A Study on Computation Capability of Biochemical Reactions

生化学反応による計算能力の研究

Fumiya Okubo

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy at Waseda University

November 2013

A Study on Computation Capability of Biochemical Reactions

Fumiya Okubo

A dissertation submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy at Waseda University

November 2013

Abstract

The purpose of this dissertation is to explore the computation capability of biochemical reactions. For this purpose, we investigate the computation model by multiset rewriting and the one based on a structure of DNA molecules in terms of formal language theory and computation theory. This thesis is organized by the following three contents.

(i) We propose new computing models called reaction automata that feature language acceptors with multiset rewriting as a computing mechanism. The notion of reaction automata is based on the formal framework of reaction systems which have been introduced by Ehrenfeucht and Rozenberg to investigate the interactive behaviors of biochemical reactions. We show that reaction automata are computationally Turing universal. Further, some subclasses of reaction automata with space complexity are investigated. Their language classes are compared to the ones in the Chomsky hierarchy and the ones accepted by the variants of Turing machines.

(ii) We introduce a new operation in formal language theory, called hairpin incompletion, inspired by DNA hairpin structures which have numerous applications to develop novel computing mechanisms in molecular computing. The hairpin in-

completion operation provides a formal language theoretic framework that models a bio-molecular technique nowadays known as Whiplash PCR. We show that a family of languages closed under intersection with regular sets, concatenation with regular sets, and finite union is closed under one-sided iterated hairpin incompletion, and that a family of languages containing all linear languages and closed under circular permutation, left derivative and substitution is also closed under iterated hairpin incompletion.

(iii) Insertion and deletion operations have a rather old history in both formal language theory. Recently, they have been drawing renewed attention in relation to the theory of molecular computing. We shall provide the characterization of context-free languages based on only insertion operations which are applied in a context-free manner and have the small length of the inserted string involved. Specifically, we show that each context-free language L can be represented in the form $L = h(L(\gamma) \cap F^+)$, where γ is an insertion system of weight $(3, 0)$ (at most three symbols are inserted in a context-free manner), h is a projection, and F^+ is a 2-star language. A similar characterization can be obtained for recursively enumerable languages, where insertion systems of weight $(3, 3)$ and 2-star languages are involved. All of these refine and improve the results by Păun et al.

Acknowledgments

Foremost, I wish to express my heartfelt gratitude to my supervisor Professor Takashi Yokomori. He gave me a valuable guidance and continued encouragements through my scientific life in graduate course. All of the works in this thesis as well as the others have been accomplished by his helpful advices, various suggestions and discussions.

I also wish to gratefully thank to Professor Satoshi Kobayashi of University of Electro-Communications, Professor Etsuro Moriya of Waseda University and Professor Yasuhiro Suzuki of Nagoya University, who participated in the dissertation committee. I have been blessed with the opportunity to collaborate with Professor Kobayashi. The studies regarding "Reaction Automata" could not be achieved without the fruitful discussions with him and his valuable comments. Professor Moriya took an interest in my thesis and spared a great deal of his precious time. He gave me valuable comments and encouragements. Professor Suzuki gave me lots of interesting topics in the research field of natural computing. His wide knowledge and novel ideas motivates me to research on this field.

I am also grateful to Dr. Kaoru Fujioka of Fukuoka Women's University, who gave me advices and encouragements about going to graduate school.

Finally, my special thanks go to my wife Yuri and my family Osamu, Yoshiko, Miki, Miho for their physical and mental support and care, which kept my enthusiasm to the research.

Contents

Chapter 1

Introduction

1.1 Computational models of biochemical reactions

In recent years, the study on computational models of biochemical reactions has attracted much attention in the field of theoretical computer science. There are two goals in exploring computational models of biochemical reactions. One is to understand the way how biochemical reactions realize information processing. "Chemical reactions" may be considered as information processing or computation which is performed by a transition of states in a solution. Regarding biochemical reactions as the base of computing mechanism, it is possible to investigate the properties of biochemical reactions from the point of view of computational theory. The other is to develop a methodology for artificially synthesizing biochemical reaction systems. In order to achieve a desired biochemical functioning, a suitable model for such a system of biochemical reactions must be pursued. Further, computer simulation may facilitate verification of the validity of the used model.

1.2 Classification of models of biochemical reactions

In general, choosing an appropriate model is a critical factor to understand a natural phenomenon through investigation based on the model. Also, it is of great importance to find the simplest among many models preserving the properties to be examined. The methodologies for investigating models of biochemical reactions are classified into three approaches as follows.

(1) *Differential equation*: When analyzing the process of chemical reactions, we measure the concentration of each molecule in the solution. For the case in which the number of molecules is enormous, it is approximated as a continuous quantity. Hence, differential equations are readily applicable to describe the variation of the concentration. The differential equation model for biochemical reactions has an advantage in that the analytic problem of examining the behaviors of reactions can be reduced to that of solving the corresponding differential equations.

(2) *Multiset rewriting system*: In the case where relatively small numbers of molecules are involved in the reaction, it is appropriate to deal with the concentration of each molecule as a discrete quantity. In such a case, therefore, the notion of a multiset is a suitable choice to represent the concentrations of molecules. A multiset rewriting system is a discrete state transition system that models chemical reactions in a natural way. For instance, vector addition systems and Petri nets, well-studied in computer science, are categorized as multiset rewriting systems. A model based on multiset rewriting system enables one to analyze the property of a chemical reaction system in a constructive manner.

(3) *Molecular computation*: In contrast to the preceding two approaches where each molecule is abstracted as a mere symbol, one can consider a formal model

which deals with structured molecules and utilizes the biochemical properties of those structured molecules. A research area called “molecular computation” aims at achieving the computation by making use of the biochemical properties of well-designed structural molecules such as DNA polymers. The first achievement was brought by the Adleman’s groundbreaking experiment in 1994. By encoding a small instance of the Hamiltonian path problem (HPP), one of the NP-complete problems, into the DNA strands, Adleman showed a novel method for solving an HPP by biochemical techniques. In the experiment, in particular, the property called “Watson-Crick complementarity” of DNA double strands plays a critical role.

The present thesis is concerned with several research topics and will develop formal models within these three approaches mentioned above.

1.3 Organization of the dissertation

The purpose of this dissertation is to explore the computation capability of biochemical reactions. For this purpose, we consider the computation model by multiset rewriting in Chapter 3 and the one based on a structure of DNA in Chapter 4 and Chapter 5. We analyze them using formal language theory and computation theory. This thesis is organized by the following chapters.

In Chapter 2, we prepare the basic notions and notations from formal language theory and multiset theory.

In Chapter 3, we propose a novel computation model by multiset rewriting; reaction automata. To construct a model for biochemical reactions, We refer reaction systems which is a formal model introduced in [6] by Ehrenfeucht and

Rozenberg. Reaction systems, for investigating the functioning of the living cell, are based on the idea that the functioning is decided by interactions between biochemical reactions, where two basic components (reactants and inhibitors) play a key role as a regulation mechanism in controlling interactions. Ehrenfeucht and Rozenberg show, in [6], that reaction systems provide a formal framework suited for investigating in an abstract level the way of emergence and evolution of biochemical events and modules.

Inspired by the notion of reaction systems, reaction automata are introduced as computing devices for accepting string languages. The notion of reaction automata is an extension of reaction systems in that reactions defined by triples consisting of reactants, inhibitors, and products are employed in reaction automata, however they deal with multisets for reactants and products (rather than usual sets as reaction systems do). Another feature that distiguishes from reaction systems is that a reaction automaton receives its input by feeding one symbol of an input string at each step of computation. Thus, reaction automata are computing models based on multiset rewriting that accept string languages.

The first result on reaction automata is that reaction automata are computationally Turing universal, that is, a recursively enumerable language is accepted by a reaction automaton. Space-bounded complexity classes of reaction automata have been also introduced, and it is explored that a class of Turing machines having an equivalent power of a class of reaction automata.

