Computing with Multisets: A Survey on Reaction Automata Theory

Takashi Yokomori[1](✉) and Fumiya Okubo[2]

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

[2] Faculty of Business Administration, Takachiho University, 2-19-1 Ohmiya, Suginami-ku, Tokyo 168-8508, Japan
fokubo@takachiho.ac.jp

1 Introduction

In Natural Computing research [18], as mathematical modeling tools of biochemical reactions, Ehrenfeucht and Rozenberg introduced a formal model called reaction systems [6] for investigating the functioning of the living cell, where two basic components (reactants and inhibitors) play a key role as a regulation mechanism in controlling interactions.

Inspired by the notion of reaction systems, *reaction automata* (RAs) have been first introduced in [10] as computing devices for accepting string languages. The notion of RAs is an extension of reaction systems in that RAs employ reactions which are defined by triples (consisting of reactants, inhibitors and products), however they entail dealing with multisets for reactants and products (rather than usual sets as reaction systems do). Thus, RAs are computing models based on multiset rewriting that accept string languages. Another feature that distinguishes RAs from reaction systems is that a reaction automaton receives its input by feeding one symbol of an input string at each step of the computation. In this respect, RAs have the taste similar to P automata, P systems with accepting configurations, in which the idea of taking input sequences of multisets into the systems from the environment was introduced (e.g., [5]).

This survey paper is primarily based on the research works on RAs achieved in [9–12,15] in which various classes of RAs are considered in four types of computation process: with/without λ-input modes in rule application of the maximally parallel manner and the sequential manner.

In what follows, we make a quick survey of the results presented in this paper. The first result in the series of papers was that RAs are computationally Turing universal [10]. In the paper, space-bounded complexity classes of RAs was also introduced, and in the follow-up paper [11], it eventually turned out that

F. Okubo—The work of T.Yokomori was in part supported by JSPS KAKENHI, Grant-in-Aid for Scientific Research (C) JP17K00021, and by Waseda University grant for Special Research Project: 2017K-121. The work of F. Okubo was in part supported by JSPS KAKENHI Grant Number JP16K16008.

F. Manea et al. (Eds.): CiE 2018, LNCS 10936, pp. 421–431, 2018.
https://doi.org/10.1007/978-3-319-94418-0_42

exponential-bounded RAs can exactly characterize the class of context-sensitive languages. Further characterization result of the class of recursively enumerable languages was developed by using the class of linear-bounded RAs together with homomorphisms. The paper [11] also considered some of the closure properties of language classes defined by linear-bounded RAs, showing that the class forms an AFL, while the issue of the computing powers of RAs with λ-input mode in sequential manner was taken up in [9] and investigated to prove that they have again the Turing universal computability. Further investigations was made to a simpler variant of RAs called Chemical reaction automata (CRAs), and it was proved that CRAs with λ-input mode in parallel manner have the Turing universal computability, while their computing powers coincide with the class of Petri net languages when working in sequential manner. In [15], several subclasses of CRAs was introduced and their computing powers was studied. We leave more details of the results to later sections.

The rest of this section is devoted to a brief preliminary. We assume that the reader is familiar with the basic notions of reaction systems as well as of formal language theory. For unexplained details in the theory of reaction systems and in formal language theory, refer to, e.g., [6,7], respectively.

We use the basic notations and definitions concerning multisets that follow [4]. A *multiset* over an alphabet V is a mapping $\mu : V \rightarrow \mathbf{N}$, 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 over V can be represented by any permutation of the string $x = a_1^{\mu(a_1)} \cdots a_n^{\mu(a_n)}$, where $V = \{a_1, a_2, \cdots, a_n\}$. (Note that for each $a \in V$, a^0 is an empty string λ.) In this sense, a multiset μ is often identified with its string representation x_μ or any permutation of x_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.

2 Reaction Automata

Inspired by the works of reaction systems (initiated by [6]), the notion of reaction automata has been introduced in [10] by extending sets in each reaction (of a reaction system) 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 (i.e., $\alpha \in A^{\#}$),

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

Notes. (1). A is often identified with its label set and is used as an alphabet. (2). The symbol $\sum_{\mathbf{a} \in A}$ denotes the sum of multisets.

