Theoretical Computer Science 429 (2012) 247–257

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Reaction automata

Fumiya Okubo [a], Satoshi Kobayashi [b], Takashi Yokomori [c,*]

[a] Graduate School of Education, Waseda University, 1-6-1 Nishiwaseda, Shinjuku-ku, Tokyo 169-8050, Japan

[b] Graduate School of Informatics and Engineering, University of Electro-Communications, 1-5-1 Chofugaoka, Chofu-shi, Tokyo 182-8585, Japan

[c] Department of Mathematics, Faculty of Education and Integrated Arts and Sciences, Waseda University, 1-6-1 Nishiwaseda, Shinjuku-ku, Tokyo 169-8050, Japan

ARTICLE INFO

Keywords:
Models of biochemical reactions
Reaction automata
Turing computability

ABSTRACT

Reaction systems are a formal model that has been introduced to investigate the interactive behaviors of biochemical reactions. Based on the formal framework of reaction systems, we propose new computing models called *reaction automata* that feature (string) language acceptors with multiset manipulation as a computing mechanism, and show that reaction automata are computationally Turing universal. Further, some subclasses of reaction automata with space complexity are investigated and their language classes are compared to the ones in the Chomsky hierarchy.

1. Introduction

In recent years, a series of seminal papers [7–9] has been published in which Ehrenfeucht and Rozenberg have introduced a formal model, called *reaction systems*, for investigating interactions between biochemical reactions, where two basic components (reactants and inhibitors) are employed as regulation mechanisms for controlling biochemical functionalities. It has been shown that reaction systems provide a formal framework best suited for investigating in an abstract level the way of emergence and evolution of biochemical functioning such as events and modules. In the same framework, they also introduced the notion of time into reaction systems and investigated notions such as reaction times, creation times of compounds and so forth. Two rather recent papers [10,11] continue the investigation of reaction systems, with the focuses on combinatorial properties of functions defined by random reaction systems and on the dependency relation between the power of defining functions and the amount of available resource.

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 $res_a(T)$, is given by P_a if a is enabled by T (i.e., if T completely includes R_a and excludes I_a). Otherwise, the result is empty. Thus, $res_a(T) = P_a$ if a is enabled on T, and $res_a(T) = \emptyset$ otherwise. The result of applying a reaction a is extended to the set of reactions A, denoted by $res_A(T)$, and an interactive process consisting of a sequence of $res_A(T)$'s is properly introduced and investigated.

In the last few decades, the notion of a multiset has frequently appeared and been investigated in many different areas such as mathematics, computer science, linguistics, and so forth. (See, e.g., [2] for the reference papers written from the viewpoint of mathematics and computer science.) The notion of a multiset has received more and more attention, particularly in the areas of biochemical computing and molecular computing (e.g., [19,23]).

* Corresponding author.
E-mail addresses: f.okubo@akane.waseda.jp (F. Okubo), satoshi@cs.uec.ac.jp (S. Kobayashi), yokomori@waseda.jp (T. Yokomori).

0304-3975/$ – see front matter © 2011 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2011.12.045

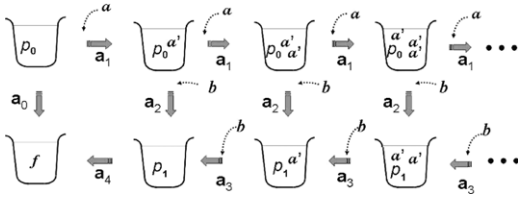

Fig. 1. A graphic illustration of interactive biochemical reaction processes for accepting strings in the language $L = \{a^n b^n \mid n \geq 0\}$ in terms of our reaction automaton $\mathcal{A}$.

Motivated by these two notions of a reaction system and a multiset, in this paper we will introduce computing devices called *reaction automata* and show that they are computationally universal by proving that any recursively enumerable language is accepted by a reaction automaton. There are two points to be remarked: on one hand, the notion of reaction automata may be taken as a kind of an extension of reaction systems in the sense that our reaction automata deal with *multisets* rather than (usual) sets as reaction systems do, in the sequence of computational process. On the other hand, however, reaction automata are introduced as computing devices that accept the sets of *string objects* (i.e., languages over an alphabet). This unique feature, i.e., a string accepting device based on multiset computing in the biochemical reaction model can be realized by introducing a simple idea of feeding an input to the device from the environment and by employing a special encoding technique.

In order to illustrate an intuitive idea of the notion of reaction automata and their behavior, we give in Fig. 1 a simple example of the behavior of a reaction automata $\mathcal{A}$ that consists of the set of objects $\{p_0, p_1, a, b, a', f\}$ (with the input alphabet $\{a, b\}$), the set of reactions $\{\mathbf{a}_0 = (p_0, \{a, b, a'\}, f), \mathbf{a}_1 = (p_0 a, \{b\}, p_0 a'), \mathbf{a}_2 = (p_0 a' b, \emptyset, p_1), \mathbf{a}_3 = (p_1 a' b, \{a\}, p_1), \mathbf{a}_4 = (p_1, \{a, b, a'\}, f)\}$, where $\{p_0\}$ is the initial multiset and $\{f\}$ is the final multiset. Note that in a reaction $\mathbf{a} = (R_a, I_a, P_a)$, multisets R_a and P_a are represented by string forms, while I_a is given as a set. In the graphic drawing of Fig. 1, each reaction $\mathbf{a}_i$ is applied to a multiset (of a test tube) after receiving an input symbol (if any is provided) from the environment. In particular, applying $\mathbf{a}_0$ to $\{p_0\}$ leads to that the empty string is accepted by $\mathcal{A}$. It is seen, for example, that reactions $\mathbf{a}_1$ and $\mathbf{a}_2$ are enabled by the multiset $T = \{p_0, a', a'\}$ only when inputs a and b, respectively, are received, which result in producing $R_1 = \{p_0, a', a', a'\}$ and $R_2 = \{p_1, a'\}$, respectively. Thus, we have that $res_{\mathbf{a}_1}(T \cup \{a\}) = R_1$ and $res_{\mathbf{a}_2}(T \cup \{b\}) = R_2$. Once applying $\mathbf{a}_2$ has brought about a change of p_0 into p_1, $\mathcal{A}$ has no possibility of accepting further inputs a's, because of the inhibitors in $\mathbf{a}_3$ or $\mathbf{a}_4$. One may easily see that $\mathcal{A}$ accepts the language $L = \{a^n b^n \mid n \geq 0\}$. We remark that reaction automata allow a multiset of reactions α to apply to a multiset of objects T in an exhaustive manner (what we call a *maximally parallel manner*), and therefore the interactive process sequence of computation is nondeterministic in that the reaction result from T may produce more than one product. The details for these are formally described in the sequel.

This paper is organized as follows. After preparing the basic notions and notations from formal language theory in Section 2, we formally introduce the main notion of reaction automata together with one language example in Section 3. Then, Section 4 describes a multistack machine (in fact, a two-stack machine) whose specific property will be demonstrated to be very useful in the proof of the main result in the next section. Thus, in Section 5 we present our main results: reaction automata are computationally universal. We also consider some subclasses of reaction automata from a viewpoint of the complexity theory in Section 6, and investigate the language classes accepted by those subclasses in comparison to the Chomsky hierarchy. Finally, concluding remarks as well as future research topics are briefly discussed in Section 7.

2. Preliminaries

We assume that the reader is familiar with the basic notions of formal language theory. For unexplained details, refer to [13].

