal{RE}$.

Proof. When λ is inputted to an RA $\mathcal{A}_M = (S, \Sigma, A, D_0, f)$ with λ-input mode, we consider that $\mathcal{A}'_M = (S \cup \{c\}, \Sigma \cup \{c\}, A, D_0, f)$ and the special symbol $c \notin S$ is inputted instead of λ. From the proof of Theorem 3.1, it obviously holds that $w = a_1 a_2 \dots a_l \in L_{sq}^{\lambda}(\mathcal{A}_M)$ if and only if $w' = c^{i_0} a_1 c^{i_1} a_2 c^{i_2} \dots a_l c^{i_l} \in L_{sq}(\mathcal{A}'_M)$ for some $i_0, i_1, i_2, \dots, i_l \geq 0$.

Using a projection $h : \Sigma \cup \{c\} \to \Sigma$ which removes c, it is obtained that $L_{sq}^{\lambda}(\mathcal{A}_M) = h(L_{sq}(\mathcal{A}'_M))$. Hence, it holds that $\mathcal{RA}_{sq}^{\lambda} = \mathcal{RE} \subseteq PR(\mathcal{RA}_{sq})$. The other inclusion is straightforward. □

Corollary 3.4. $\mathcal{RA}_{sq} \subseteq \mathcal{L}_1(LOG, NON)$.

Proof. Let $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be an RA and $\pi = D_0, D_1, \dots \in IP_{sq}(\mathcal{A}, w)$ for the input $w \in \Sigma^*$ with $|w| = n$. By the same way of the proof of "only if" part of Theorem 3.1, we construct $M_{\mathcal{A}}$.

Then, it holds that $|D_i| \leq ci + |D_0|$, where $c = max_{a \in A}(|P_a - R_a|)$ and $1 \leq i \leq n$. We can easily confirm that the workspace of $M_{\mathcal{A}}$ after reading i symbols of w is bounded by $\log(ci)$. Hence, $\mathcal{A}$ can be simulated by $M_{\mathcal{A}}$ with $L(M_{\mathcal{A}}) \in \mathcal{L}_1(LOG, NON)$. □

There are many related works on language acceptors based on multiset rewriting, such as a variant of P systems, called P automata investigated in the literature (*e.g.*, [2–4]). A P automaton is a finite automata-like computing model in which a configuration comprises a tuple of multisets each of which consists of objects from each membrane region. On receiving an input (a multiset) from the environment at each step of computation, it changes its configuration by making region-wise applications of the equipped rules. An input sequence of multisets is accepted if the transition reaches a final state after reading the whole input, and the language accepted by a P automaton is defined as a mapping image of those accepted multiset sequences. In this sense, reaction automata may also be regarded as a simplified variants of P automata with no membrane structure.

Let us denote the class of languages accepted by P automata with sequential rule applications by $\mathcal{PA}_{sq}$. In [2], it is proved that $\mathcal{L}_1(LOG, LOG) = \mathcal{PA}_{sq}$. Hence, the following corollary holds from Corollary 3.4.

Corollary 3.5. $\mathcal{LRA}_{sq} \subseteq \mathcal{L}_1(LOG, LOG) = \mathcal{PA}_{sq}$.

Next, we consider a necessary condition for a language to be in $\mathcal{L}_1(LOG, NON)$ and in $\mathcal{RA}_{sq}$. Let Σ be an alphabet with $|\Sigma| \geq 2$ and $h : \Sigma^* \to \Sigma^*$ be an injection. Then, the following lemma follows.

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press

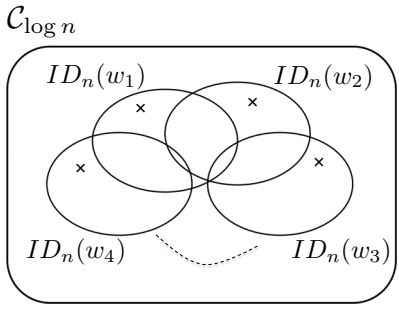


FIGURE 3. Proof sketch of Lemma 1.

Lemma 3.6. *It holds that* $\{wh(w) \mid w \in \Sigma^*\} \notin \mathcal{L}_1(LOG, NON)$.

Proof. Assume that there is a $\log n$-restricted 1-way TM $M = (Q, \Gamma, \Sigma, q_0, F, \delta)$ such that $L(M) = \{wh(w) \mid w \in \Sigma^*\}$. Let $|Q| = m_1$, $|\Gamma| = m_2$, $|\Sigma| = m_3 \geq 2$ and the input string be $wh(w)$ with $|w| = n$.