In the existing works on reaction automata, two ways of applying reactions have been considered: the sequential manner and the maximally parallel manner. The former manner was adopted in [10,11], while the literatures [9,12,15] studied the latter as well.

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:

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

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.

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.

Unlike the reaction system, a reaction automaton takes its input symbol from the environment into the current multiset (or state) representing current configuration, from time to time during the computing process. This idea was already considered and realized in the P automata theory (e.g., [5]).

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

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

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

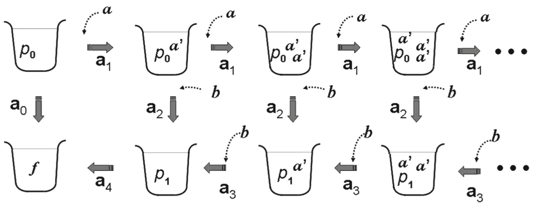

Fig. 1. An illustration of interactive processes for accepting the language $L = \{a^nb^n \mid n \geq 0\}$ in terms of the reaction automaton $\mathcal{A}$.

By $IP_X(\mathcal{A}, w)$ we denote the set of all interactive processes in $\mathcal{A}$ with input w in X manner. If for an input string $w = a_1 \cdots a_n$, it is allowed that $a_i = \lambda$ for some $1 \leq i \leq n$, then an interactive process is said to be *with λ-input mode*. By $IP^{\lambda}_X(\mathcal{A}, w)$ we denote the set of all interactive processes in $\mathcal{A}$ with λ-input mode in X manner for the input w ([11]).

For an interactive process π in $\mathcal{A}$ with input w, if $En^X_A(D_m) = \emptyset$ for some $m \geq |w|$, then we have that $Res^X_A(D_m) = \{D_m\}$ 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 the successive multisets $D_{m+1}, D_{m+2}, \ldots$ are omitted.

Definition 5. Let $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be an RA and $X \in \{sq, mp\}$. The *languages accepted by* $\mathcal{A}$ are defined as follows:

$$\begin{aligned} L_X(\mathcal{A}) = \{w \in \Sigma^* \mid\ & \text{there exists } \pi \in IP_X(\mathcal{A}, w) \text{ such that } \pi \text{ converges on } D_m \\ & \text{for some } m \geq |w| \text{ and } f \text{ is included in } D_m\} \\ L^{\lambda}_X(\mathcal{A}) = \{w \in \Sigma^* \mid\ & \text{there exists } \pi \in IP^{\lambda}_X(\mathcal{A}, w) \text{ satisfying the same as } L_X(\mathcal{A})\}. \end{aligned}$$

Example 1. Let us consider a reaction automaton $\mathcal{A} = (\{p_0, p_1, a, b, a', f\}, \{a, b\}, \{\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4\}, p_0, f)$, where $\mathbf{a}_0 = (p_0, aba', f)$, $\mathbf{a}_1 = (p_0a, b, p_0a')$, $\mathbf{a}_2 = (p_0a'b, \emptyset, p_1)$, $\mathbf{a}_3 = (p_1a'b, a, p_1)$, $\mathbf{a}_4 = (p_1, aba', f)$. 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 is easily seen that $\pi \in IP_{sq}(\mathcal{A}, w)$ and $w \in L_{sq}(\mathcal{A})$. Further, we see that $L_{sq}(\mathcal{A}) = \{a^nb^n \mid n \geq 0\}$ which is a context-free language.

We remark that this interactive process can be also performed by $\mathcal{A}$ in maximally parallel manner, i.e. $\pi \in IP_{mp}(\mathcal{A}, w)$. In fact, it holds that $L_{mp}(\mathcal{A}) = \{a^nb^n \mid n \geq 0\}$.

3 Main Results on RAs

This section presents some results on reaction automata (RAs) that have been established in an earlier stage. In what follows, a language accepted by an RA is often referred to as an *RA language*.

3.1 Computing Powers of RAs and Their Subclasses

It has been proven that the accepting power of reaction automata with both manners of applying reactions coincides with that of Turing machines.