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 $k \geq 0$, let $pref_k(x)$ be the prefix of a string x of length k. For a string $w = a_1 a_2 \cdots a_n \in V^*$, w^R is the reversal of w, that is, $(a_1 a_2 \cdots a_n)^R = a_n \cdots a_2 a_1$. Further, for a string $x = a_1 a_2 \cdots a_n \in V^*$, $\hat{x}$ denotes the hat version of x, i.e., $\hat{x} = \hat{a_1}\hat{a_2} \cdots \hat{a_n}$, where each $\hat{a}_i$ is in an alphabet $\hat{V} = \{\hat{a} \mid a \in V\}$ such that $V \cap \hat{V} = \emptyset$.

We use the basic notations and definitions regarding multisets that follow [4,15]. 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 $a \in V$. A multiset μ may be represented as a vector, $\mu(V) = (\mu(a_1), \ldots, \mu(a_n))$, for an ordered set $V = \{a_1, \ldots, a_n\}$. We can also represent the multiset μ by any permutation of the string $w_\mu = a_1^{\mu(a_1)} \cdots a_n^{\mu(a_n)}$. Conversely, 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 three 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 $a \in V$,
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 $\mu_2 \subseteq \mu_1$ only).

A multiset μ_1 is called *multisubset* of μ_2 if $\mu_1 \subseteq \mu_2$. The sum for a family of multisets $\mathcal{M} = \{\mu_i\}_{i \in I}$ is also denoted by $\sum_{i \in I} \mu_i$. 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 $|\mu| = \sum_{a \in V} \mu(a)$.

We introduce an injective function $stm : V^* \to V^\#$ that maps a string to a multiset in the following manner:

$$\begin{cases} stm(a_1 a_2 \cdots a_n) = a_1 a_2^2 \cdots a_n^{2^{n-1}} & (\text{for } n \geq 1) \\ stm(\lambda) = \lambda. \end{cases}$$

3. Reaction automata

As is previously mentioned, a novel formal model called reaction systems has been introduced in order to investigate the property of interactions between biochemical reactions, where two basic components (reactants and inhibitors) are employed as regulation mechanisms for controlling biochemical functionalities [7–9]. Reaction systems provide a formal framework best suited for investigating the way of emergence and evolution of biochemical functioning on an abstract level.

By recalling from [7] basic notions related to reactions systems, we first extend them (defined on the sets) to the notions on the multisets. Then, we shall introduce our notion of *reaction automata* which plays a central role in this paper.

Definition 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 $R_\mathbf{a}, P_\mathbf{a} \in S^\#$, $I_\mathbf{a} \subseteq S$ and $R_\mathbf{a} \cap I_\mathbf{a} = \emptyset$.

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

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

In what follows, we usually identify the set of reactions A with the set of labels $Lab(A)$ of reactions in A, and often use the symbol A as a finite alphabet.

Definition 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 maximally parallel manner* if there is no $\beta \in A^\#$ such that $\alpha \subset \beta$, and α and β are enabled by T.
(3) By $En_A^p(T)$ we denote the set of all multisets of reactions $\alpha \in A^\#$ which are enabled by T in maximally parallel manner.
(4) The *results of* A *on* T, denoted by $Res_A(T)$, is defined as follows:

$$Res_A(T) = \{T - R_\alpha + P_\alpha \mid \alpha \in En_A^p(T)\}.$$

Note that we have $Res_A(T) = \{T\}$ if $En_A^p(T) = \emptyset$. Thus, if no multiset of reactions $\alpha \in A^\#$ is enabled by T in maximally parallel manner, then T remains unchanged.

Remarks 1. (i) It should be also noted that the definition of the results of A on T (given in (4) above) is in contrast to the original one in [7], because we adopt the assumption of *permanency of elements*: any element that is not a reactant for any active reaction *does* remain in the result after the reaction.
(ii) In general, $En_A^p(T)$ may contain more than one element, and therefore, so may $Res_A(T)$.
(iii) For simplicity, I_a is often represented as a string rather than a set.

Example 1. Let $S = \{a, b, c, d, e\}$ and consider the following set $A = \{\mathbf{a}, \mathbf{b}, \mathbf{c}\}$ of reactions in S:

$$\mathbf{a} = (b^2, a, c), \qquad \mathbf{b} = (c^2, \emptyset, b), \qquad \mathbf{c} = (bc, d, e).$$

(i) Consider a finite multiset $T = b^4cd$. Then, $\alpha_1 = \mathbf{a}$ is enabled by T, while neither $\mathbf{b}$ nor $\mathbf{c}$ is enabled by T, because $R_\mathbf{b} \not\subseteq T$ and $I_\mathbf{c} \cap T \neq \emptyset$. Further, $\alpha_2 = \mathbf{a}^2$ is not only enabled by T but also enabled by T in maximally parallel manner, because no β with $\alpha_2 \subset \beta$ is enabled by T. Since $R_{\mathbf{a}^2} = b^4$, $P_{\mathbf{a}^2} = c^2$, and $En_A^p(T) = \{\mathbf{a}^2\}$, we have

$$Res_A(T) = \{T - R_{\mathbf{a}^2} + P_{\mathbf{a}^2}\} = \{c^3 d\}.$$

(ii) Consider $T' = b^3c^2e$. Then, $\beta_1 = \mathbf{ab}$ and $\beta_2 = \mathbf{ac}$ are enabled by T', while $\mathbf{bc}$ is not. Further, it is seen that both β_1 and β_2 are enabled by T' in maximally parallel manner, and $En_A^p(T') = \{\mathbf{ab}, \mathbf{ac}\}$. Thus, we have

$$Res_A(T') = \{b^2ce, c^2e^2\}.$$

If we take $T'' = bcd$, then none of the reactions from A is enabled by T''. Therefore, we have $Res_A(T'') = T''$.

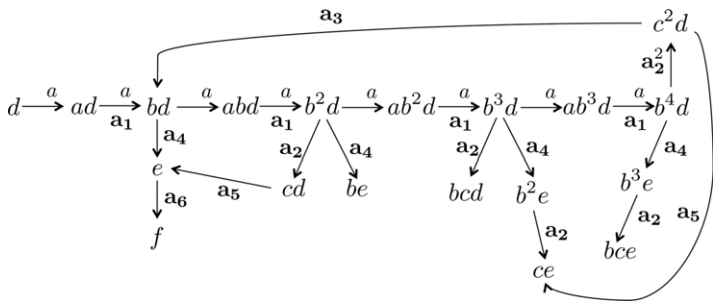

Fig. 2. (a) Reaction diagram: interactive processes for accepting a^2, a^4 and a^8 in $\mathcal{A}$. Some arrows are associated with a multiset of reactions applied at the step.

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

Definition 3 (*Reaction Automata*)**.** 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 4. Let $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be an RA and $w = a_1 \cdots a_n \in \Sigma^*$. An *interactive process in* $\mathcal{A}$ *with input* w is an infinite sequence $\pi = D_0, \ldots, D_i, \ldots$, where

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

By $IP(\mathcal{A}, w)$ we denote the set of all interactive processes in $\mathcal{A}$ with input w.

In order to represent an interactive process π, we also use the "arrow notation" for $\pi : (a_1, D_0) \to \cdots \to (a_n, D_{n-1}) \to (D_n) \to (D_{n+1}) \to \cdots$, or alternatively, $D_0 \to^{a_1} D_1 \to^{a_2} D_2 \to^{a_3} \cdots \to^{a_{n-1}} D_{n-1} \to^{a_n} D_n \to D_{n+1} \to \cdots$.

For an interactive process π in $\mathcal{A}$ with input w, if $En^p_A(D_m) = \emptyset$ for some $m \ge |w|$, then we have that $Res_A(D_m) = \{D_m\}$ and $D_m = D_{m+1} = \cdots$. In this case, considering the smallest m, we say that π *converges on* D_m (at the m-th step). When an interactive process π converges on D_m, each D_i of π is omitted for $i \ge m+1$.