In Chapter 4, we introduce a new operation in formal language theory, called hairpin incompletion, inspired by intra molecular phenomena in molecular biology. DNA hairpin structures have numerous applications to develop novel computing mechanisms in molecular computing.

A hairpin structure is well-known as one of the most popular secondary structures for a single stranded DNA (or RNA) molecule to form, with the help of Watson-Crick complementarity and annealing, under a certain biochemical condition in a solution. The hairpin incompletion is related to the known investigations on computation by a hairpin structure in the following points:

- the hairpin incompletion is a natural extension of the notion of bounded hairpin completion introduced and studied in [15] which is a restricted variant of the hairpin completion with the property that the length of the prefix (suffix) prolongation is constantly bounded.

- the hairpin incompletion is also regarded as a restricted variant of the notion of hairpin lengthening recently introduced in [23] in which the prolongation of a strand that allows to stop itself at any position in the process of completing a hairpin structure.

- the hairpin incompletion can provide a purely formal framework that models a bio-molecular technique called Whiplash PCR that has nowadays been recognized as a promising experimental technique and has been proposed in [13] by Hagiya et al.

We study the closure properties of language families under both the operation and its iterated version. It is shown that any family of languages with certain closure properties is closed under the hairpin incompletion. We then consider the case of applying the iterated hairpin incompletion operations, and show that every abstract family of languages (AFL) is closed under the iterated one-sided hairpin incompletion. This result is further extended to the general case of the iterated

hairpin incompletion, and it is shown that any family of languages including all linear languages and with certain closure properties is also closed under the iterated hairpin incompletion, and as a corollary that the family of context-free languages is closed under the iterated hairpin incompletion.

In Chapter 5, we consider insertion and deletion operations which have a rather old history in both formal language theory, and computing models based on insertion-deletion have been recently drawing renewed attention in relation to the theory of molecular computing. From the viewpoint of biochemically implementing those computing models, it is of crucial importance to investigate the computing power of context-free operations of insertion-deletion, because of their simplicity in comparison to the context-dependent counterparts.

In this chapter, we shall provide the following characterization of context-free languages that are based on only insertion operations applied in a context-free manner and as small as possible in the length of the inserted string involved. Specifically, it is proved that for each λ-free context-free language L there exist a projection h, a context-free insertion system γ, and a star language F^+ such that $L = h(L(\gamma) \cap F^+)$, where γ only allows inserting at most three symbols in a context-free manner, and the length of each string in F is no more than two. Further, we shall show that a manner of construction used in the proof can be applied to characterize recursively enumerable languages in a similar form of $h(L(\gamma) \cap F^+)$, for some insertion system γ and the same type of F. All of these refine and improve the results for the language families in [33].

Chapter 2

Preliminary

2.1 Formal language theory

We assume that the reader is familiar with the basic notions of formal language theory, for unexplained details refer to [37]. In particular, for the notions of abstract family of languages, we refer to [40].

The set of natural numbers, $\{0, 1, 2, \dots\}$ is denoted by $\mathbf{N}$. For a set S, $|S|$ denotes the cardinality of S. The family of finite subsets of a set S is denoted by $\mathcal{P}(S)$. The empty set is denoted by $\emptyset$.

An *alphabet* is a finite nonempty set of abstract symbols. For an alphabet V, V^* is the set of all finite-length strings of symbols from V, where λ is the empty string and $|w|$ is the length of $w \in V^*$. For a symbol a in V we denote by $|w|_a$ the number of occurences of a in w. Moreover, V^+ is defined as $V^+ = V^* - \{\lambda\}$. For $k \geq 0$, we define $V^{\geq k} = \{w \in V^* \,|\, |w| \geq k\}$.

For a Chomsky grammar $G = (N, T, S, P)$, the set of the labels of P is denoted by $Lab(P) = \{r \,|\, r : A \to \alpha \in P\}$.

For an alphabet V, let $\bar{V} = \{\bar{a} \,|\, a \in V\}$ ($\bar{a}$ is barred copy of a.). V and $\bar{V}$ are considered to be disjoint. If V contains k symbols, then the *Dyck language* over

V and $\bar{V}$ is the language generated by the context-free grammar $G = (\{S\}, V \cup \bar{V}, S, P)$, where $P = \{S \to SS, S \to \lambda, S \to aS\bar{a} \mid a \in V\}$. Let *Dyck* be the class of Dyck languages.

We denote by $\mathcal{RE}$, $\mathcal{CS}$, $\mathcal{CF}$, $\mathcal{LIN}$ $\mathcal{REG}$ and $\mathcal{FIN}$ the families of recursively enumerable, context-sensitive, context-free, linear, regular and finite languages, respectively.

The boolean operations (with languages) are denoted as usual: $\cup$ – union, $\cap$ – intersection, $\bar{\cdot}$ – complementation.

For $k \geq 0$, let $pref_k(w)$ and $suf_k(w)$ be the prefix and the suffix of w of length k, respectively. For $k \geq 0$, we define $Pref_{\leq k}(w) = \{pref_i(w) \mid 0 \leq i \leq k\}$ and $Suf_{\leq k}(w) = \{suf_i(w) \mid 0 \leq i \leq k\}$. For $k \geq 1$, let $Inf_k(w)$ be the set of infixes of w of length k. If $|w| \leq k-1$, then $pref_k(w)$, $suf_k(w)$ and $Inf_k(w)$ are all undefined. (Note that for $w \in V^+$, $pref_k(w)$ and $suf_k(w)$ are elements in $Inf_k(w)$.) For $k \geq 0$, let $pInf_k(w)$ be the set of proper infixes of w of length k, while if $|w| = k$ or $k+1$, then $pInf_k(w) = \emptyset$.

The concatenation of L_1, L_2 is $L_1L_2 = \{xy \mid x \in L_1, y \in L_2\}$. By wL (Lw) we simply denote $\{w\}L$ ($L\{w\}$), i.e., the concatenation of w with a language L. The left (right) quotient of a language $L_1 \subseteq V^*$ with respect to $L_2 \subseteq V^*$ is

$$L_2 \backslash L_1 = \{w \in V^* \mid \text{there is } x \in L_2 \text{ such that } xw \in L_1\}$$

$$(L_1 / L_2 = \{w \in V^* \mid \text{there is } x \in L_2 \text{ such that } wx \in L_1\}).$$

The left (right) derivative of a language L with a word w is defined by $w\backslash L = \{x \in V^* \mid wx \in L\}$ ($L/w = \{x \in V^* \mid xw \in L\}$).

For a word $w = a_1 a_2 \cdots a_n \in V^*$, w^R is the reversal of w, that is, $(a_1 a_2 \cdots a_n)^R = a_n \cdots a_2 a_1$. For $x, y \in V^*$ we define their *shuffle* by

$$shuf(x, y) = \{x_1 y_1 \dots x_n y_n \mid x = x_1 \dots x_n, y = y_1 \dots y_n, x_i, y_i \in V^*, 1 \leq i \leq n, n \geq 1\}.$$

The notion of shuffle is extended to languages L_1, L_2 as

$$Shuf(L_1, L_2) = \{shuf(x, y) \mid x \in L_1, y \in L_2\}.$$

We define further:

$$\begin{aligned}
&L^0 = \{\lambda\},\\
&L^{i+1} = LL^i,\ i \geq 0,\\
&L^* = \bigcup_{i=0}^{\infty} L^i \text{ (the } * \text{-Kleene closure)},\\
&L^+ = \bigcup_{i=1}^{\infty} L^i \text{ (the } + \text{-Kleene closure)}.
\end{aligned}$$

A morphism $h : V^* \to U^*$ such that $h(a) \in U$ for all $a \in V$ is called a *coding*, and it is a *weak coding* if $h(a) \in U \cup \{\lambda\}$ for all $a \in V$. A weak coding is a *projection* if $h(a) \in \{a, \lambda\}$ for each $a \in V$. For a morphism h we define a mapping h^{-1} by $h^{-1}(w) = \{x \in V^* \mid h(x) = w\}$ and we call it an *inverse morphism*.