Theorem 1 ([9,10])**.** *The followings 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.
(3) There exists a recursively enumerable language which cannot be accepted by any reaction automaton in sequential manner.

The proofs are based on two facts: (i) every recursively enumerable language is accepted by a *restricted* two-stack machine (Theorem 8.13 in [7]) and (ii) one can devise an injective function to encode strings into multisets. The result (2) is in marked contrast to (3), which clarifies the computing power of λ-input mode in sequential manner.

The space complexity issues of reaction automata (RAs) have been considered. By restricting the volume of each multiset that is produced in an interactive process by RA, some subclasses of RAs were introduced and investigated on relations between classes of languages accepted by those subclasses of RAs and language classes in the Chomsky hierarchy.

Let $\mathcal{A}$ be an RA and $X \in \{sq, mp\}$. Motivated by the notion of a workspace for a phrase-structure grammar, we define the counterpart of RA as follows: For $w \in L_X(\mathcal{A})$ with $n = |w|$, the *workspace of $\mathcal{A}$ for w* is defined as:

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

Definition 6. Let f be a function defined on $\mathbf{N}$ and $X \in \{sq, mp\}$.

(1) An RA $\mathcal{A}$ is *f(n)-bounded* if for any $w \in L_X(\mathcal{A})$ with $n = |w|$, $WS(w, \mathcal{A}) \leq f(n)$.
(2) If a function $f(n)$ 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 $\mathcal{C}o\mathcal{RA}_X$ (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 $\mathcal{C}o\mathcal{RA}_X^\lambda$ (resp. $\mathcal{LRA}_X^\lambda$, $\mathcal{ERA}_X^\lambda$, $\mathcal{RA}_X^\lambda$).

In order to explore and make clearer inclusion relations among language classes (considered so far), it is necessary to find a family of languages (or at least a particular language) with certain properties which plays a role of witness to distinguish one class from the other. The following lemma is useful for the purpose.

Lemma 1 ([9]). *Let Σ be an alphabet with $|\Sigma| \geq 2$ and $h : \Sigma^* \rightarrow \Sigma^*$ be an injection, and consider $L_h = \{wh(w) \mid w \in \Sigma^*\}$. Then, L_h is not in $\mathcal{RA}_{sq}$.*

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

Theorem 2 ([9,10]). *The following inclusions hold :*

(1). $\mathcal{REG} = \mathcal{CoRA}_{mp} \subset \mathcal{LRA}_{mp} \subset \mathcal{ERA}_{mp} \subseteq \mathcal{RA}_{mp} = \mathcal{RE}$
(2). $\mathcal{REG} = \mathcal{CoRA}_{sq} \subset \mathcal{LRA}_{sq} \subset \mathcal{RA}_{sq} \subset \mathcal{RA}^{\lambda}_{sq} = \mathcal{RE}$.
(3). $\mathcal{ERA}^{\lambda}_{sq} = \mathcal{ERA}_{mp} = \mathcal{CS}$.
(4). *$\mathcal{CF}$, $\mathcal{LRA}_{mp}$ and $\mathcal{RA}_{sq}$ are incomparable.*

Thus, new characterizations of the classes $\mathcal{REG}, \mathcal{CS}$ and $\mathcal{RE}$ have been established in terms of the subclasses of RA languages $\mathcal{CoRA}_{mp}, \mathcal{ERA}_{mp}, \mathcal{RA}_{mp}$, respectively. As seen later, however, the class $\mathcal{CF}$ has been proved incomparable to *any* known class of languages defined by RAs so far, exhibiting a unique position within the RA language hierarchy.

3.2 Some Other Characterizations of RA Language Classes

One of the primary issues in the formal language theory is to investigate the closure properties of a language class under various language operations. When featuring the classes $\mathcal{LRA}_{mp}$ and $\mathcal{LRA}^{\lambda}_{mp}$, the following has been proven.