Definition 5. Let $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be an RA. The *language accepted by* $\mathcal{A}$, denoted by $L(\mathcal{A})$, is defined as follows:

$$L(\mathcal{A}) = \{w \in \Sigma^* \mid \text{there exists } \pi \in IP(\mathcal{A}, w) \text{ that converges on } D_m \text{ at the } m\text{-th step for some } m \ge |w|, \text{and } f \subseteq D_m\}.$$

Example 2. Let us consider a reaction automaton $\mathcal{A} = (S, \Sigma, A, D_0, f)$ defined as follows:

$S = \{a, b, c, d, e, f\}$ with $\Sigma = \{a\}$,

$A = \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5, \mathbf{a}_6\}$, where

$\mathbf{a}_1 = (a^2, \emptyset, b)$, $\mathbf{a}_2 = (b^2, ac, c)$, $\mathbf{a}_3 = (c^2, b, b)$,

$\mathbf{a}_4 = (bd, ac, e)$, $\mathbf{a}_5 = (cd, b, e)$, $\mathbf{a}_6 = (e, abc, f)$,

$D_0 = d$.

Let $w = aaaaaaaa \in S^*$ be the input string and consider an interactive process π such that

$$\pi : d \to^a ad \to^a bd \to^a abd \to^a b^2d \to^a ab^2d \to^a b^3d \to^a ab^3d \to^a b^4d \to c^2d \to bd \to e \to f.$$

It can be easily seen that $\pi \in IP(\mathcal{A}, w)$ and $w \in L(\mathcal{A})$. Fig. 2 illustrates the whole view of possible interactive processes in $\mathcal{A}$ with inputs a^2, a^4 and a^8. For instance, since $\mathbf{a}_2^2 \in En^p_A(b^4d)$, it holds that $c^2d \in Res_A(b^4d)$. Hence, the step $b^4d \to c^2d$ is valid. We can also see that $L(\mathcal{A}) = \{a^{2^n} \mid n \ge 1\}$ which is context-sensitive.

4. Multistack machines

A multistack machine is a deterministic pushdown automaton with several stacks [13]. It is known that a two-stack machine is equivalent to a Turing machine as a language accepting device.

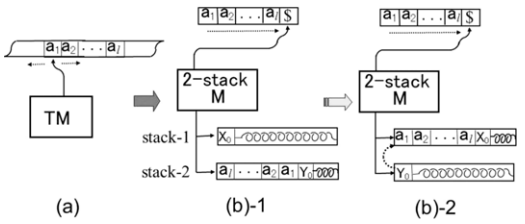

Fig. 3. (a) Turing machine (TM); (b)Two-stack machine M simulating TM, where \$ is the end marker for the input.

A k-stack machine $M = (Q, \Sigma, \Gamma, \delta, p_0, Z_0, F)$ is defined as follows: Q is a set of states, Σ is an input alphabet, Γ is a stack alphabet, $Z_0 = (Z_{01}, Z_{02}, \ldots, Z_{0k})$ is the k-tuple of the initial stack symbols, $p_0 \in Q$ is the initial state, F is a set of final states, δ is a transition function defined in the form: $\delta(p, a, X_1, X_2, \ldots, X_k) = (q, \gamma_1, \gamma_2, \ldots, \gamma_k)$, where $p, q \in Q$, $a \in \Sigma \cup \{\lambda\}$, $X_i \in \Gamma$, $\gamma_i \in \Gamma^*$ for each $1 \leq i \leq k$. This rule means that in state p, with X_i on the top of i-th stack, if the machine reads a from the input, then go to state q, and replace the the top of each i-th stack with γ_i for $1 \leq i \leq k$. We assume that each rule has a unique label and all labels of rules in δ is denoted by $Lab(\delta)$. Note that the k-stack machine can make a λ-move, but there cannot be a choice of a λ-move or a non-λ-move due to the deterministic property of the machine. The k-stack machine accepts a string by entering a final state.

In this paper, we consider a modification on a multistack (in fact, two-stack) machine. Recall that in the simulation of a given Turing machine TM with an input $w = a_1 a_2 \cdots a_\ell$ in terms of a multistack machine M, one can assume the following (see [13]):

(i) At first, two-stack machine M is devoted to making the copy of w on stack-2. This is illustrated in (a) and (b)-1 of Fig. 3, for the case of $k = 2$. *M requires only non-λ-moves.*
(ii) Once the whole input w is read-in by M, *no more access to the input tape of M is necessary*. After having w^R on stack-2, M moves over w^R (from stack-2) to produce w on stack-1, as shown in (b)-2. These moves only require λ-moves and after this, each computation step of M with respect to w is performed by a λ-move, without any access to w on the input tape.
(iii) Each stack has its own stack alphabet, each one being different from the others, and a set of final states is a singleton. Once M enters the final state, it immediately halts. Further, during a computation, each stack is not emptied.

Hence, without changing the computation power, we may restrict all computations of a multistack machine that satisfies the conditions (i), (ii), (iii). We call this modified multistack machine a *restricted multistack machine*.

In summary, a restricted k-stack machine M_r is composed by $2k + 5$ elements as follows:

$$M_r = (Q, \Sigma, \Gamma_1, \Gamma_2, \ldots, \Gamma_k, \delta, p_0, Z_{01}, Z_{02}, \ldots, Z_{0k}, f),$$

where for each $1 \leq i \leq k$, $Z_{0i} \in \Gamma_i$ is the initial symbol for the i-th stack used only for the bottom, $f \in Q$ is a final state, and its computation proceeds only in the above mentioned way (i), (ii), (iii). Especially, λ-moves are used after all non-λ-moves in a computation of M_r.

Proposition 1 (*Theorem 8.13 in [13]*)**.** *Every recursively enumerable language is accepted by a restricted two-stack machine.*

5. Main results

In this section we shall show the equivalence of the accepting powers between reaction machines and Turing machines. Taking Proposition 1 into consideration, it should be enough for the purpose of this paper to prove the following theorem.

Theorem 1. *If a language L is accepted by a restricted two-stack machine, then L is accepted by a reaction automaton.*

[Construction of an RA]
Let $M = (Q, \Sigma, \Gamma_1, \Gamma_2, \delta, p_0, X_0, Y_0, f)$ be a restricted two-stack machine with $\Gamma_1 = \{X_0, X_1, \ldots, X_n\}$, $\Gamma_2 = \{Y_0, Y_1, \ldots, Y_m\}$, $n, m \geq 1$, where $\Gamma = \Gamma_1 \cup \Gamma_2$, X_0 and Y_0 are the initial stack symbols for stack-1 and stack-2, respectively, and we may assume that $\Gamma_1 \cap \Gamma_2 = \emptyset$.