We define $\mathcal{C}_{\log n}$ as the set of all possible IDs of M before reading the input string w. Recall that an ID of M is expressed as (w_1qw_2, x_1qx_2) with $w = w_1w_2$. In the case where $|w_1| = i$, the number of cases of the contents of the worktape x_1x_2 is bounded by $(m_2)^{\log i}$ and the number of cases of the position of the head for the worktape is bounded by $\log i + 1$. Hence, it holds that

$$\begin{aligned} |\mathcal{C}_{\log n}| &\leq \sum_{i=1}^{n} (m_1 \cdot (m_2)^{\log i} \cdot (\log i + 1)) \\ &\leq n \cdot m_1 \cdot (m_2)^{\log n} \cdot (\log n + 1). \end{aligned}$$

Since it holds that $|\Sigma^n| = (m_3)^n$, if n is sufficiently large, we obtain the inequality $|\mathcal{C}_{\log n}| < |\Sigma^n|$.

For $w \in \Sigma^*$, let $ID_n(w) = \{C_n \in \mathcal{C}_{\log n} \mid \pi = C_0, \ldots, C_n, \ldots \in ID(M, w)\}$, *i.e.*, $ID_n(w)$ is the set of IDs which appear as the nth elements of sequences in $ID(M, w)$. From the assumption that $L(M) = \{wh(w) \mid w \in \Sigma^*\}$ and h is an injection, we can show that for any two distinct strings $w_1, w_2 \in \Sigma^n$, $ID_n(w_1)$ and $ID_n(w_2)$ are incomparable. This is because if $ID_n(w_1) \subseteq ID_n(w_2)$, then the string $w_2h(w_1)$ is accepted by M, which means that $h(w_1) = h(w_2)$ and contradicts that h is an injection.

Since for any two distinct strings $w_1, w_2 \in \Sigma^n$, $ID_n(w_1)$ and $ID_n(w_2)$ are incomparable and $ID_n(w_1), ID_n(w_2) \subseteq \mathcal{C}_{\log n}$, it holds the following inequality (see Fig. 3):

$$|\{ID_n(w) \mid w \in \Sigma^n\}| \leq |\mathcal{C}_{\log n}| < |\Sigma^n|.$$

However, the inequality $|\{ID_n(w) \mid w \in \Sigma^n\}| < |\Sigma^n|$ contradicts that for any two distinct strings $w_1, w_2 \in \Sigma^n$, it holds that $ID_n(w_1) \neq ID_n(w_2)$. □

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press

Corollary 3.7. *It holds that* $\{wh(w) \mid w \in \Sigma^*\} \notin \mathcal{RA}_{sq}$.

Then, we consider the relation between the language classes accepted by RAs in maximally parallel manner and ones in sequential manner. For the sake of comparing the classes of languages $\mathcal{LRA}_{mp}$ and $\mathcal{RA}_{sq}$, remind the following propositions shown in [15, 16].

Proposition 3.8 [16]**.** *For any context-sensitive language* $L \subseteq \Sigma^*$*, there exists a linear-bounded RA* $\mathcal{A}$ *such that* $w \in L$ *if and only if* $c^{2^n}w \in L_{mp}(\mathcal{A})$ *(or* $c^{2^n}w \in L_{mp}(\mathcal{A})$*) with* $|w| = n$ *and* $c \notin \Sigma$.

Proposition 3.9 [15]**.** *It holds that* $\{ww^R \mid w \in \Sigma^*\} \notin \mathcal{LRA}_{mp}$.

Theorem 3.10. $\mathcal{LRA}_{mp}$*,* $\mathcal{RA}_{sq}$ *and* $\mathcal{CF}$ *are pairwise incomparable.*

Proof. $(\mathcal{LRA}_{mp} - (\mathcal{RA}_{sq} \cup \mathcal{CF}) \neq \emptyset)$ From Proposition 3.8, it holds that $L = \{wwc^{2^{2n}} \mid w \in \Sigma^*, |w| = n\} \in \mathcal{LRA}_{mp}$. Let h be an injection such that $h(w) = wc^{2^{2n}}$ with $|w| = n$. On the other hand, from Corollary 3.7 it obviously holds that $L \notin \mathcal{RA}_{sq} \cup \mathcal{CF}$.

$(\mathcal{RA}_{sq} - (\mathcal{LRA}_{mp} \cup \mathcal{CF}) \neq \emptyset)$ Let $\mathcal{CM}$ be the class of all commutative languages. Using a vector of natural numbers V, a commutative language L can be written as $L = \{w \in \Sigma^* \mid \psi(w) \in V\}$. For $w \in L$, let us consider a computation of a k-counter machine M accepting L such that after storing $\psi(w)$ in the k counters in the first $|w|$ steps, M confirms that $\psi(w) \in V$. This computation can be simulated by an RA $\mathcal{A}_M$ with ordinary input mode, because

- in the first $|w|$ steps, storing the number of each symbol appearing in w can realize by $\mathcal{A}_M$ without λ-move,
- after these steps, $\mathcal{A}_M$ can simulate M as the proof of Theorem 1.