Theorem 3 ([10,11])

(1). *$\mathcal{LRA}_{mp}$ is closed under union, intersection, concatenation, derivative, λ-free morphisms, λ-free gsm-mappings and shuffle, while* **not** *closed under complementation, quotient by regular languages, morphisms or gsm-mappings.*
(2). *$\mathcal{LRA}^{\lambda}_{mp}$ is closed under union, intersection, concatenation, Kleene $+$, Kleene $*$, derivative, λ-free morphisms, inverse morphisms, λ-free gsm-mappings and shuffle.*

We remark that in order to prove some of the negative closure properties of $\mathcal{LRA}_{mp}$, the following lemma is of crucial importance.

Lemma 2 ([10]). *For an alphabet Σ with $|\Sigma| \geq 2$, let $h : \Sigma^* \rightarrow \Sigma^*$ be an injection such that for any $w \in \Sigma^*$, $|h(w)|$ is bounded by a polynomial of $|w|$. Then, the language $L_h = \{wh(w) \mid w \in \Sigma^*\}$ is not in $\mathcal{LRA}_{mp}$.*

Further characterization results of $\mathcal{RE}$ have been developed by using $\mathcal{LRA}_{mp}$ and $\mathcal{RA}_{sq}$together with homomorphisms and regular languages.

Theorem 4 ([9,11])

(1). *For any recursively enumerable language L, there exists an LRA $\mathcal{A}$ such that $L = h(L_{mp}(\mathcal{A}))$ for some projection h.*
(2). *For any recursively enumerable language L, there exists an LRA $\mathcal{A}$ such that $L = R\backslash L_{mp}(\mathcal{A})$ (or $L_{mp}(\mathcal{A})/R$) for some regular language R.*
(3). *For any recursively enumerable language L, there exists an RA $\mathcal{A}$ such that $L = h(L_{sq}(\mathcal{A}))$ for some projection h.*

4 Chemical Reaction Automata

As a simple and modified version of a reaction automaton, a *chemical reaction automaton* (CRA) has been introduced and investigated [12]. Rather lately, this computing model CRA turned out to be important, because it can provide an online computational model for a molecular programming language called Chemical reaction networks (CRNs [19]). It is known that CRNs involve wet implementations by a molecular reaction primitive called DNA strand displacement (DSD) systems.

Specifically, a CRA is a 5-tuple (S, Σ, A, D_0, F), where each reaction in A is of the form $(R, \emptyset, P)$ (each reaction in CRA has *no inhibitor*), and F is a *finite set* of final multisets. For convenience, each reaction in A is denoted by $R \to P$. In an interactive process of CRA, if $En^X_A(D) = \emptyset$, $Res^X_A(D)$ is undefined. A language accepted by a CRA $\mathcal{A} = (S, \Sigma, A, D_0, F)$ is defined by

$$L^\lambda_X(\mathcal{A}) = \{w \in \Sigma^*_\lambda \mid \pi : D_0 \to^{a_1} D_1 \to^{a_2} \cdots \to^{a_n} D \in IP^\lambda_X(A, w),\, D \in F\}.$$

Remarks: The acceptance condition of CRA computation is slightly different from that of RA computation. A CRA accepts an input string if the final multiset *coincides* with an element of F, while an RA accepts an input string if the final multiset *includes* a particular symbol f. This difference is significant to obtain the results in our paper.

4.1 The Computation Power of CRAs

It has been shown that CRAs working in maximally parallel manner are computationally Turing universal. Our proof requires the fact that (i) a two-counter machine is equivalent to a Turing machine as a language accepting device [7] as well as the result that (ii) for any k-counter machine, there effectively exists an equivalent CRA. From (i) and (ii), the following is derived:

Theorem 5 ([12]). *The computational power of CRAs with λ-input mode in maximally parallel manner is equivalent to that of Turing machines.*

A naive question now arises: Concerning the computing power of CRAs whether or not there exists a real gap between working in maximally parallel manner and in sequential manner. This question is solved as a corollary of the next theorem.