We construct an RA $\mathcal{A}_M = (S, \Sigma, A, D_0, f')$ as follows:

$$S = Q \cup \hat{Q} \cup \Sigma \cup \Gamma \cup \hat{\Gamma} \cup Lab(\delta) \cup \{f'\},$$
$$A = A_0 \cup A_a \cup \hat{A}_a \cup A_\lambda \cup \hat{A}_\lambda \cup A_X \cup \hat{A}_X \cup A_Y \cup \hat{A}_Y \cup A_f \cup \hat{A}_f,$$
$$D_0 = p_0 X_0 Y_0,$$

where the set of reactions A consists of the following 5 categories :

(1) $A_0 = \{(p_0 a X_0 Y_0, Lab(\delta), \hat{q} \cdot stm(\hat{x}) \cdot stm(\hat{y}) \cdot r') \mid r : \delta(p_0, a, X_0, Y_0) = (q, x, y),\ r' \in Lab(\delta)\}$,

(2) $A_a = \{(paX_iY_jr, \hat{\Gamma}, \hat{q} \cdot stm(\hat{x}) \cdot stm(\hat{y}) \cdot r') \mid a \in \Sigma,\ r : \delta(p, a, X_i, Y_j) = (q, x, y),\ r' \in Lab(\delta)\}$,

$\hat{A}_a = \{(\hat{p}a\hat{X}_i\hat{Y}_jr, \Gamma, q \cdot stm(x) \cdot stm(y) \cdot r') \mid a \in \Sigma,\ r : \delta(p, a, X_i, Y_j) = (q, x, y),\ r' \in Lab(\delta)\}$,

(3) $A_\lambda = \{(pX_iY_jr, \Sigma \cup \hat{\Gamma}, \hat{q} \cdot stm(\hat{x}) \cdot stm(\hat{y}) \cdot r') \mid r : \delta(p, \lambda, X_i, Y_j) = (q, x, y),\ r' \in Lab(\delta)\}$,

$\hat{A}_\lambda = \{(\hat{p}\hat{X}_i\hat{Y}_jr, \Sigma \cup \Gamma, q \cdot stm(x) \cdot stm(y) \cdot r') \mid r : \delta(p, \lambda, X_i, Y_j) = (q, x, y),\ r' \in Lab(\delta)\}$,

(4) $A_X = \{(X_k^2, \hat{Q} \cup \hat{\Gamma} \cup (Lab(\delta) - \{r\}) \cup \{f'\}, \hat{X}_k^{2^{|x|}}) \mid 0 \le k \le n,\ r : \delta(p, a, X_i, Y_j) = (q, x, y)\}$,

$\hat{A}_X = \{(\hat{X}_k^2, Q \cup \Gamma \cup (Lab(\delta) - \{r\}) \cup \{f'\}, X_k^{2^{|x|}}) \mid 0 \le k \le n,\ r : \delta(p, a, X_i, Y_j) = (q, x, y)\}$,

$A_Y = \{(Y_k^2, \hat{Q} \cup \hat{\Gamma} \cup (Lab(\delta) - \{r\}) \cup \{f'\}, \hat{Y}_k^{2^{|y|}}) \mid 0 \le k \le m,\ r : \delta(p, a, X_i, Y_j) = (q, x, y)\}$,

$\hat{A}_Y = \{(\hat{Y}_k^2, Q \cup \Gamma \cup (Lab(\delta) - \{r\}) \cup \{f'\}, Y_k^{2^{|y|}}) \mid 0 \le k \le m,\ r : \delta(p, a, X_i, Y_j) = (q, x, y)\}$,

(5) $A_f = \{(f, \hat{\Gamma}, f')\}$,

$\hat{A}_f = \{(\hat{f}, \Gamma, f')\}$.

Proof. We shall give an informal description on how to simulate M with an input $w = a_1 a_2 \cdots a_\ell$ in terms of $\mathcal{A}_M$ constructed above.

M starts its computation from the state p_0 with X_0 and Y_0 on the top of stack-1 and stack-2, respectively. This initial step is performed in $\mathcal{A}_M$ by applying a reaction in A_0 to $D_0 = p_0X_0Y_0$ together with a_1. In order to read the whole input w into $\mathcal{A}_M$, applying reactions in (2) and (4) leads to an interactive process in $\mathcal{A}_M : D_0 \to^{a_1} D_1 \to^{a_2} D_2 \to^{a_3} \cdots \to^{a_\ell} D_\ell$, where D_ℓ just corresponds to the configuration of M depicted in (b)-1 of Fig. 3. After this point, only reactions from (3), (4) and (5) are available in $\mathcal{A}_M$, because M makes only λ-moves.

Suppose that for $k \ge 1$, after making k-steps M is in the state p and has $\alpha_k \in \Gamma_1^*$ and $\beta_k \in \Gamma_2^*$ on the stack-1 and the stack-2, respectively. Then, from the manner of constructing A, it is seen that in the corresponding interactive process in $\mathcal{A}_M$, we have :

$$\begin{cases} D_k = p \cdot stm(\alpha_k) \cdot stm(\beta_k) \cdot r & (\text{if } k \text{ is even}) \\ D_k = \hat{p} \cdot stm(\hat{\alpha}_k) \cdot stm(\hat{\beta}_k) \cdot r & (\text{if } k \text{ is odd}) \end{cases}$$

for some $r \in Lab(\delta)$, where the rule labeled by r may be used at the $(k+1)$-th step. (Recall that $stm(x)$ is a multiset, in a special 2-power form, representing a string x.) Thus, the multisubset "$stm(\alpha_k)stm(\beta_k)$" in D_k is denoted by the strings in either Γ^* or $\hat{\Gamma}^*$ in an alternate fashion, depending upon the value k. Since there is no essential difference between strings denoted by Γ^* and its hat version, we only argue about the case when k is even.

Suppose that M is in the state p and has $\alpha = X_{i1} \cdots X_{it}X_0$ on the stack-1 and $\beta = Y_{j1} \cdots Y_{js}Y_0$ on the stack-2, where the leftmost element is the top symbol of the stack. Further, let r be the label of a transition $\delta(p, a_{k+1}, X_{i1}, Y_{j1}) = (q, x, y)$ (if $1 \le k \le l-1$) or $\delta(p, \lambda, X_{i1}, Y_{j1}) = (q, x, y)$ (if $l \le k$) in M to be applied. Then, the two stacks are updated as $\alpha' = xX_{i2} \cdots X_{it}X_0$ and $\beta' = yY_{j2} \cdots Y_{js}Y_0$. In order to simulate this move of M, we need to prove that it is possible in $\mathcal{A}_M$, $D_k \to^{a_{k+1}} D_{k+1}$ (if $1 \le k \le l-1$) or $D_k \to D_{k+1}$ (if $l \le k$), where

$$D_k = p \cdot stm(X_{i1}X_{i2} \cdots X_{it}X_0) \cdot stm(Y_{j1}Y_{j2} \cdots Y_{js}Y_0)r$$

$$D_{k+1} = \hat{q} \cdot stm(\hat{x}\hat{X}_{i2} \cdots \hat{X}_{it}\hat{X}_0) \cdot stm(\hat{y}\hat{Y}_{j2} \cdots \hat{Y}_{js}\hat{Y}_0)r'$$

for some $r' \in Lab(\delta)$. Taking a close look at D_k, we have that

$$D_k = pX_{i1}Y_{j1}r \cdot X_{i2}^2X_{i3}^{2^2} \cdots X_{it}^{2^{t-1}}X_0^{2^t} \cdot Y_{j2}^2Y_{j3}^{2^2} \cdots Y_{js}^{2^{s-1}}Y_0^{2^s},$$

from which it is easily seen that a multiset of reactions $\mathbf{z} = \mathbf{r}\mathbf{x_{i2}} \cdots \mathbf{x_{it}^{2^{t-2}}}\mathbf{x_0^{2^{t-1}}}\mathbf{y_{j2}} \cdots \mathbf{y_{js}^{2^{s-2}}}\mathbf{y_0^{2^{s-1}}}$ is in $En^p_{\mathcal{A}_M}(a_{k+1} + D_k)$ (if $1 \le k \le l-1$) or in $En^p_{\mathcal{A}_M}(D_k)$ (if $l \le k$), i.e., it is enabled by $a_{k+1} + D_k$ (if $1 \le k \le l-1$) or D_k (if $l \le k$) in a maximally parallel manner, where

$$\begin{cases} \mathbf{r} = (pa_{k+1}X_{i1}Y_{j1}r, \hat{\Gamma}, \hat{q} \cdot stm(\hat{x})stm(\hat{y})r') \in A_a & (\text{if } 1 \le k \le l-1) \\ \mathbf{r} = (pX_{i1}Y_{j1}r, \Sigma \cup \hat{\Gamma}, \hat{q} \cdot stm(\hat{x})stm(\hat{y})r') \in A_\lambda & (\text{if } l \le k), \end{cases}$$

for some $r' \in Lab(\delta)$,

$$\mathbf{x}_i = (X_i^2, \hat{Q} \cup \hat{\Gamma} \cup Lab(\delta) - \{r\} \cup \{f'\}, \hat{X}_i^{2^{|x|}}) \in A_X \quad (\text{for } i = 0, i2, \ldots, it),$$

$$\mathbf{y}_j = (Y_j^2, \hat{Q} \cup \hat{\Gamma} \cup Lab(\delta) - \{r\} \cup \{f'\}, \hat{Y}_j^{2^{|y|}}) \in A_Y \quad (\text{for } j = 0, j2, \ldots, js).$$