For families of languages $\mathcal{L}$, $\mathcal{L}_1$ and $\mathcal{L}_2$, we introduce the following families of languages:

$$WC(\mathcal{L}) = \{h(L) \mid h \text{ is a weak coding}, L \in \mathcal{L}\}$$

$$PR(\mathcal{L}) = \{h(L) \mid h \text{ is a projection}, L \in \mathcal{L}\}$$

$$H^{-1}(\mathcal{L}) = \{h^{-1}(L) \mid h \text{ is a morphism}, L \in \mathcal{L}\}$$

$$\mathcal{L}_1 \cap \mathcal{L}_2 = \{L_1 \cap L_2 \mid L_1 \in \mathcal{L}_1, L_2 \in \mathcal{L}_2\}$$

A generalized sequential machine (gsm) is a system $g = (K, V_1, V_2, s_0, F, \delta)$, where K is a set of states, $s_0 \in K$ is an initial state, $F \subseteq K$ is a set of final states, V_1, V_2 are alphabets (the input and the output alphabet, respectively), and $\delta : K \times V_1 \to \mathcal{P}(V_2^* \times K)$. If $\delta(s, a) \subseteq V_2^+ \times K$ for all $s \in K, a \in V_1$, then g is said to be *λ-free*. For $s, s' \in K, a \in V_1, y \in V_1^*, x, z \in V_2^*$, we write $(x, s, ay) \vdash (xz, s', y)$ if $(z, s') \in \delta(s, a)$. Then, for $w \in V_1^*$, we define

$$g(w) = \{z \in V_2^* \mid (\lambda, s_0, w) \vdash^* (z, s, \lambda), s \in F\}.$$

The mapping g is extended in the natural way to languages over V_1.

A family of languages is *nontrivial* if it contains at least one language different from $\emptyset$ and $\{\lambda\}$. A nontrivial family of languages is called a *trio* if it is closed under λ-free morphisms, inverse morphisms, and with regular languages. A trio closed under union is called a *semi-AFL* (AFL is an abbreviation of abstract family of languages). A semi-AFL closed under concatenation and Kleene + is called an *AFL*. A trio/semi-AFL/AFL is said to be *full* if it is closed under arbitrary morphisms (and Kleene $*$ in the case of AFL's).

2.2 Multiset theory

We use the basic notations regarding multisets that follow [35, 41]. 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:

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

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 $|\mu| = \sum_{a \in V} \mu(a)$.

Chapter 3

Reaction automata

3.1 Introduction

In recent years, a series of seminal papers [6, 7, 8] 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. Rather recent two papers [9, 10] 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., [1] 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., [32, 41]).

Motivated by these two notions of a reaction system and a multiset, in this chapter we will introduce computing devices called *reaction automata* and show their computational power. 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 this sense, reaction automata may also be regarded as a

simplified variants of P automata introduced by Csuhaj-Varju and Vaszil in [5] with no membrane structure.

This chapter is organized as follows. In Section 3.2, we formally describe the notion of reaction automata (RAs) and the classes of languages accepted by them together with four examples. In addition, Turing machines and their several variants, e.g., restricted multistack machines, $s(n)$-restricted Turing machines, are introduced. Then, in Section 3.3, we present the main results: reaction automata are computationally universal in two ways for applying reactions, i.e., maximally parallel manner and sequential manner with allowing λ-input. We also consider some subclasses of reaction automata from a viewpoint of the complexity theory in Section 3.4, and investigate those classes of languages in comparison to Chomsky hierarchy. Finally, concluding remarks as well as future research topics are briefly discussed in Section 3.5.

3.2 Preliminaries

3.2.1 Formal definition of 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 ([6, 7, 8]). 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 [6] 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 chapter.

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,

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

We consider two ways for applying reactions, i.e., sequential manner and maximally parallel manner.

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

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

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

If it is allowed that $a_i = \lambda$ for some several $1 \leq i \leq n$, 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 $Res_A^X(D_m) = \{D_m\}$ 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:

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

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

Definition 6. Let $X = \{sq, mp\}$. The class of languages accepted by RAs in X manner is denoted by $\mathcal{RA}_X$. The class of languages accepted by RAs with λ-input mode in X manner is denoted by $\mathcal{RA}_X^\lambda$.

3.2.2 Examples

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

$$S = \{p_0, p_1, a, b, a', f\} \text{ with } \Sigma = \{a, b\},$$

$$A = \{\mathbf{a}_0, \mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4\}, \text{ where}$$

$$\mathbf{a}_0 = (p_0, aba', f), \ \mathbf{a}_1 = (p_0a, b, p_0a'), \ \mathbf{a}_2 = (p_0a'b, \emptyset, p_1),$$

$$\mathbf{a}_3 = (p_1a'b, a, p_1), \ \mathbf{a}_4 = (p_1, aba', f),$$

$$D_0 = p_0.$$

Figure 3.1 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

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

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 $L_{mp}(\mathcal{A}) = \{a^nb^n \mid n \geq 0\}$.

Example 2. Let $L_1 = \{a^nb^nc^n \mid n \geq 0\}$ and consider an RA $\mathcal{A}_1 = (S, \Sigma, A, D_0, f)$

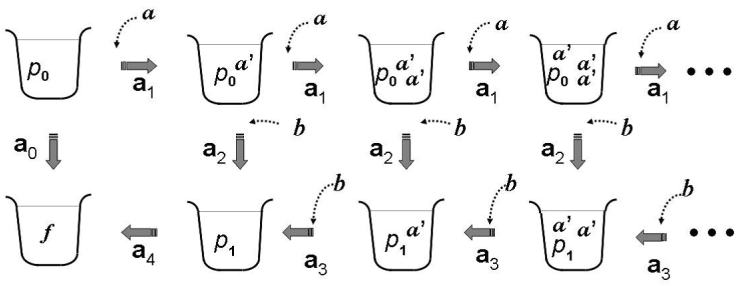

Figure 3.1: A graphic illustration of interactive processes for accepting strings in the language $L = \{a^n b^n \mid n \geq 0\}$ in terms of a reaction automaton $\mathcal{A}$.

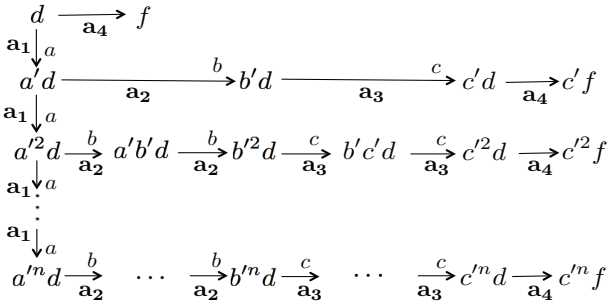

Figure 3.2: Reaction diagram of $\mathcal{A}_1$ which accepts $L_1 = \{a^n b^n c^n \mid n \geq 0\}$.

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 $L_1 = L_{mp}(\mathcal{A}_1) = L_{sq}(\mathcal{A}_1)$ (see Figure 3.2).

Example 3. Let $L_2 = \{a^m b^m c^n d^n \mid m, n \geq 0\}$ and consider an RA $\mathcal{A}_2 = (S, \Sigma, A, D_0, f)$

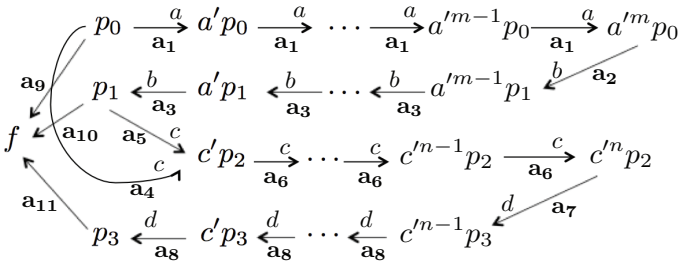

Figure 3.3: Reaction diagram of $\mathcal{A}_2$ which accepts $L_2 = \{a^m b^m c^n d^n \mid m, n \geq 0\}$.