Theorem 6 ([12]). *A language L is generated by a Petri net system if and only if L is accepted by a CRA with λ-input mode in sequential manner.*

Note that among several types of Petri net languages, here we mean a language of L^{λ}-type in [17]. Since the class of Petri net languages is strictly included in the class of context-sensitive languages and is incomparable to the class of context-free languages [17], it turns out that the computational power of CRAs with λ-input mode in sequential manner is *less* powerful than that of CRAs with λ-input mode in maximally parallel manner.

4.2 Determinism and Reversibility in CRAs

In this section we introduce the notions of determinism and reversibility into CRAs, and investigate the computational powers of those classes of CRAs in comparison with the language classes of Chomsky hierarchy.

The computing power of reversible CRAs involves the physical realization of molecular programming of chemical reaction networks with DNA strand displacement system implementation [19], and therefore, it is of great significance to elucidate the computing capabilities of both deterministic and reversible CRAs from the theoretical viewpoint of molecular computing.

Unlike the determinism of conventional computation models such as push-down automata, since a reactant is not divided into "input" part and "memory (multiset)" part, the determinism of CRAs cannot be decided only by a form of transition rules. This comes from the property of multiset memory, that is, from the current configuration alone, a CRA cannot identify a reactant of the next reaction to be applied. Therefore, the determinism of CRA has to be defined so as to exclude any branching computation, regardless of a non-empty input or empty input.

A CRA is said to be *deterministic* if for every input symbol $a \in \Sigma$ and every reachable configuration, the resultant multiset after a reaction is unique. Similar to the definition of deterministic pushdown automata, this condition is extended to the case of λ-input mode.

It is not trivial, only from their definitions, to recognize the difference of computing powers of the determinism between realtime CRA and CRA with λ-input mode, where "realtime" means "no λ input is allowed". The following result might be rather unexpected in some sense.

Lemma 3 ([15]). *If a language L is accepted by a DCRA with λ-input mode, then L is also accepted by a realtime DCRA.*

Note that by our definition of DCRA, without receiving an input symbol, no configuration of a DCRA with λ-input mode can have an enabled reaction, while this is not the case for a realtime DCRA. In order to make a reaction, a realtime DCRA always requires an input symbol, even if its configuration has an enabled reaction with no input symbol.

Information preserving computations (forward and backward deterministic computations) are very important and are considered as "reversibility" in

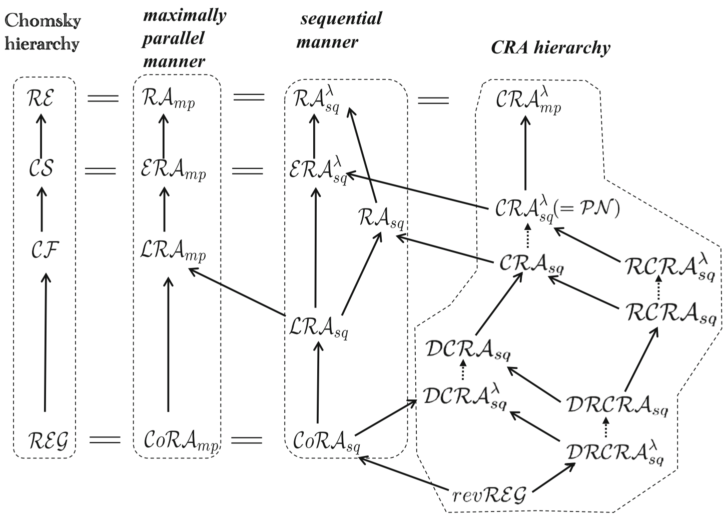
Chomsky hierarchy
maximally parallel manner
sequential manner
CRA hierarchy
$\mathcal{RE}$ = $\mathcal{RA}_{mp}$ = $\mathcal{RA}_{sq}^{\lambda}$ = $\mathcal{CRA}_{mp}^{\lambda}$
$\mathcal{CS}$ = $\mathcal{ERA}_{mp}$ = $\mathcal{ERA}_{sq}^{\lambda}$
$\mathcal{RA}_{sq}$
$\mathcal{CRA}_{sq}^{\lambda}(=\mathcal{PN})$
$\mathcal{CF}$
$\mathcal{LRA}_{mp}$
$\mathcal{CRA}_{sq}$
$\mathcal{RCRA}_{sq}^{\lambda}$
$\mathcal{RCRA}_{sq}$
$\mathcal{LRA}_{sq}$
$\mathcal{DCRA}_{sq}$
$\mathcal{DCRA}_{sq}^{\lambda}$
$\mathcal{DRCRA}_{sq}$
$\mathcal{REG}$ = $\mathcal{C}o\mathcal{RA}_{mp}$ = $\mathcal{C}o\mathcal{RA}_{sq}$
$\mathcal{DRCRA}_{sq}^{\lambda}$
$rev\mathcal{REG}$