The result of the multiset of the reactions $\mathbf{z}$ is

$$\begin{aligned}&\hat{q}\cdot stm(\hat{x})stm(\hat{y})r'\cdot \hat{X}_{i2}^{2^{|x|}}\cdots\hat{X}_{it}^{2^{t-2+|x|}}\hat{X}_0^{2^{t-1+|x|}}\cdot\hat{Y}_{j2}^{2^{|x|}}\cdots\hat{Y}_{js}^{2^{s-2+|x|}}\hat{Y}_0^{2^{s-1+|x|}}\\&=\hat{q}\cdot stm(\hat{x}\hat{X}_{i2}\cdots\hat{X}_{it}\hat{X}_0)\cdot stm(\hat{y}\hat{Y}_{j2}\cdots\hat{Y}_{js}\hat{Y}_0)r'\\&=D_{k+1}.\end{aligned}$$

Thus, in fact it holds that $D_k\to^{a_{k+1}}D_{k+1}$ (if $1\le k\le l-1$) or $D_k\to D_{k+1}$ (if $l\le k$) in $\mathcal{A}_M$.

We note that there is a possibility that undesired reaction $\mathbf{r}'$ can be enabled at the $(k+1)$th step, where $\mathbf{r}'$ is of the form

$$\begin{cases}\mathbf{r}' = (pa_{k+1}X_{iu}Y_{jv}r,\ \hat{\Gamma},\ \hat{q'}\cdot stm(\hat{x'})stm(\hat{y'})r')\in A_a & (\text{if } 1\le k\le l-1)\\ \mathbf{r}' = (pX_{iu}Y_{jv}r,\ \Sigma\cup\hat{\Gamma},\ \hat{q'}\cdot stm(\hat{x'})stm(\hat{y'})r')\in A_\lambda & (\text{if } l\le k),\end{cases}$$

with $u\neq 1$ or $v\neq 1$, that is, the reactant of $\mathbf{r}'$ contains a stack symbol which is not the top of stack. If a multiset of reactions $\mathbf{z}'=\mathbf{r}'\mathbf{x}'_{\mathbf{1}}\cdots\mathbf{x}'_{\mathbf{t}'}\mathbf{y}'_{\mathbf{1}}\cdots\mathbf{y}'_{\mathbf{s}'}$ with $\mathbf{x}'_{\mathbf{1}},\ldots,\mathbf{x}'_{\mathbf{t}'}\in A_X,\mathbf{y}'_{\mathbf{1}},\ldots,\mathbf{y}'_{\mathbf{s}'}\in A_Y$ is used at the $(k+1)$th step, then D_{k+1} contains *both* the symbols without hat (in Γ) and the symbols with hat (in $\hat{Q}$ and $\hat{\Gamma}$). This is because in this case, X_{i1} or Y_{j1} in D_k which is not consumed at the $(k+1)$-th step remains in D_{k+1} (since the total numbers of X_{i1} and Y_{j1} are *odd*, these objects cannot be consumed out by the reactions from (4)). Hence, no reaction is enabled at the $(k+2)$-th step and f' is never derived after this wrong step.

From the arguments above, it holds that for an input $w\in\Sigma^*$, M enters the final state f (and halts) if and only if there exists $\pi: D_0,\ldots,D_i,\ldots\in IP(\mathcal{A}_M,w)$ such that D_{k-1} contains f or $\hat{f}$, D_k contains f', and π converges on D_k, for some $k\ge 1$. Therefore, we have that $L(M)=L(\mathcal{A}_M)$ holds. □

Corollary 1. *Every recursively enumerable language is accepted by a reaction automaton.*

Recall the way of constructing reactions A of $\mathcal{A}_M$ in the proof of Theorem 1. The reactions in categories (1), (2), (3) would not satisfy the condition of determinacy which is given immediately below. However, we can easily modify $\mathcal{A}_M$ to meet the condition.

Definition 6. Let $\mathcal{A}_M=(S,\Sigma,A,D_0,f')$ be an RA. Then, $\mathcal{A}_M$ is *deterministic* if for $a=(R,I,P)$, $a'=(R',I',P')\in A$, $(R=R')\wedge(I=I')$ implies that $a=a'$.

Theorem 2. *If a language L is accepted by a restricted two-stack machine, then L is accepted by a deterministic reaction automaton.*

Proof. Let $M=(Q,\Sigma,\Gamma_1,\Gamma_2,\delta,p_0,X_0,Y_0,f)$ be a restricted two-stack machine. For the RA $\mathcal{A}_M=(S,\Sigma,A,D_0,f')$ constructed for the proof of Theorem 1, we consider $\mathcal{A}'_M=(S\cup Lab\hat{(}\delta),\Sigma,A',D_0,f')$, where A' consists of the following 5 categories :

$$\begin{aligned}
&(1)\ A_0=\{(p_0aX_0Y_0,\ Lab(\delta)\cup\{\hat{r'}\},\ \hat{q}\cdot stm(\hat{x})\cdot stm(\hat{y})\cdot\hat{r'})\mid r:\delta(p_0,a,X_0,Y_0)=(q,x,y),\ r'\in Lab(\delta)\},\\
&(2)\ A_a=\{(paX_iY_jr,\ \hat{\Gamma}\cup\{\hat{r'}\},\ \hat{q}\cdot stm(\hat{x})\cdot stm(\hat{y})\cdot\hat{r'})\mid a\in\Sigma,\ r:\delta(p,a,X_i,Y_j)=(q,x,y),\ r'\in Lab(\delta)\},\\
&\quad\hat{A}_a=\{(\hat{p}a\hat{X}_i\hat{Y}_jr,\ \Gamma\cup\{r'\},\ q\cdot stm(x)\cdot stm(y)\cdot r')\mid a\in\Sigma,\ r:\delta(p,a,X_i,Y_j)=(q,x,y),\ r'\in Lab(\delta)\},\\
&(3)\ A_\lambda=\{(pX_iY_jr,\ \Sigma\cup\hat{\Gamma}\cup\{\hat{r'}\},\ \hat{q}\cdot stm(\hat{x})\cdot stm(\hat{y})\cdot\hat{r'})\mid r:\delta(p,\lambda,X_i,Y_j)=(q,x,y),\ r'\in Lab(\delta)\},\\
&\quad\hat{A}_\lambda=\{(\hat{p}\hat{X}_i\hat{Y}_jr,\ \Sigma\cup\Gamma\cup\{r'\},\ q\cdot stm(x)\cdot stm(y)\cdot r')\mid r:\delta(p,\lambda,X_i,Y_j)=(q,x,y),\ r'\in Lab(\delta)\},\\
&(4)\ A_X=\{(X_k^2,\ \hat{Q}\cup\hat{\Gamma}\cup(Lab\hat{(}\delta)-\{\hat{r}\})\cup\{f'\},\ \hat{X}_k^{2^{|x|}})\mid 0\le k\le n,\ r:\delta(p,a,X_i,Y_j)=(q,x,y)\},\\
&\quad\hat{A}_X=\{(\hat{X}_k^2,\ Q\cup\Gamma\cup(Lab(\delta)-\{r\})\cup\{f'\},\ X_k^{2^{|x|}})\mid 0\le k\le n,\ r:\delta(p,a,X_i,Y_j)=(q,x,y)\},\\
&\quad A_Y=\{(Y_k^2,\ \hat{Q}\cup\hat{\Gamma}\cup(Lab\hat{(}\delta)-\{\hat{r}\})\cup\{f'\},\ \hat{Y}_k^{2^{|y|}})\mid 0\le k\le m,\ r:\delta(p,a,X_i,Y_j)=(q,x,y)\},\\
&\quad\hat{A}_Y=\{(\hat{Y}_k^2,\ Q\cup\Gamma\cup(Lab(\delta)-\{r\})\cup\{f'\},\ Y_k^{2^{|y|}})\mid 0\le k\le m,\ r:\delta(p,a,X_i,Y_j)=(q,x,y)\},\\
&(5)\ A_f=\{(f,\hat{\Gamma},f')\},\\
&\quad\hat{A}_f=\{(\hat{f},\Gamma,f')\}.
\end{aligned}$$