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 $L_2 = L_{mp}(\mathcal{A}_2) = L_{sq}(\mathcal{A}_2)$ (see Figure 3.3).

Example 4. Let $\mathcal{A}_3 = (S, \Sigma, A, D_0, f)$ be an RA defined as follows:

$S = \{c, p_0, p_1, n_1, c_1, c_2, d, e, f\}$, with $\Sigma = \{c\}$,

$A = \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5, \mathbf{a}_6, \mathbf{a}_7, \mathbf{a}_8\}$, where

$\mathbf{a}_1 = (p_0, c, p_1)$, $\mathbf{a}_2 = (p_1, ef, p_1n_1)$,

$\mathbf{a}_3 = (c, p_1, c_1)$, $\mathbf{a}_4 = (c_1^2, p_0c_2e, c_2)$, $\mathbf{a}_5 = (c_2^2, p_0c_1e, c_1)$,

$\mathbf{a}_6 = (c_1d, p_0c_2, e)$, $\mathbf{a}_7 = (c_2d, p_0c_1, e)$, $\mathbf{a}_8 = (e, p_0cc_1c_2, f)$,

$D_0 = dp_0$.

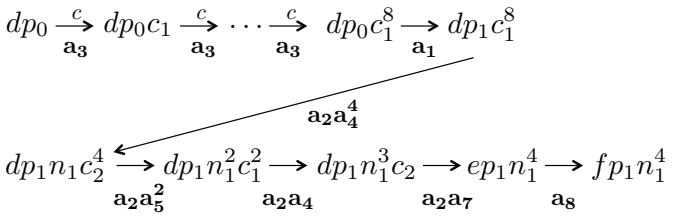

Figure 3.4: Reaction diagram for accepting c^8 in $\mathcal{A}_3$.

Then, it holds that $L_{mp}(\mathcal{A}_3) = \{c^{2^n} \mid n \geq 0\}$. Figure 3.4 illustrates the interactive process in $\mathcal{A}_3$ with the input c^8.

3.2.3 Restricted multistack machines

A multistack machine is a deterministic pushdown automaton with several stacks ([14]). It is known that a two-stack machine is equivalent to a Turing machine (TM) as a language accepting device.

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 chapter, 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 [14]):

(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 Figure 3.5, 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

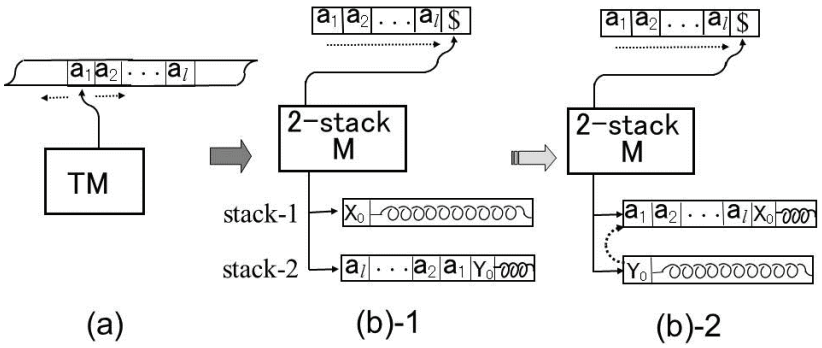

Figure 3.5: (a) Turing machine (TM); (b)Two-stack machine M simulating TM, where \$ is the end marker for the input.

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 [14]) *Every recursively enumerable language is accepted by a restricted two-stack machine.*

3.2.4 Turing machines and variants

In [4], in order to look for a Turing machine corresponding to P automata, a variant of a Turing machine restricted on the workspace, called a restricted $s(n)$ space bounded Turing machine, is introduced. Here, we consider the relaxation of that restriction.

Definition 7. A one-way nondeterministic Turing machine is $s(n)$-restricted if for every accepted input of length n, there is an accepting computation where the number of cells on the worktape *before reading the whole input* is bounded by $s(d)$, where d is the number of input tape cells already read.

The difference between "$s(n)$-restricted" and "restricted $s(n)$ space bounded (in [4])" is that for the case "$s(n)$-restricted", *no restriction* is imposed on the

workspace *after reading the whole input*. We say that a one-way nondeterministic Turing machine M is LOG-restricted, LIN-restricted or NON-restricted if M is logarithmic-restricted, linear function-restricted or not restricted.

Definition 8. $\mathcal{L}_1(\mathcal{X}, \mathcal{Y})$ denotes the class of languages accepted by $\mathcal{X}$-restricted $\mathcal{Y}$-space-bounded one-way nondeterministic Turing machines, where $\mathcal{X}, \mathcal{Y} \in \{LOG, LIN, NON\}$.

Note that (i) $\mathcal{L}_1(NON, \mathcal{Y})$ is the class of language accepted by $\mathcal{Y}$-space-bounded (in usual sense in space complexity theory) one-way nondeterministic Turing machines, (ii) the class of language accepted by restricted $\mathcal{X}$ space bounded one-way nondeterministic Turing machines (defined in [4]) is equivalent to $\mathcal{L}_1(\mathcal{X}, \mathcal{X})$.

Lastly, we introduce a notation about instantaneous descriptions (IDs) of offline Turing machines. For an offline Turing machine $M = (Q, \Sigma, \Gamma, \delta, p_0, F)$ and an input string w, an ID can be expressed by $(w_1 q w_2, x_1 q x_2)$, where $q \in Q$ is the current state, $w_1 w_2 \in \Sigma^*$ is the input string, $x_1 x_2 \in \Gamma^*$ is the content of the work-tape of M, and the head of M points the first symbols of w_2 and x_2. By $ID(M, w)$, we denotes the set of all sequences of the IDs which express computations of M with the input w.

3.3 The computation power of reaction automata

3.3.1 The case of maximally parallel manner

In this section, we shall show the equivalence of the accepting powers between Turing machines and reaction automata in maximally parallel manner. Taking Proposition 1 into consideration, it should be enough 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 in maximally parallel manner.*

[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, repsectively, 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 :

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

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

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

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

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

(4) $A_X = \{(X_k^2, \hat{Q} \cup \hat{\Gamma} \cup (Lab(\delta) - \{r\}) \cup \{f'\}, \hat{X}_k^{2^{|x|}})$

$$| \, 0 \leq k \leq n,\ r : \delta(p, a, X_i, Y_j) = (q, x, y)\},$$

$\hat{A}_X = \{(\hat{X}_k^2, Q \cup \Gamma \cup (Lab(\delta) - \{r\}) \cup \{f'\}, X_k^{2^{|x|}})$

$$| \, 0 \leq k \leq n,\ r : \delta(p, a, X_i, Y_j) = (q, x, y)\},$$

$A_Y = \{(Y_k^2, \hat{Q} \cup \hat{\Gamma} \cup (Lab(\delta) - \{r\}) \cup \{f'\}, \hat{Y}_k^{2^{|y|}})$

$$| \, 0 \leq k \leq m,\ r : \delta(p, a, X_i, Y_j) = (q, x, y)\},$$

$\hat{A}_Y = \{(\hat{Y}_k^2, Q \cup \Gamma \cup (Lab(\delta) - \{r\}) \cup \{f'\}, Y_k^{2^{|y|}})$

$$| \, 0 \leq k \leq m,\ r : \delta(p, a, X_i, Y_j) = (q, x, y)\},$$

(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 $D_0 = p_0 X_0 Y_0$ 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 Figure 3.5. 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 $k \geq 1$, 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 $1 \leq k \leq l-1$) or $\delta(p, \lambda, X_{i1}, Y_{j1}) = (q, x, y)$ (if $l \leq k$) 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 $\mathcal{A}_M$, $D_k \rightarrow^{a_{k+1}} D_{k+1}$ (if $1 \leq k \leq l-1$) or $D_k \rightarrow D_{k+1}$ (if $l \leq k$), 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

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

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 $En^{mp}_{\mathcal{A}_M}(a_{k+1} + D_k)$ (if $1 \leq k \leq l-1$) or in $En^{mp}_{\mathcal{A}_M}(D_k)$ (if $l \leq k$), i.e., it is enabled by $a_{k+1} + D_k$ (if $1 \leq k \leq l-1$) or D_k (if $l \leq k$) in maximally parallel manner, where

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

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

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

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

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 $D_k \rightarrow^{a_{k+1}} D_{k+1}$ (if $1 \leq k \leq l-1$) or $D_k \rightarrow D_{k+1}$ (if $l \leq k$) 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

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

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 $\pi : D_0, \ldots, D_i, \ldots \in IP_{mp}(\mathcal{A}_M, w)$ such that D_{k-1} contains f or $\hat{f}$, D_k contains f', and π converges on D_k, for some $k \geq 1$. Therefore, we have that $L(M) = L_{mp}(\mathcal{A}_M)$ holds.