Fig. 2. Inclusion relations among a variety of RA language classes at large.

many existing research papers (e.g., [2,3]). However, in order to understand their properties in more details, we want to take a position to think of them apart(i.e., separate into forward and backward determinisms). Thus, in our view the "reversibility" simply means that the previous configuration of computation can be uniquely determined (backward determinism).

A CRA is said to be *reversible* if for every input symbol $a \in \Sigma$ and every reachable configuration D, the set of configurations which directly reaches D with a is a singleton. For the case of λ-input mode, this condition is extended to the case of λ-input mode in a natural manner.

The following lemma holds true for a deterministic and reversible CRA (abbrev. DRCRA).

Lemma 4 ([15]). *If a language L is accepted by a DRCRA with λ-input mode, then L is also accepted by a realtime DRCRA.*

By $\mathcal{CRA}_{sq}$, $\mathcal{CRA}_{sq}^{\lambda}$, $\mathcal{DCRA}_{sq}$, $\mathcal{DCRA}_{sq}^{\lambda}$, $\mathcal{RCRA}_{sq}$, $\mathcal{RCRA}_{sq}^{\lambda}$, $\mathcal{DRCRA}_{sq}$, and $\mathcal{DRCRA}_{sq}^{\lambda}$, we denote the classes of languages accepted by realtime CRAs, CRAs with λ-input mode, realtime DCRAs, DCRAs with λ-input mode, realtime RCRAs, RCRAs with λ-input mode, realtime deterministic and reversible CRAs (DRCRAs), DRCRAs with λ-input mode, respectively.

Remarks: Due to the space limitation, most of the details on the results of DCRAs and RCRAs are omitted and the reader is advised to refer to [15]. Instead, Fig. 2 summarizes the inclusion relations among various classes of CRA languages discussed in this paper, where $\mathcal{PN}$ is the class of Petri net languages of L^{λ}-type [17] and $rev\mathcal{REG}$ is the class of zero-reversible regular languages [1].

Last but not least, it should be noted the following:

(1). Another type of "finite automata with multiset memory (FAMMs)" was proposed and investigated in [14] which employs a rule application mode similar to (but different from) that of RAs, and with FAMM framework a new characterization of Chomsky hierarchy was established.
(2). The reader is kindly advised to refer to another survey paper on reaction automata theory [13] for more details on the results and discussion left out in this paper because of the space limit.

5 Future Research Topics

Many subjects remain to be investigated along the research direction suggested by reaction automata.

- **Inclusion relations and Computing powers:**
  - Further refinements of the hierarchy of RA language classes and CRA language classes are strongly encouraged to clarify the inclusion relations in Fig. 2.
  - It is of great importance to explore the relationships between subclasses of $\mathcal{RA}$ and others defined by computing devices based on the multiset rewriting, such as a variety of P-systems and their variants (e.g., P automata and dP automata [5,16]).
  - It is also intriguing to clarify the relationships between subclasses of CRAs studied here and others defined by reversible computing models such as reversible pushdown automata [8].
  - It remains open whether or not deterministic CRAs (with/without λ-input mode) in maximally parallel manner are Turing universal.
- **Complexity issues:** There remain open time complexity issues to be studied in the hierarchies of $\mathcal{RA}$ and $\mathcal{CRA}$ classes. For example, no efforts have been made yet for investigating the *time* complexity of any class from the hierarchies.
- **Decision problems:** One may be encouraged to study a variety of the decision problems on subclasses within $\mathcal{RA}$ hierarchy. For example, it is an interesting question to explore the equivalence problem for the class $\mathcal{LRA}$ or the classes of deterministic/reversible CRAs.
- **Other issues:** It would also be useful to develop methods for simulating a variety of chemical reactions in the real world application, by the use of the framework based on reaction automata. For that purpose, investigating stochastic models based on RAs has to be conducted, and such stochastic versions of RAs may provide useful simulation tools for analyzing any natural phenomena modeled by RAs.