The reactions in categories (1), (2), (3) in A' meet the condition where $\mathcal{A}'_M$ is deterministic, since the inhibitor of each reaction includes r' or $\hat{r'}$. We can easily observe that the equation $L(M)=L(\mathcal{A}'_M)$ is proved in a manner similar to the proof of Theorem 1. □

Corollary 2. *Every recursively enumerable language is accepted by a deterministic reaction automaton.*

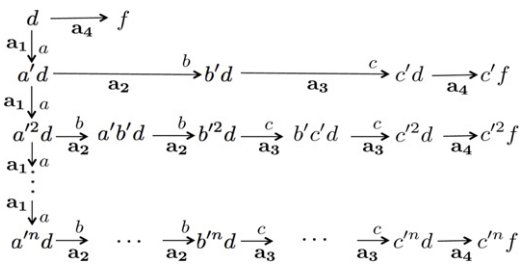

Fig. 4. Reaction diagram of $\mathcal{A}_1$ which accepts $L_1 = \{a^n b^n c^n \mid n \geq 0\}$.

6. Space complexity classes of RAs

We now consider space complexity issues of reaction automata. That is, we introduce some subclasses of reaction automata and investigate the relationships between classes of languages accepted by those subclasses of automata and language classes in the Chomsky hierarchy.

Let $\mathcal{A}$ be an RA and f be a function defined on **N**. Motivated by the notion of a workspace for a phrase-structure grammar [21], we define: for $w \in L(\mathcal{A})$ with $n = |w|$, and for π in $IP(\mathcal{A}, w)$,

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

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

$$WS(w, \mathcal{A}) = \min_\pi\{WS(w, \pi)\ |\pi \in IP(\mathcal{A}, w) \text{ that converges on } D_m \text{ for some } m \geq n \text{ and } D_m \text{ includes the final state}\}.$$

Definition 7. (i). An RA $\mathcal{A}$ is $f(n)$-*bounded* if for any $w \in L(\mathcal{A})$ with $n = |w|$, $WS(w, \mathcal{A})$ is bounded by $f(n)$.
(ii). If a function $f(n)$ is a constant k (resp. linear, polynomial, exponential), then $\mathcal{A}$ is termed k-bounded (resp. linearly-bounded, polynomially-bounded, exponentially-bounded), and denoted by k-RA (resp. *lin*-RA, *poly*-RA, *exp*-RA). Further, the class of languages accepted by k-RA (resp. *lin*-RA, *poly*-RA, *exp*-RA, arbitrary RA) is denoted by k-$\mathcal{RA}$ (resp. $\mathcal{LRA}$, $\mathcal{PRA}$, $\mathcal{ERA}$, $\mathcal{RA}$).

Let us denote by $\mathcal{REG}$ (resp. $\mathcal{LIN}$, $\mathcal{CF}$, $\mathcal{CS}$, $\mathcal{RE}$) the class of regular (resp. linear context-free, context-free, context-sensitive, recursively enumerable) languages.

Example 3. Let $L_1 = \{a^n b^n c^n \mid n \geq 0\}$ and consider an RA $\mathcal{A}_1 = (S, \Sigma, A, D_0, f)$ defined as follows:

$S = \{a, b, c, d, a', b', c', f\}$ with $\Sigma = \{a, b, c\}$,
$A = \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4\}$, where
$\mathbf{a}_1 = (a, bb', a')$, $\mathbf{a}_2 = (a'b, cc', b')$, $\mathbf{a}_3 = (b'c, \emptyset, c')$, $\mathbf{a}_4 = (d, abca'b', f)$,
$D_0 = d$.

Then, it holds that $L_1 = L(\mathcal{A}_1)$ (see Fig. 4).

Example 4. Let $L_2 = \{a^m b^m c^n d^n \mid m, n \geq 0\}$ and consider an RA $\mathcal{A}_2 = (S, \Sigma, A, D_0, f)$ defined as follows:

$S = \{a, b, c, d, a', c', p_0, p_1, p_2, p_3, f\}$ with $\Sigma = \{a, b, c, d\}$,
$A = \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5, \mathbf{a}_6, \mathbf{a}_7, \mathbf{a}_8, \mathbf{a}_9, \mathbf{a}_{10}, \mathbf{a}_{11}\}$, where
$\mathbf{a}_1 = (ap_0, bc, a'p_0)$, $\mathbf{a}_2 = (a'bp_0, c, p_1)$, $\mathbf{a}_3 = (a'bp_1, c, p_1)$, $\mathbf{a}_4 = (cp_0, d, c'p_2)$,
$\mathbf{a}_5 = (cp_1, d, c'p_2)$, $\mathbf{a}_6 = (cp_2, d, c'p_2)$, $\mathbf{a}_7 = (c'dp_2, \emptyset, p_3)$, $\mathbf{a}_8 = (c'dp_3, \emptyset, p_3)$,
$\mathbf{a}_9 = (p_0, abcd, f)$, $\mathbf{a}_{10} = (p_1, abcda', f)$, $\mathbf{a}_{11} = (p_3, abcda'c', f)$,
$D_0 = p_0$.

Then, it holds that $L_2 = L(\mathcal{A}_2)$ (see Fig. 5).

It should be noted that $\mathcal{A}_1$ and $\mathcal{A}_2$ are both *lin*-RAs, therefore, the class of languages $\mathcal{LRA}$ includes a context-sensitive language L_1 and a non-linear context-free language L_2.

Lemma 1. *For an alphabet* Σ *with* $|\Sigma| \geq 2$, *let* $h : \Sigma^* \to \Sigma^*$ *be an injection such that for any* $w \in \Sigma^*$, $|h(w)|$ *is bounded by a polynomial of* $|w|$. *Then, there is no polynomially-bounded reaction automaton which accepts the language* $L = \{wh(w) \mid w \in \Sigma^*\}$.