Hence, it holds that $\mathcal{CM} \subset \mathcal{RA}_{sq}$.

On the other hand, $\mathcal{CM}$ and $\mathcal{CS}(\supset \mathcal{LRA}_{mp})$ is obviously incomparable. Therefore, it holds that $\mathcal{RA}_{sq} - (\mathcal{LRA}_{mp} \cup \mathcal{CF}) \neq \emptyset$.
$(\mathcal{CF} - (\mathcal{LRA}_{mp} \cup \mathcal{RA}_{sq}) \neq \emptyset)$ From Corollary 3.7 and Proposition 3.9, it holds that $\{ww^R \mid w \in \{a, b\}^*\} \in \mathcal{CF} - (\mathcal{LRA}_{mp} \cup \mathcal{RA}_{sq})$. □

Corollary 3.11. *It holds that* $\mathcal{LRA}_{sq} \subset \mathcal{RA}_{sq}$ *and* $\mathcal{LRA}_{sq} \subset \mathcal{LRA}_{mp}$.

Proof. From the definition, it is obviously holds that $\mathcal{LRA}_{sq} \subseteq \mathcal{RA}_{sq}$. For the proof of $\mathcal{LRA}_{sq} \subseteq \mathcal{LRA}_{mp}$, let $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be a linear-bounded RA. Construct a linear-bounded RA $\mathcal{A}' = (S \cup \{s\}, \Sigma, A', D_0 \cup \{s\}, f)$, where

$$A' = \{a' = (R + s, I, P + s) \mid a = (R, I, P) \in A\}.$$

Then, it holds that $L_{sq}(\mathcal{A}) = L_{mp}(\mathcal{A}')$ and $\mathcal{LRA}_{sq} \subseteq \mathcal{LRA}_{mp}$. Using Theorem 3.10, it is shown that $\mathcal{LRA}_{sq} \subset \mathcal{RA}_{sq}$ and $\mathcal{LRA}_{sq} \subset \mathcal{LRA}_{mp}$. □

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press

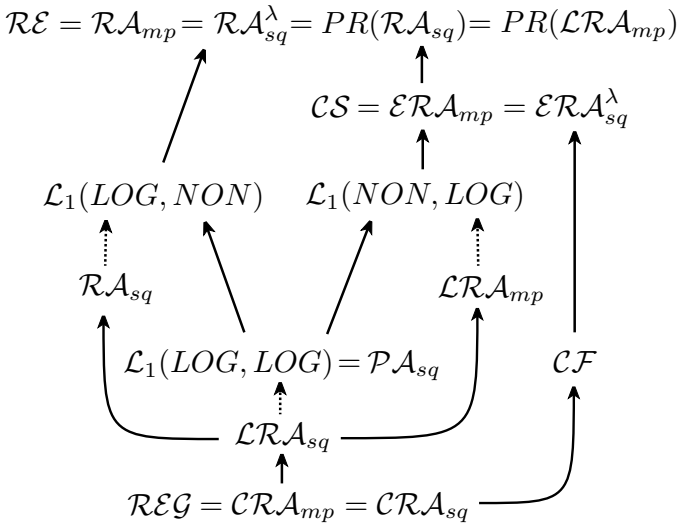


FIGURE 4. The diagram of the relation between the language classes regarding RA. A proper inclusion relation is denoted by a solid line and an inclusion relation is denoted by a broken line.

4. Conclusion

Based on the formal framework of reaction systems [5], reaction automata (RAs) were introduced in [15] as language acceptors with multiset rewriting mechanism. In this paper, we have continued the investigation of RAs with a focus on the two ways of rule applications, maximally parallel manner and sequential manner. Considering some restrictions on the workspace and λ-input mode, we have introduced the classes of languages accepted by the variants of RAs, such as $\mathcal{RA}_X$, $\mathcal{RA}^{\lambda}_X$, $\mathcal{LRA}_X$, $\mathcal{LRA}^{\lambda}_X$ with $X \in \{sq, mp\}$, and investigated the computation powers of them. In order to explore Turing machines (TMs) corresponding to those classes of RAs, we have also introduced a new variant of TMs with restricted workspace, called $s(n)$-restricted TMs.