Finally, considering the natural correspondence to (or analogy of) classic theory of automata, we conclude this survey with our firm belief that Reaction Automata are computational devices which deserve much further research efforts.

References

1. Angluin, D.: Inference of reversible languages. J. ACM **29**(3), 741–765 (1982)
2. Alhazov, A., Freund, R., Morita, K.: Sequential and maximally parallel multiset rewriting: reversibility and determinism. Nat. Comput. **11**, 95–106 (2012)
3. Bennett, C.H.: Logical reversibility of computation. IBM J. Res. Dev. **17**(6), 525–532 (1973)
4. Calude, C.S., PĂun, G., Rozenberg, G., Salomaa, A. (eds.): WMC 2000. LNCS, vol. 2235. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45523-X
5. Csuhaj-Varju, E., Vaszil, G.: P automata. In: The Oxford Handbook of Membrane Computing, pp. 145–167 (2010)
6. Ehrenfeucht, A., Rozenberg, G.: Reaction systems. Fundam. Inform. **75**, 263–280 (2007)
7. Hopcroft, J.E., Motwani, T., Ullman, J.D.: Introduction to Automata Theory, Language and Computation, 2nd edn. Addison-Wesley, Boston (2003)
8. Kutrib, M., Malcher, A.: Reversible pushdown automata. J. Comput. Syst. Sci. **78**, 1814–1827 (2012)
9. Okubo, F.: Reaction automata working in sequential manner. RAIRO Theor. Inform. Appl. **48**, 23–38 (2014)
10. Okubo, F., Kobayashi, S., Yokomori, T.: Reaction automata. Theor. Comput. Sci. **429**, 247–257 (2012)
11. Okubo, F., Kobayashi, S., Yokomori, T.: On the properties of language classes defined by bounded reaction automata. Theor. Comput. Sci. **454**, 206–221 (2012)
12. Okubo, F., Yokomori, T.: The computational capability of chemical reaction automata. In: Murata, S., Kobayashi, S. (eds.) DNA 2014. LNCS, vol. 8727, pp. 53–66. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-11295-4_4. Also, in Natural Computing, vol. 15, pp. 215–224 (2016)
13. Okubo, F., Yokomori, T.: Recent developments on reaction automata theory: a survey. In: Suzuki, Y., Hagiya, M. (eds.) Recent Advances in Natural Computing. MI, vol. 9, pp. 1–22. Springer, Tokyo (2015). https://doi.org/10.1007/978-4-431-55105-8_1
14. Okubo, F., Yokomori, T.: Finite automata with multiset memory: a new characterization of chomsky hierarchy. Fundam. Inform. **138**, 31–44 (2015)
15. Okubo, F., Yokomori, T.: The computing power of determinism and reversibility in chemical reaction automata. In: Adamatzky, A. (ed.) Reversibility and Universality. ECC, vol. 30, pp. 279–298. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-73216-9_13
16. Paun, G., Rozenberg, G., Salomaa, A.: The Oxford Handbook of Membrane Computing. Oxford University Press, Inc., New York (2010)
17. Peterson, J.L.: Petri Net Theory and the Modelling of Systems. Prentice-Hall, Englewood Cliffs (1981)
18. Rozenberg, G., Back, T., Kok, J.N. (eds.): Handbook of Natural Computing. Section IV: Molecular Computation, vol. 3, pp. 1071–1355. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-540-92910-9
19. Thachuk, C., Condon, A.: Space and energy efficient computation with DNA strand displacement systems. In: Stefanovic, D., Turberfield, A. (eds.) DNA 2012. LNCS, vol. 7433, pp. 135–149. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-32208-2_11