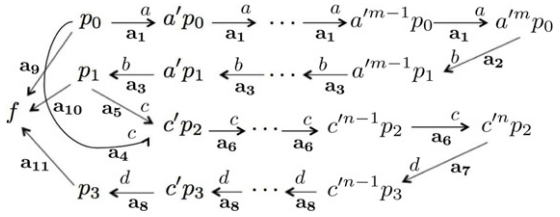

Fig. 5. Reaction diagram of $\mathcal{A}_2$ which accepts $L_2 = \{a^m b^m c^n d^n \mid m, n \geq 0\}$.

Proof. Assume that there is a *poly*-RA $\mathcal{A} = (S, \Sigma, A, D_0, f)$ such that $L(\mathcal{A}) = \{wh(w) \mid w \in \Sigma^*\}$. Let $|S| = m_1$, $|\Sigma| = m_2 \geq 2$ and the input string be $wh(w)$ with $|w| = n$.

Since $|h(w)|$ is bounded by a polynomial of $|w|$, $|wh(w)|$ is also bounded by a polynomial of n. Hence, for each D_i in an interactive process $\pi \in IP(\mathcal{A}, wh(w))$, it holds that $|D_i| \leq p(n)$ for some polynomial $p(n)$ from the definition of a *poly*-RA.

Let $\mathcal{D}_{p(n)} = \{D \in S^{\#} \mid |D| \leq p(n)\}$. Then, it holds that

$$|\mathcal{D}_{p(n)}| = \sum_{k=0}^{p(n)} {}_{m_1}\mathrm{H}_k = \sum_{k=0}^{p(n)} \frac{(k+m_1-1)!}{k! \cdot (m_1-1)!} = \frac{(p(n)+m_1)!}{p(n)! \cdot m_1!} = \frac{(p(n)+m_1)(p(n)+m_1-1)\cdots(p(n)+1)}{m_1!}.$$

($_{m_1}\mathrm{H}_k$ denotes the number of repeated combinations of m_1 things taken k at a time.)

Therefore, there is a polynomial $p'(n)$ such that $|\mathcal{D}_{p(n)}| = p'(n)$. Since it holds that $|\Sigma^n| = (m_2)^n$, if n is sufficiently large, we obtain the inequality $|\mathcal{D}_{p(n)}| < |\Sigma^n|$.

For $i \geq 0$ and $w \in \Sigma^*$, let $I_i(w) = \{D_i \in \mathcal{D}_{p(n)} \mid \pi = D_0, \ldots, D_i, \ldots \in IP(\mathcal{A}, w)\} \subseteq \mathcal{D}_{p(n)}$, i.e., $I_i(w)$ is the set of multisets in $\mathcal{D}_{p(n)}$ which appear as the i-th elements of interactive processes in $IP(\mathcal{A}, w)$. From the fact that $L(\mathcal{A}) = \{wh(w) \mid w \in \Sigma^*\}$ and h is an injection, we can show that for any two distinct strings $w_1, w_2 \in \Sigma^n$, $I_n(w_1)$ and $I_n(w_2)$ are incomparable. This is because if $I_n(w_1) \subseteq I_n(w_2)$, the string $w_2 h(w_1)$ is accepted by $\mathcal{A}$, which means that $h(w_1) = h(w_2)$ and contradicts that h is an injection.

Since for any two distinct strings $w_1, w_2 \in \Sigma^n$, $I_n(w_1)$ and $I_n(w_2)$ are incomparable and $I_n(w_1), I_n(w_2) \subseteq \mathcal{D}_{p(n)}$, it holds that

$$|\{I_n(w) \mid w \in \Sigma^n\}| \leq |\mathcal{D}_{p(n)}| < |\Sigma^n|.$$

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

Theorem 3. *The following inclusions hold:*

(1) $\mathcal{RE}\mathcal{G} = k\text{-}\mathcal{RA} \subset \mathcal{LRA} \subseteq \mathcal{PRA} \subset \mathcal{ERA} \subseteq \mathcal{RA} = \mathcal{RE}$ (for each $k \geq 1$).
(2) $\mathcal{LRA} \subset \mathcal{CS} \subseteq \mathcal{ERA}$.
(3) $\mathcal{LIN}$ ($\mathcal{CF}$) *and* $\mathcal{LRA}$ *are incomparable.*

Proof. (1) From the definitions, the inclusion $\mathcal{RE}\mathcal{G} \subseteq 1\text{-}\mathcal{RA}$ is straightforward. Conversely, for a given k-RA $\mathcal{A} = (S, \Sigma, A, D_0, f)$ and for $w \in L(\mathcal{A})$, there exists a π in $IP(\mathcal{A}, w)$ such that for each D_i appearing in π, we have $|D_i| \leq k$. Let $Q = \{D \in S^{\#} \mid |D| \leq k\}$ and $F = \{D \mid D \in Q, f \subseteq D, Res_A(D) = \{D\}\}$, and construct an NFA $M = (Q, \Sigma, \delta, D_0, F)$, where δ is defined by $\delta(D, a) \ni D'$ if $D \to^a D'$ for $a \in \Sigma \cup \{\lambda\}$. Then, it is seen that $L(\mathcal{A}) = L(M)$, and $k\text{-}\mathcal{RA} \subseteq \mathcal{RE}\mathcal{G}$, thus we obtain that $\mathcal{RE}\mathcal{G} = k\text{-}\mathcal{RA}$. The other inclusions are all obvious from the definitions. The language $L = \{a^n b^n \mid n \geq 0\}$ proves the proper inclusion : $\mathcal{RE}\mathcal{G} \subset \mathcal{LRA}$. A proper inclusion $\mathcal{PRA} \subset \mathcal{ERA}$ is due to that $L_3 = \{ww^R \mid w \in \{a, b\}^*\} \in \mathcal{ERA} - \mathcal{PRA}$, which follows from Lemma 1.

(2) Given an *lin*-RA $\mathcal{A}$, one can consider a linearly bounded automaton (LBA) M that simulates an interactive process π in $IP(\mathcal{A}, w)$ for each w, because of the linear boundedness of $\mathcal{A}$. This implies that $\mathcal{LRA} \subseteq \mathcal{CS}$. A proper inclusion is due to that $L_3 = \{ww^R \mid w \in \{a, b\}^*\} \in \mathcal{LIN} - \mathcal{LRA}$, which follows from Lemma 1.

Further, for a given LBA M, one can find an equivalent two-stack machine M_s whose stack lengths are linearly bounded by the input length. This implies, from the proof of Theorem 1, that M_s is simulated by an RA $\mathcal{A}$ that is exponentially bounded. Thus, it holds that $\mathcal{CS} \subseteq \mathcal{ERA}$.

(3) The language $L_1 = \{a^n b^n c^n \mid n \geq 0\}$ (resp. $L_2 = \{a^m b^m c^n d^n \mid m, n \geq 0\}$) is in $\mathcal{LRA} - \mathcal{CF}$ (resp. $\mathcal{LRA} - \mathcal{LIN}$), while, again from Lemma 1, the language L_3 is in $\mathcal{LIN} - \mathcal{LRA}$ (see Fig. 6). This completes the proof. □

7. Concluding remarks

Based on the formal framework presented in a series of papers [7–11], we have introduced the notion of reaction automata and investigated the language accepting powers of the automata. Roughly, a reaction automaton may be characterized in terms of three key words as follows : a *language accepting device* based on the *multiset rewriting* in the *maximally parallel manner*. Specifically, we have shown that in a computing schema with one-pot solution and a finite

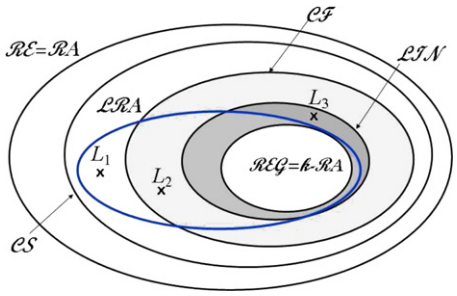

Fig. 6. Language class relations in the Chomsky hierarchy: $L_1 = \{a^n b^n c^n \mid n \geq 0\}$; $L_2 = \{a^m b^m c^n d^n \mid m, n \geq 0\}$; $L_3 = \{w w^R \mid w \in \{a, b\}^*\}$.

number of molecular species, reaction automata can perform the Turing universal computation. The idea behind their computing principle is to simulate the behavior of two pushdown stacks in terms of multiset rewriting with the help of an encoding technique, where both the manner of maximally parallel rewriting and the role of the inhibitors in each reaction are effectively utilized.