□

Corollary 1. *Every recursively enumerable language is accepted by a reaction automaton in maximally parallel manner.*

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 9. 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 in maximally parallel manner.*

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 $\mathcal{A}'_M = (S \cup La\hat{b}(\delta), \Sigma, A', D_0, f')$, where A' consists of the following 5

categories :

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

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

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

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

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

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

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

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

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

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

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

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 $L(M) = L_{mp}(\mathcal{A}'_M)$ is proved in the manner similar to the proof of Theorem 1. □

Corollary 2. *Every recursively enumerable language is accepted by a deterministic reaction automaton in maximally parallel manner.*

3.3.2 The case of sequential manner

In this section, we shall show the equivalence of the accepting powers between Turing machines and reaction automata 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 3. *Every recursively enumerable language is accepted by a reaction automaton with λ-input mode in sequential manner.*

[Construction of an RA]

Let $M = (Q, \Sigma, \Gamma_1, \Gamma_2, \delta, p_0, X_0, Y_0, f)$ be a 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, repsectively, 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 :

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

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

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

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

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

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

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

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

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

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

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 in the sequential mode.

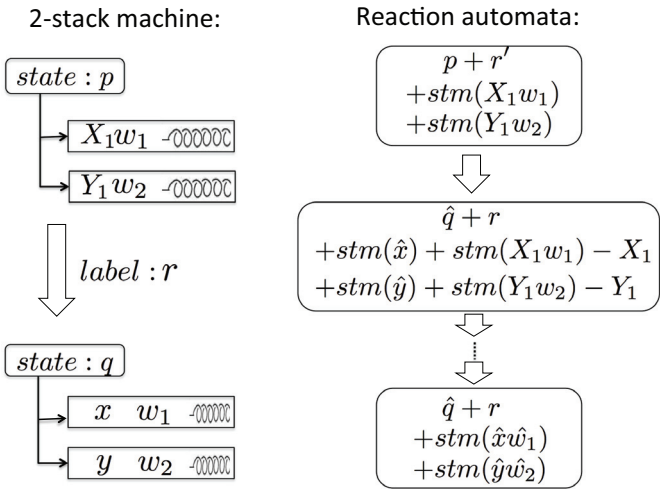

Figure 3.6: Simulation of a 2-stack machine by a sequential RA.

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 $\delta(p, a_k, X_{i1}, Y_{j1}) = (q, x, y)$ or $\delta(p, \lambda, X_{i1}, Y_{j1}) = (q, x, y)$ 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 assume that the multiset before inputting a_k is

$$D_{k-1} = p \cdot stm(X_{i1} X_{i2} \cdots X_{it} X_0) \cdot stm(Y_{j1} Y_{j2} \cdots Y_{js} Y_0) \cdot r',$$

for some $r' \in Lab(\delta)$. Using the reaction in A_a or A_λ, it is possible in $\mathcal{A}_M$, $D_{k-1} \to^{a_k} D_{k_1}$ or $D_{k-1} \to D_{k_1}$, where

$$D_{k_1} = \hat{q} \cdot stm(X_{i1} X_{i2} \cdots X_{it} X_0) \cdot stm(Y_{j1} Y_{j2} \cdots Y_{js} Y_0) \cdot r + stm(\hat{x}) \cdot stm(\hat{y}) - X_{i1} Y_{j1}.$$

Then, we can see that reactions $\mathbf{z} = \mathbf{x_{i2}} \cdots \mathbf{x_{it}}^{2^{t-2}} \mathbf{x_0}^{2^{t-1}} \mathbf{y_{j2}} \cdots \mathbf{y_{js}}^{2^{s-2}} \mathbf{y_0}^{2^{s-1}}$ are enabled in the next $(1 + \cdots + 2^{t-1}) + (1 + \cdots + 2^{s-1})$ steps, in no particular order, where

$\mathbf{x_{i2}}, \ldots, \mathbf{x_{it}}$ is in A_X and $\mathbf{y_{j2}}, \ldots, \mathbf{y_{js}}$ is in A_Y. The result of the reactions $\mathbf{z}$, D_k, is

$$\begin{aligned} D_k &= \hat{q} \cdot stm(\hat{x})stm(\hat{y}) \cdot r \cdot \hat{X}_{i2}^{2^{|x|}} \cdots \hat{X}_{it}^{2^{t-2+|x|}} \hat{X}_0^{2^{t-1+|x|}} \cdot \hat{Y}_{j2}^{2^{|x|}} \cdots \hat{Y}_{js}^{2^{s-2+|x|}} \hat{Y}_0^{2^{s-1+|x|}} \\ &= \hat{q} \cdot stm(\hat{x}\hat{X}_{i2} \cdots \hat{X}_{it}\hat{X}_0) \cdot stm(\hat{y}\hat{Y}_{j2} \cdots \hat{Y}_{js}\hat{Y}_0) \cdot r. \end{aligned}$$

Thus, it holds that $D_{k-1} \rightarrow^{a_k} D_{k_1} \rightarrow \cdots \rightarrow D_k$ or $D_{k-1} \rightarrow D_{k_1} \rightarrow \cdots \rightarrow D_k$ in $\mathcal{A}_M$ in the sequential mode. On the other hand, if the next symbol a_{k+1} is inputted during these steps, $\mathcal{A}_M$ immediately halts and f' is never derived.

We note that there is a possibility that undesired reaction $\mathbf{a}'$ can be enabled by D_{k-1}, where $\mathbf{a}'$ is of the form

$$\begin{cases} \mathbf{a}' &= (pa_kX_{iu}Y_{jv}r', \hat{Q} \cup \hat{\Gamma}, \hat{q}' \cdot stm(\hat{x}')stm(\hat{y}') \cdot r'') \in A_a \\ \mathbf{a}' &= (pX_{iu}Y_{jv}r', \hat{Q} \cup \Sigma \cup \hat{\Gamma}, \hat{q}' \cdot stm(\hat{x}')stm(\hat{y}') \cdot r'') \in A_\lambda, \end{cases}$$

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 reactions $\mathbf{z}' = \mathbf{x'_1} \cdots \mathbf{x'_{t'}}\mathbf{y'_1} \cdots \mathbf{y'_{s'}}$ with $\mathbf{x'_1}, \ldots, \mathbf{x'_{t'}} \in A_X$, $\mathbf{y'_1}, \ldots, \mathbf{y'_{s'}} \in A_Y$ are used in the next $(t' + s')$ steps, then $D_{k_{1+t'+s'}}$ contains *both* the symbols without hat (in Γ) and the symbols with hat (in $\hat{Q}$ and $\hat{\Gamma}$). This is because X_{i1} or Y_{j1} in D_{k-1} remains in D_k (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 by $D_{k_{1+t'+s'}}$ 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 $\pi \in IP^\lambda_{sq}(\mathcal{A}_M, w)$ such that D_{k-1} contains f or $\hat{f}$, D_k contains f', and π converges on D_k, for some $k \geq 1$. Therefore, we have that $L(M) = L^\lambda_{sq}(\mathcal{A}_M)$ holds. □

Corollary 3. $\mathcal{RA}^\lambda_{sq} = \mathcal{RE}$.

