On language classes defined by reaction automata

Fumiya OKUBO

1 Introduction

Motivated by two notions of a reaction system ([3, 4, 5]) and a multiset ([1]), 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.

This paper is organized as follows. After preparing the basic notions and notations in Section 2, we introduce the main notion of reaction automata together with one language example in Section 3. Moreover 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 4, 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 discussed in Section 5.

2 Preliminaries

We assume that the reader is familiar with the basic notions of formal language theory. For unexplained details, refer to [9].

We use the basic notations regarding multisets that follow [2, 10]. 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 often identify a multiset μ with its string representation $w_\mu = a_1^{\mu(a_1)} \cdots a_n^{\mu(a_n)}$ or any permutation of w_μ. 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:

$$\begin{array}{ll} \textit{Inclusion}: & \mu_1 \subseteq \mu_2 \ \text{ iff } \ \mu_1(a) \leq \mu_2(a), \\ \textit{Sum}: & (\mu_1 + \mu_2)(a) = \mu_1(a) + \mu_2(a), \\ \textit{Intersection}: & (\mu_1 \cap \mu_2)(a) = \min\{\mu_1(a), \mu_2(a)\}, \\ \textit{Difference}: & (\mu_1 - \mu_2)(a) = \mu_1(a) - \mu_2(a), \\ & \text{(for the case } \mu_2 \subseteq \mu_1 \text{ only)}, \end{array}$$

for each $a \in V$. The sum for a family of multisets $\mathcal{M} = \{\mu_i\}_{i \in I}$ is 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 $|\mu| = \sum_{a \in V} \mu(a)$.

3 Reaction Automata

Inspired by the works of reaction systems, we have introduced the notion of reaction automata in [7] by extending sets in each reaction to multisets. Here, we start by recalling basic notions concerning reaction automata.

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,

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

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. 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.
(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:

$$Res_A^X(T) = \{T - R_\alpha + P_\alpha \,|\, \alpha \in En_A^X(T)\},$$

We note that $Res_A^X(T) = \{T\}$ 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 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 4. Let $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be an RA, $w = a_1 \cdots a_n \in \Sigma^*$ and $X \in \{sq, mp\}$. An *interactive process in* $\mathcal{A}$ *with input* w in X manner is an infinite sequence $\pi = D_0, \ldots, D_i, \ldots$, where

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

In order to represent an interactive process π, we also use the "arrow notation" for $\pi : D_0 \to^{a_1} D_1 \to^{a_2} D_2 \to^{a_3} \cdots \to^{a_{n-1}} D_{n-1} \to^{a_n} D_n \to D_{n+1} \to \cdots$. 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 [8], if it is allowed that $a_i = \lambda$ for some several $1 \le i \le n$, for an input string $w = a_1 \cdots a_n$, 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 $m \geq |w|$, 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). 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 $i \geq m+1$.

Definition 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:

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

Example 1. Let us consider a reaction automaton $\mathcal{A} = (S, \Sigma, A, D_0, f)$ defined as follows:

$S = \{p_0, p_1, a, b, a', f\}$ with $\Sigma = \{a, b\}$,
$A = \{\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4\}$, where
$\mathbf{a}_0 = (p_0, aba', f)$, $\mathbf{a}_1 = (p_0a, b, p_0a')$, $\mathbf{a}_2 = (p_0a'b, \emptyset, p_1)$,
$\mathbf{a}_3 = (p_1a'b, a, p_1)$, $\mathbf{a}_4 = (p_1, aba', f)$,
$D_0 = p_0$.

Figure 1 illustrates the whole view of possible interactive processes in $\mathcal{A}$ with inputs a^nb^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

$$\pi : p_0 \to^a p_0a' \to^a p_0a'^2 \to^a p_0a'^3 \to^b p_1a'^2 \to^b p_1a' \to^b p_1 \to f.$$

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

We note the following remark: 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 $L_{mp}(\mathcal{A}) = \{a^nb^n \mid n \geq 0\}$.

We shall show the equivalence of the accepting powers between reaction machines and Turing machines. For the details of proof, we refer [6], [7].

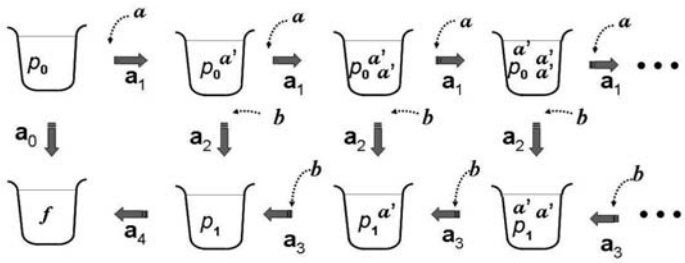

Figure 1: A graphic illustration of interactive processes for accepting strings in the language $L = \{a^n b^n \mid n \geq 0\}$ in terms of a reaction automaton $\mathcal{A}$.

Theorem 1. ([6], [7]) *The following relations hold* :
(1) *Every recursively enumerable language is accepted by a reaction automaton in maximally parallel manner.*
(2) *Every recursively enumerable language is accepted by a reaction automaton in sequential manner with λ-input mode.*

On the other hand, the equivalence may not hold for reaction automata in sequential manner with ordinary input mode.