There already exist quite a few works investigating the notion of a multiset and its related topics [2] in which multiset automata and grammars are formulated and explored largely from the formal language theoretic point of view. Rather recent papers [16,17] focus on the accepting power of multiset pushdown automata to characterize the classes of multiset languages through investigating their closure properties.

To the authors' knowledge, however, relatively few works have been devoted to computing languages with multiset rewriting/communicating mechanism. Among them, one can find some papers published in the area of membrane computing (or spiking neural P-systems) where a string is encoded in some manner as a natural number and a language is specified as a set of natural numbers (e.g., [3]). Further, recent developments concerning P-automata and its variant called dP-automata are noteworthy in the sense that they may give rise to a new type of computing devices that could be a bridge between P-system theory and the theory of reaction systems and automata [5,14,20].

In fact, a certain number of computing devices similar to reaction automata have already been investigated in the literature. Among others, parallel labeled rewrite transition systems are proposed and investigated [12] in which multiset automata may be regarded as special type of reaction automata, whereas neither regulation by inhibitors nor maximally parallel manner of applying rules is employed in their rewriting process. A quite recent article [1] investigates the power of maximally parallel multiset rewriting systems (MPMRSs) and proves the existence of a universal MPMRS having smaller number of rules, which directly implies the existence of a universal antiport P-systems, with one membrane, having a smaller number of rules. In contrast to reaction automata, a universal MPMRS computes any partially recursive function provided that the input is the encoding of a register machine computing a target function.

Turning to the formal grammars, one can find random context grammars [6] and their variants (such as semi-conditional grammars in [18]) that employ regulated rewriting mechanisms called permitting symbols and forbidding symbols. The roles of these two are corresponding to reactants and inhibitors in reactions, whereas they deal with sets of strings (i.e., languages in the usual sense) rather than multisets. We finally refer to an article on stochastic computing models based on chemical kinetics, which proves that well-mixed finite stochastic chemical reaction networks with a fixed number of species can achieve Turing universal computability with an arbitrarily low error probability [22]. In this paper, we have shown that non-stochastic chemical reaction systems with a finite number of molecular species can also achieve Turing universality with the help of an inhibition mechanism.

Many subjects remain to be investigated along the research direction suggested by reaction automata in this paper. First, it is of importance to completely characterize the computing powers and the closure properties of complexity subclasses of reaction automata introduced in this paper. Secondly, from the viewpoint of designing chemical reactions, it is useful to explore a methodology for "chemical reaction programming" in terms of reaction automata. It is also interesting to simulate a variety of chemical reactions in the real world by the use of the framework of reaction automata.

Acknowledgements

The authors gratefully acknowledge useful remarks and comments by anonymous referees which improved an earlier version of this paper. The work of F. Okubo was possible due to Waseda University Grant for Special Research Projects: 2011A-842. The work of S. Kobayashi was in part supported by Grants-in-Aid for Scientific Research (C) No.22500010, Japanese Society for the Promotion of Science. The work of T. Yokomori was in part supported by Waseda University Grant for Special Research Projects: 2011B-056.

References

[1] A. Alhazov, S. Verlan, Minimization strategies for maximally parallel multiset rewriting systems, Theoretical Computer Science 412 (2011) 1587–1591.
[2] C. Calude, Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Multiset Processing, in: LNCS, vol. 2235, Springer, 2001.

[3] H. Chen, R. Freund, M. Ionescu, Gh. Păun, M.J. Pérez-Jiménez, On string languages generated by spiking neural P systems, in: Proceedings of the 4th Brainstorming Week on Membrane Computing, Seville, Spain, 2006, pp. 169–194.
[4] E. Csuhaj-Varju, C. Martin-Vide, V. Mitrana, Multiset automata, in: C. Calude, Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Multiset Processing, in: LNCS, vol. 2235, Springer, 2001, pp. 69–83.
[5] E. Csuhaj-Varju, M. Oswald, G. Vaszil, P automata, in: Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Handbook of Membrane Computing, Oxford University Press, 2010, pp. 144–167.
[6] J. Dassow, Gh. Păun, Regulated Rewriting in Formal Language Theory, in: EATCS Monographs on TCS, vol. 18, Springer-Verlag, 1989.
[7] A. Ehrenfeucht, G. Rozenberg, Reaction systems, Fundamenta Informaticae 75 (2007) 263–280.
[8] A. Ehrenfeucht, G. Rozenberg, Events and modules in reaction systems, Theoretical Computer Science 376 (2007) 3–16.
[9] A. Ehrenfeucht, G. Rozenberg, Introducing time in reaction systems, Theoretical Computer Science 410 (2009) 310–322.
[10] A. Ehrenfeucht, M. Main, G. Rozenberg, Combinatorics of life and death in reaction systems, Intern. J. Foundations of Computer Science 21 (2010) 345–356.
[11] A. Ehrenfeucht, M. Main, G. Rozenberg, Functions defined by reaction systems, Intern. J. Foundations of Computer Science 22 (2011) 167–178.
[12] Y. Hirshfeld, F. Moller, Pushdown automata, multiset automata, and petri nets, Theoretical Computer Science 256 (2001) 3–21.
[13] J.E. Hopcroft, T. Motwani, J.D. Ullman, Introduction to Automata Theory, Language and Computation, 2nd ed., Addison-Wesley, 2003.
[14] M. Ionescu, Gh. Păun, M.J. Pérez-Jiménez, T. Yokomori, Spiking neural dP systems, Fundamenta Informaticae 111 (2011) 423–436.
[15] M. Kudlek, C. Martin-Vide, Gh. Păun, Toward a formal macroset theory, in: C. Calude, Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Multiset Processing, in: LNCS, vol. 2235, Springer, 2001, pp. 123–134.
[16] M. Kudlek, P. Totzke, G. Zetzsche, Multiset pushdown automata, Fundamenta Informaticae 93 (2009) 221–233.
[17] M. Kudlek, P. Totzke, G. Zetzsche, Properties of multiset language classes defined by multiset pushdown automata, Fundamenta Informaticae 93 (2009) 235–244.
[18] Gh. Păun, A variant of random context grammars: semi-conditional grammars, Theoretical Computer Science 41 (1985) 1–17.
[19] Gh. Păun, Computing with membranes, Journal of Computer and System Sciences 61 (2000) 108–143.
[20] Gh. Păun, M.J. Pérez-Jiménez, P and dP automata: a survey, in: Lecture Notes in Computer Science, vol. 6570, Springer, 2011, pp. 102–115.
[21] A. Salomaa, Formal Languages, Academic Press, New York, 1973.
[22] D. Soloveichik, M. Cook, E. Winfree, J. Bruck, Computation with finite stochastic chemical reaction networks, Natural Computing 7 (2008) 615–633.
[23] Y. Suzuki, Y. Fujiwara, J. Takabayashi, H. Tanaka, Artificial life applications of a class of P systems, in: C. Calude, Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), Multiset Processing, in: LNCS, vol. 2235, Springer, 2001, pp. 299–346.