Corollary 4. $PR(\mathcal{RA}_{sq}) = \mathcal{RE}$.

Proof. When λ is inputted to an RA $\mathcal{A}_M = (S, \Sigma, A, D_0, f)$ with λ-input mode, we consider that $\mathcal{A}'_M = (S \cup \{c\}, \Sigma \cup \{c\}, A, D_0, f)$ and the special symbol $c \notin S$ is inputted instead of λ. From the proof of Theorem 3, it obviously holds that $w = a_1 a_2 \cdots a_l \in L^{\lambda}_{sq}(\mathcal{A}_M)$ if and only if $w' = c^{i_0} a_1 c^{i_1} a_2 c^{i_2} \cdots a_l c^{i_l} \in L_{sq}(\mathcal{A}'_M)$ for some $i_0, i_1, i_2, \ldots, i_l \geq 0$.

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

Corollary 5. $\mathcal{RA}_{sq} \subseteq \mathcal{L}_1(LOG, NON)$.

Proof. Let $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be an RA and $\pi = D_0, D_1, \ldots \in IP_{sq}(\mathcal{A}, w)$ for the input $w \in \Sigma^*$ with $|w| = n$. By the same way of the proof of "only if" part of Theorem 3, we construct $M_{\mathcal{A}}$.

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

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

Lemma 1. *It holds that* $\{wh(w) \mid w \in \Sigma^*\} \notin \mathcal{L}_1(LOG, NON)$.

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

We define $C_{\log n}$ as the set of all possible IDs of M before reading w which is a part of the input string. Then, it holds that

$$|C_{\log n}| \leq \sum_{i=1}^{n} (m_1 \cdot (i+1) \cdot (\log i + 1) \cdot (m_2+1)^{\log i}) \leq n \cdot m_1 \cdot (n+1) \cdot (\log n + 1) \cdot (m_2+1)^{\log n}.$$

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

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

Since for any two distinct strings $w_1, w_2 \in \Sigma^n$, $ID_n(w_1)$ and $ID_n(w_2)$ are incomparable and $ID_n(w_1), ID_n(w_2) \subseteq C_{\log n}$, it holds the following inequality (see Figure 3.7):

$$|\{ID_n(w) \,|\, w \in \Sigma^n\}| \leq |C_{\log n}| < |\Sigma^n|.$$

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

Corollary 6. *It holds that* $\{wh(w) \,|\, w \in \Sigma^*\} \notin \mathcal{RA}_{sq}$.

Corollary 7. *There exists a recursively enumerable language which cannot be accepted by any reaction automaton in sequential manner.*

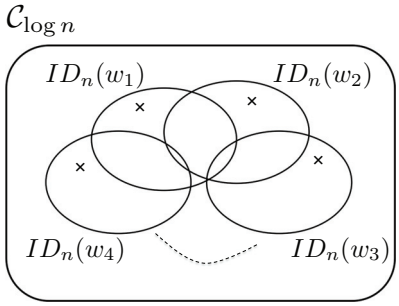

Figure 3.7: Proof sketch of Lemma 1.

3.4 Space complexity issues

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.

3.4.1 Bounded reaction automata

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 ([40]), we define: for $w \in L_X(\mathcal{A})$ with $n = |w|$, and for π in $AIP_X(\mathcal{A}, w)$,

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

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

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

Definition 10. Let s be a function defined on $\mathbf{N}$ and $X = \{sq, mp\}$.
(1) An RA $\mathcal{A}$ is $s(n)$-*bounded* if for any $w \in L_X(\mathcal{A})$ with $n = |w|$, $WS(w, \mathcal{A})$ is bounded by $s(n)$.

(2) If a function $s(n)$ is a constant k (linear, exponential), then $\mathcal{A}$ is termed constant-bounded (resp. linear-bounded, exponential-bounded).
(3) The class of languages accepted by constant-bounded RAs (linear-bounded, polynomial-bounded, exponential-bounded RAs) in X manner is denoted by $\mathcal{CRA}_X$ (resp. $\mathcal{LRA}_X$, $\mathcal{PRA}_X$, $\mathcal{ERA}_X$).
(4) The class of languages accepted by constant-bounded RAs (linear-bounded, polynomial-bounded, exponential-bounded RAs) with λ-input mode in X manner is denoted by $\mathcal{CRA}_X^\lambda$ (resp. $\mathcal{LRA}_X^\lambda$, $\mathcal{PRA}_X^\lambda$, $\mathcal{ERA}_X^\lambda$).

3.4.2 The closure properties of $\mathcal{LRA}_{mp}$

We investigate the closure properties of the class $\mathcal{LRA}_{mp}$ under various language operations. To this aim, it is convenient to prove the following that one may call *normal form lemma* for a bounded class of RAs.

In what follows, we assume that (i) the symbols (such as S, Σ', S_1, S_2, Q,etc.) used in the construction for the background set in the proof denote mutually disjoint sets, and (ii) the symbols (such as p_0, p_1, c, d, f',etc.) are newly introduced in the proof. In addition, we consider RAs *only* in *maximally parallel manner* in Section 3.4.2 and Section 3.4.3.

Definition 11. An $s(n)$-bounded RA $\mathcal{A} = (S, \Sigma, A, D_0, f)$ is said to be in *normal form* if f appears only in a converging state of an interactive process.

Lemma 2. *For an $s(n)$-bounded RA $\mathcal{A} = (S, \Sigma, A, D_0, f)$, there exists an $s(n)$-bounded RA $\mathcal{A}' = (S', \Sigma, A', D_0', f')$ such that $L_{mp}(\mathcal{A}) = L_{mp}(\mathcal{A}')$ and f' appears only in a convergeing state of an interactive process.*

Proof. For an LRA $\mathcal{A} = (S, \Sigma, A, D_0, f)$, construct an RA $\mathcal{A}' = (S', \Sigma, A', D_0', f')$

and a mapping $h : S'^{\#} \to S'^{\#}$ as follows:

$$S' = S \cup \Sigma' \cup \{p_0, p_1, c, d, f'\}, \text{ where } \Sigma' = \{a' | a \in \Sigma\},$$

$$A' = \{(h(R), h(I) \cup f', h(P) + c) \,|\, (R, I, P) \in A\}$$

$$\cup \{(a, \emptyset, a') \,|\, a \in \Sigma\} \cup \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4\}, \text{ where}$$

$$\mathbf{a}_1 = (p_0, \Sigma, p_1), \ \mathbf{a}_2 = (c, \emptyset, \lambda), \ \mathbf{a}_3 = (f, cp_0, f'), \ \mathbf{a}_4 = (d, \emptyset, h(D_0)),$$

$$D'_0 = dp_0,$$

and

$$\begin{cases} h(a) = a' & (\text{for } a \in \Sigma), \\ h(a) = a & (\text{for } a \in S' - \Sigma). \end{cases}$$

Let $w \in \Sigma^*$ with $|w| = n$. Then, there exists an interactive process $\pi = D_0, \ldots, D_m \in AIP_{mp}(\mathcal{A}, w)$ which converges on D_m if and only if there exists $\pi' = D'_0, \ldots, D'_{m+3} \in AIP_{mp}(\mathcal{A}', w)$ which converges on D'_{m+3} such that

$$\begin{cases} D'_1 = h(D_0) + p_0 + a'_1, & \\ D'_{i+1} = h(D_i) + p_0 + c^{j_i} + a'_{i+1} & (\text{for } 1 \le i \le n-1, \text{ and some } j_i \ge 1), \\ D'_{i+1} = h(D_i) + p_1 + c^{k_i} & (\text{for } n \le i \le m, \text{ and some } k_i \ge 1), \\ D'_{m+2} = h(D_m) + p_1, & \\ D'_{m+3} = h(D_m) - f + f'p_1. & \end{cases}$$