Theorem 2. ([6]) *There exists a recursively enumerable language which cannot be accepted by any reaction automaton in sequential manner.*

4 Space Complexity Classes

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 $\mathbf{N}$. otivated by the notion of a workspace for a phrase-structure grammar ([9]), we define: for $w \in L(\mathcal{A})$ 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 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 $w \in L(\mathcal{A})$ with $n = |w|$, $WS(w, \mathcal{A})$

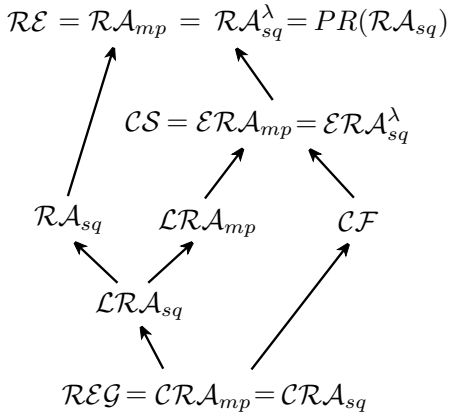

Figure 2: The diagram of the relation between the language classes regarding RA. A proper inclusion relation is denoted by a solid line.

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 $\mathcal{CRA}^{\lambda}_X$ (resp. $\mathcal{LRA}^{\lambda}_X$, $\mathcal{ERA}^{\lambda}_X$, $\mathcal{RA}^{\lambda}_X$).

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

For reaction automata and their space-bounded subclasses, the following results have been shown in [6], [7], [8].

Proposition 1. ([6], [7], [8]) *The following relations hold* :
(1) $\mathcal{REG} = \mathcal{CRA}_{mp} = \mathcal{CRA}_{sq} \subset \mathcal{LRA}_{sq} \subset \mathcal{LRA}_{mp} \subset \mathcal{ERA}_{mp} \subset \mathcal{RA}_{mp}$.
(2) $\mathcal{CF}$, $\mathcal{LRA}_{mp}$ *and* $\mathcal{RA}_{sq}$ *are incomparable one another.*
(3) $\mathcal{RE} = \mathcal{RA}_{mp} = PR(\mathcal{LRA}_{mp}) = PR(\mathcal{RA}_{sq})$.
(4) $\mathcal{CS} = \mathcal{ERA}_{mp} = \mathcal{ERA}^{\lambda}_{sq}$.

5 Concluding Remarks

Based on the formal framework presented in a series of papers [3, 4, 5], 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 two ways of rule applications, *maximally parallel manner* and *sequential manner*. Specifically, we have shown that reaction automata can perform the Turing universal computation in both ways of rule applications.

Moreover, we investigate reaction automata with a focus on the formal language theoretic properties of subclasses of reaction automata. Figure 2 summarizes the relationship among the classes of languages accepted by various types of RAs, TMs and the Chomsky hierarchy. Specifically, we have shown the followings:

- a language L is accepted by an exponential-bounded RA in maximally parallel manner if and only if L is a context-sensitive language,
- a language L is accepted by an exponential-bounded RA with λ-input mode in sequential manner if and only if L is a context-sensitive language,
- any recursively enumerable language can be expressed as a homomorphic image of a language in $\mathcal{RA}_{sq}$,
- the three classes of languages $\mathcal{CF}$, $\mathcal{LRA}_{mp}$ and $\mathcal{RA}_{sq}$ are incomparable one another.

Many subjects remain to be investigated along the research direction suggested by reaction automata in this paper. Most of all, it is of importance to explore the relationship between RAs and other computing devices that are based on the multiset rewriting, such as a variety of P-systems and their variants ([2]). It would be also 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 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.

[References]

[1] C. Calude, Gh. Paun, G. Rozenberg and A. Salomaa (Eds.), *Multiset Processing*, LNCS 2235, Springer, 2001.

[2] Gh. Paun, G. Rozenberg, A. Salomaa (Eds.), *Handbook of Membrane Computing*, Oxford University Press, 2010.

[3] A. Ehrenfeucht, G. Rozenberg, Reaction systems, *Fundamenta Infor-maticae* vol.75, pp.263-280, 2007.

[4] A. Ehrenfeucht, G. Rozenberg, Events and modules in reaction systems, *Theoretical Computer Science* vol.376, pp.3-16, 2007.

[5] A. Ehrenfeucht, G. Rozenberg, Introducing time in reaction systems, *Theoretical Computer Science* vol.410, pp.310-322, 2009.

[6] F. Okubo, On the Computational Power of Reaction Automata Work-ing in Sequential Manner, *4th Workshop on Non-Classical Models for Automata and Applications*, book@ocg.at series 290, pp.149-164, sterre-ichische Computer Gesellschaft, 2012.

[7] F. Okubo, S. Kobayashi, T. Yokomori, Reaction Automata, *Theoretical Computer Science*, 429, pp.247-257, 2012.

[8] F. Okubo, S. Kobayashi, T. Yokomori, On the Properties of Language Classes Dened by Bounded Reaction Automata, *Theoretical Computer Science*, 454, pp.206-221, 2012.

[9] A. Salomaa, *Formal Languages*, Academic Press, New York, 1973.

[10] Y. Suzuki, Y. Fujiwara, J. Takabayashi, H. Tanaka, Articial Life Ap-plications of a Class of P Systems, in: *Multiset Processing*, C. Calude, Gh. Păun, G. Rozenberg, A. Salomaa (Eds.), LNCS 2235, Springer, pp.299-346, 2001.