Figure 4 summarizes the relationship among the classes of languages accepted by various types of RAs, TMs and the Chomsky hierarchy. Specifically, we have shown the following:

- a language L is accepted by an $s(n)$-bounded RA with λ-input mode in sequential manner if and only if L is accepted by an $\log s(n)$-bounded one-way TM;
- any recursively enumerable language can be expressed as a homomorphic image of a language in $\mathcal{RA}_{sq}$;
- the class of languages $\mathcal{LRA}_{sq}$ is included in the class of languages accepted by P automata in sequential manner;

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press

- the three classes of languages $\mathcal{LRA}_{mp}$, $\mathcal{RA}_{sq}$ and $\mathcal{CF}$ are pairwise incomparable.

There are several subjects remaining to be investigated. First, it is open whether or not the following proper inclusion relations hold:

- $\mathcal{LRA}_{mp} \subset \mathcal{L}_1(NON, LOG)$;
- $\mathcal{LRA}_{sq} \subset \mathcal{L}_1(LOG, LOG)$;
- $\mathcal{RA}_{sq} \subset \mathcal{L}_1(LOG, NON)$.

Secondly, to explore the computation powers of deterministic reaction automata and time-bounded reaction automata is open and important issues. Lastly, it would be useful to develop a method for simulating a variety of chemical reactions in the real world by the use of the framework based on reaction automata.

Acknowledgements. The author is grateful to Takashi Yokomori for helpful discussions which improved the paper. The author gratefully acknowledges the anonymous referee of this paper for helpful suggestions. Especially, the proof of Theorem 1 has been greatly simplified by the useful comments.

References

[1] C. Calude, Gh. Păun, G. Rozenberg and A. Salomaa, *Multiset Processing.* In vol. 2235 of *Lect. Notes Comput. Sci.* Springer (2001).
[2] E. Csuhaj-Varjú, O.H. Ibarra and Gy. Vaszil, On the computational complexity of P automata. *Nat. Comput.* **5** (2006) 109–126.
[3] E. Csuhaj-Varjú, M. Oswald and Gy. Vaszil, P automata, in *The Oxford Handbook of Membrane Computing* (2010) 145–167.
[4] E. Csuhaj-Varjú and Gy. Vaszil, P automata or purely communicating accepting P systems. In vol. 2597 of *Lect. Notes Comput. Sci.* Springer (2003) 219–233.
[5] A. Ehrenfeucht and G. Rozenberg, Reaction systems. *Fund. Inform.* **75** (2007) 263–280.
[6] A. Ehrenfeucht and G. Rozenberg, Events and modules in reaction systems. *Theoret. Comput. Sci.* **376** (2007) 3–16.
[7] A. Ehrenfeucht and G. Rozenberg, Introducing time in reaction systems. *Theoret. Comput. Sci.* **410** (2009) 310–322.
[8] A. Ehrenfeucht, M. Main and G. Rozenberg, Combinatorics of life and death in reaction systems. *Int. J. Found. Comput. Sci.* **21** (2010) 345–356.
[9] A. Ehrenfeucht, M. Main and G. Rozenberg, Functions defined by reaction systems. *Int. J. Found. Comput. Sci.* **22** (2011) 167–178.
[10] P.C. Fischer, Turing Machines with Restricted Memory Access. *Inform. Control* **9** (1966) 364–379.
[11] M. Hirvensalo, On probabilistic and quantum reaction systems. *Theoret. Comput. Sci.* **429** (2012) 134–143.
[12] J.E. Hopcroft, T. Motwani and J.D. Ullman, *Introduction to automata theory, language and computation,* 2nd edition. Addison-Wesley (2003).
[13] M. Kudlek, C. Martin-Vide and Gh. Păun, Toward a formal macroset theory, in *Multiset Processing*, vol. 2235 of *Lect. Notes Comput. Sci.*, edited by C. Calude, Gh. Păun, G. Rozenberg and A. Salomaa. Springer (2001) 123–134.
[14] F. Okubo, On the Computational Power of Reaction Automata Working in Sequential Manner, in *Proc. of 4th Workshop on Non-Classical Models for Automata and Applications*, vol. 290 of book@ocg.at series. Öesterreichische Comput. Gesellschaft (2012) 149–164.

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press

[15] F. Okubo, S. Kobayashi and T. Yokomori, Reaction Automata. *Theoret. Comput. Sci.* **429** (2012) 247–257.
[16] F. Okubo, S. Kobayashi and T. Yokomori, On the Properties of Language Classes Defined by Bounded Reaction Automata. *Theoret. Comput. Sci.* **454** (2012) 206–221.
[17] A. Salomaa, *Formal Languages*. Academic Press, New York (1973).
[18] A. Salomaa, Functions and sequences generated by reaction systems. *Theoret. Comput. Sci.* **466** (2012) 87–96.

Communicated by M. Holzer.
Received January 29, 2013. Accepted December 11, 2013.

https://doi.org/10.1051/ita/2013047 Published online by Cambridge University Press