Note that (i) $j_i \le |D_{i-1}| \le s(n)$, $k_i \le |D_{i-1}| \le s(n)$. (ii) there may be D_i in π with $f \subseteq D_i$, $0 \le i \le m-1$, but f' cannot be derived from the corresponding state D'_{i+1} in π', since the blocking symbol c exists in D'_{i+2}. Moreover, the workspace of $\mathcal{A}'$ is obviously $s(n)$-bounded. □

Theorem 4. *$\mathcal{LRA}_{mp}$ is closed under union, intersection, concatenation, derivative, λ-free morphisms, λ-free gsm-mappings and shuffle.*

Proof. Let $\mathcal{A}_1 = (S_1, \Sigma, A_1, D_0^{(1)}, f_1)$ and $\mathcal{A}_2 = (S_2, \Sigma, A_2, D_0^{(2)}, f_2)$ be LRAs in normal form with $(S_1 - \Sigma) \cap (S_2 - \Sigma) = \emptyset$. Moreover, let $\Sigma_1 = \{a^{(1)} \,|\, a \in \Sigma\}$,

$\Sigma_2 = \{a^{(2)} \mid a \in \Sigma\}$, $h_1 : {S_1}^{\#} \to {S_1}^{\#}$ and $h_2 : {S_2}^{\#} \to {S_2}^{\#}$ are defined as follows:

$$\begin{cases} h_i(a) = a^{(i)} & (\text{for } a \in \Sigma), \\ h_i(a) = a & (\text{for } a \in S_i - \Sigma), \end{cases}$$

for $i \in \{1, 2\}$. It is important in the proof of "union", "intersection", "concatenation" and "shuffle" parts, that $h_1(S_1)$ and $h_2(S_2)$ are disjoint.

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

$$\begin{aligned} S &= S_1 \cup S_2 \cup \Sigma_1 \cup \Sigma_2 \cup \{d, f\}, \\ A &= \{(h_i(R), h_i(I) \cup \{f\}, h_i(P)) \mid (R, I, P) \in A_i, i \in \{1, 2\}\} \\ &\quad \cup \{(a, \emptyset, a^{(1)}a^{(2)}) \mid a \in \Sigma\} \\ &\quad \cup \{(f_i, \emptyset, f) \mid i \in \{1, 2\}\} \\ &\quad \cup \{(d, \emptyset, h_1(D_0^{(1)}) + h_2(D_0^{(2)})\}, \\ D_0 &= d. \end{aligned}$$

Let $w = a_1 \cdots a_n$ and let $m = min\{m_1, m_2\}$, $m_1, m_2 \geq 0$. Then, for $i = 1$ or $i = 2$, there exists an interactive process $\pi_i = D_0^{(i)}, \ldots, D_{m_i}^{(i)} \in AIP_{mp}(\mathcal{A}_i, w)$ which converges on $D_{m_i}^{(i)}$ if and only if there exists $\pi = D_0, \ldots, D_{m+2} \in AIP_{mp}(\mathcal{A}, w)$ such that

$$\begin{cases} D_{k+1} = h_1(D_k^{(1)}) + h_2(D_k^{(2)}) + a_{k+1}^{(1)} a_{k+1}^{(2)} & (\text{for } 0 \leq k \leq n-1), \\ D_{k+1} = h_1(D_k^{(1)}) + h_2(D_k^{(2)}) & (\text{for } n \leq k \leq m). \end{cases}$$

(Note that either $D_m^{(1)}$ or $D_m^{(2)}$ includes f_1 and f_2, respectively, if and only if D_{m+2} includes f.)

Hence, it holds that $L_{mp}(\mathcal{A}) = L_{mp}(\mathcal{A}_1) \cup L_{mp}(\mathcal{A}_2)$ and the workspace of $\mathcal{A}$ is linear-bounded.

[intersection] In the LRA $\mathcal{A}$ constructed in the proof of "union" part, we replace (i)$\{(f_i, \emptyset, f) \mid i \in \{1, 2\}\}$ by $\{(f_1 f_2, \emptyset, f)\}$, (ii)$m = min\{m_1, m_2\}$ by $m' = max\{m_1, m_2\}$. Then, it is easily seen that that $L_{mp}(\mathcal{A}) = L_{mp}(\mathcal{A}_1) \cap L_{mp}(\mathcal{A}_2)$ holds.

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

$$
\begin{aligned}
S &= S_1 \cup S_2 \cup \Sigma_1 \cup \Sigma_2 \cup \{p_1, p_2, d, f\},\\
A &= \{(h_i(R), h_i(I) \cup \{f\}, h_i(P)) \mid (R, I, P) \in A_i, i \in \{1, 2\}\}\\
&\quad \cup \{(a, p_2, a^{(1)}) \mid a \in \Sigma\} \cup \{(d, \emptyset, h_1(D_0^{(1)}))\}\\
&\quad \cup \{(a, p_1, a^{(2)} \mid a \in \Sigma\} \cup \{(p_1 a, \emptyset, p_2 a^{(2)} + h_2(D_0^{(2)})) \mid a \in \Sigma\}\\
&\quad \cup \{(f_1 f_2, \emptyset, f)\},\\
D_0 &= d p_1.
\end{aligned}
$$

Let $w_1, w_2 \in \Sigma^*$ with $|w_1| = n_1$, $|w_2| = n_2$ and $w_1 w_2 = a_1 \cdots a_n$. Then, for $i = 1$ and $i = 2$, there exists an interactive process $\pi_i = D_0^{(i)}, \ldots, D_{m_i}^{(i)} \in AIP_{mp}(\mathcal{A}_i, w_i)$ which converges on $D_{m_i}^{(i)}$ if and only if there exists $\pi = D_0, \ldots, D_{m_1+m_2+2} \in AIP_{mp}(\mathcal{A}, w_1 w_2)$ such that

$$
\begin{cases}
(i)\, D_{k+1} = h_1(D_k^{(1)}) + p_1 + a_{k+1}^{(1)} & (\text{for } 0 \le k \le n_1 - 1),\\
(ii)\, D_{k+1} = h_1(D_k^{(1)}) + h_2(D_{k-n_1}^{(2)}) + p_2 + a_{k+1}^{(2)} & (\text{for } n_1 \le k \le n - 1),\\
(iii)\, D_{k+1} = h_1(D_k^{(1)}) + h_2(D_{k-n_1}^{(2)}) + p_2 & (\text{for } n \le k \le m_1 + m_2).
\end{cases}
$$

Note that for D_k in (i), a rule in $\{(a, p_1, a^{(2)}) \mid a \in \Sigma\}$ and $\{(p_1 a, \emptyset, p_2 a^{(2)} + h_2(D_0^{(2)})) \mid a \in \Sigma\}$ is nondeterministically chosen to be applied in the next step. If a rule in $\{(a, p_1, a^{(2)}) \mid a \in \Sigma\}$ is chosen, D_{k+1} is in (ii).

Hence, it holds that $L_{mp}(\mathcal{A}) = L_{mp}(\mathcal{A}_1) \cdot L_{mp}(\mathcal{A}_2)$ and the workspace of $\mathcal{A}$ is linear-bounded.

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

$$
\begin{aligned}
S &= S_1 \cup S_2 \cup \Sigma_1 \cup \Sigma_2 \cup \{d, f\}, \\
A &= \{(h_i(R), h_i(I) \cup \Sigma_j \cup \{f\}, h_i(P)) \mid (R, I, P) \in A_i, i, j \in \{1, 2\}, i \neq j\} \\
&\quad \cup \{(a, \emptyset, a^{(i)}) \mid a \in \Sigma, i \in \{1, 2\}\} \\
&\quad \cup \{(d, \emptyset, h_1(D_0^{(1)}) + h_2(D_0^{(2)}))\} \cup \{(f_1 f_2, \emptyset, f)\}, \\
D_0 &= d.
\end{aligned}
$$

Let $w_1, w_2 \in \Sigma^*$ with $|w_1| = n_1$, $|w_2| = n_2$ and let $w = a_1 \cdots a_n \in shuf(w_1, w_2)$. Then, for $i = 1$ and $i = 2$, there exists an interactive process $\pi_i = D_0^{(i)}, \ldots, D_{m_i}^{(i)} \in AIP_{mp}(\mathcal{A}_i, w_i)$ which converges on $D_{m_i}^{(i)}$ if and only if there exists $\pi = D_0, \ldots, D_{m_1+m_2+2} \in AIP_{mp}(\mathcal{A}, w)$ such that

$$
\begin{cases}
D_{k+1} = h_1(D_{k'}^{(1)}) + h_2(D_{k-k'}^{(2)}) + a_{k+1}^{(i)} & (\text{for } 0 \leq k \leq n-1), \\
D_{k+1} = h_1(D_{k-n_2}^{(1)}) + h_2(D_{(k-n_1)}^{(2)}) & (\text{for } n \leq k \leq m),
\end{cases}
$$

where $i = 1$ or $i = 2$ and $0 \leq k' \leq k$. Note that $i = 1$ ($i = 2$) means that only π_1 (resp. π_2) advances to the next step and the value of k' (resp. $k - k'$) is increased by one.

Hence, it holds that $L_{mp}(\mathcal{A}) = Shuf(L_{mp}(\mathcal{A}_1), L_{mp}(\mathcal{A}_2))$ and the workspace of $\mathcal{A}$ is linear-bounded.

[right derivative] For an LRA $\mathcal{A} = (S, \Sigma, A, D_0, f)$ in normal form and $x = a_1 \cdots a_n \in \Sigma^+$, construct an RA $\mathcal{A}' = (S', \Sigma, A', D'_0, f')$ and a mapping $h : S'^{\#} \to S'^{\#}$ as follows:

$$
\begin{aligned}
S' &= S \cup \Sigma' \cup Q \cup \{f'\}, \text{ where } \Sigma' = \{a' | a \in \Sigma\}, Q = \{q_i \,|\, 0 \le i \le n\}, \\
A' &= \{(h(R), h(I) \cup \{f'\}, h(P)) \,|\, (R, I, P) \in A\} \\
&\quad \cup \{(a, \emptyset, a') \,|\, a \in \Sigma\} \\
&\quad \cup \{(q_i, \Sigma, a'_{i+1} q_{i+1}) \,|\, 0 \le i \le n-1\} \\
&\quad \cup \{(f q_n, \Sigma, f')\}, \\
D'_0 &= h(D_0) + q_0,
\end{aligned}
$$

and

$$
\begin{cases}
h(a) = a' & (\text{for } a \in \Sigma), \\
h(a) = a & (\text{for } a \in S' - \Sigma).
\end{cases}
$$

Let $wx \in \Sigma^*$ with $w = b_1 \cdots b_l$, $l \ge 1$. Then, there exists an interactive process $\pi = D_0, \ldots, D_m \in AIP_{mp}(\mathcal{A}, wx)$ which converges on D_m if and only if there exists $\pi' = D'_0, \ldots, D'_{m+2} \in AIP_{mp}(\mathcal{A}', w)$ such that

$$
\begin{cases}
D'_{k+1} = h(D_k) + q_0 b'_{k+1} & (\text{for } 0 \le k \le l-1), \\
D'_{k+1} = h(D_k) + q_{k-l+1} a'_{k-l+1} & (\text{for } l \le k \le l+n-1), \\
D'_{k+1} = h(D_k) + q_n & (\text{for } l+n \le k \le m).
\end{cases}
$$

Hence, it holds that $L_{mp}(\mathcal{A})/x = L_{mp}(\mathcal{A}')$ and the workspace of $\mathcal{A}'$ is linear-bounded.

[left derivative] Let $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be an LRA in normal form and $x = a_1 \cdots a_n \in \Sigma^+$ and $\Sigma_i = \{a^{(i)} \,|\, a \in \Sigma\}$ for $1 \le i \le n$, $Q = \{q_i \,|\, 0 \le i \le n\}$. Construct

an RA $\mathcal{A}' = (S', \Sigma, A', D'_0, f)$ and a mapping $h_n : S'^{\#} \to S'^{\#}$ as follows:

$$
\begin{aligned}
S' &= S \cup (\bigcup_{1 \le i \le n} \Sigma_i) \cup Q \cup \{d\}, \\
A' &= \{(h_n(R), h_n(I), h_n(P)) \mid (R, I, P) \in A\} \\
&\quad \cup \{(q_i, \emptyset, a_{i+1}^{(n)} q_{i+1}) \mid 0 \le i \le n-1\} \\
&\quad \cup \{(a, \emptyset, a^{(1)}) \mid a \in \Sigma\}, \\
&\quad \cup \{(a^{(i)}, \emptyset, a^{(i+1)}) \mid a \in \Sigma, 1 \le i \le n-1, n \ge 2\}, \\
&\quad \cup \{(d, \emptyset, h_n(D_0))\}, \\
D'_0 &= dq_0,
\end{aligned}
$$

and

$$
\begin{cases}
h_n(a) = a^{(n)} & (\text{for } a \in \Sigma), \\
h_n(a) = a & (\text{for } a \in S' - \Sigma).
\end{cases}
$$

Let $xw \in \Sigma^*$ with $w = b_1 \cdots b_l$, $l \ge 1$. Then, there exists an interactive process $\pi = D_0, \ldots, D_m \in AIP_{mp}(\mathcal{A}, xw)$ which converges on D_m if and only if there exists $\pi' = D'_0, \ldots, D'_{m+1} \in AIP_{mp}(\mathcal{A}', w)$ such that

$$
\begin{cases}
D'_{k+1} = h_n(D_k) + q_{k+1} a_{k+1}^{(n)} b_1^{(k)} b_2^{(k-1)} \cdots b_k^{(1)} & (\text{for } 1 \le k \le n-1 \text{ and } n \ge 2), \\
D'_{k+1} = h_n(D_k) + q_n b_{k-n+1}^{(n)} b_{k-n}^{(n-1)} \cdots b_k^{(1)} & (\text{for } n \le k \le l+n-1), \\
D'_{k+1} = h_n(D_k) + q_n & (\text{for } l+n \le k \le m).
\end{cases}
$$

Hence, it holds that $x \backslash L_{mp}(\mathcal{A}) = L_{mp}(\mathcal{A}')$ and the workspace of $\mathcal{A}'$ is linear-bounded.

[λ-free gsm-mappings] For an LRA $\mathcal{A} = (S, \Sigma, A, D_0, f)$ in normal form and a gsm-mapping $g = (Q, \Sigma, \Delta, \delta, p_0, F)$, construct an RA $\mathcal{A}' = (S', \Delta, A', D'_0, f')$ and

a mapping $h : S'^{\#} \to S'^{\#}$ as follows:

$$
\begin{aligned}
S' &= S \cup \Sigma' \cup \Delta \cup Q \cup \{c, d, f'\}, \text{ where } \Sigma' = \{a' \mid a \in \Sigma\}, \\
A' &= \{(h(R), h(I) \cup \{c, f'\}, h(P)) \mid (R, I, P) \in A\} \\
&\quad \cup \{(b, \emptyset, b^2) \mid b \in \Delta\} \\
&\quad \cup \{(pc + stm(x), \emptyset, qda) \mid (q, x) \in \delta(p, a)\} \\
&\quad \cup \{(pd + stm(x), \emptyset, qda) \mid (q, x) \in \delta(p, a), |x| = 1\} \\
&\quad \cup \{(f'' f, \Sigma \cup \{c, d\}, f') \mid f'' \in F\} \\
&\quad \cup \{(c, \emptyset, c)\} \cup \{(d, \emptyset, c)\} \cup \{(d, \Sigma, \lambda)\}, \\
D'_0 &= h(D_0) + cp_0.
\end{aligned}
$$

and

$$
\begin{cases}
h(a) = a' & (\text{for } a \in \Sigma), \\
h(a) = a & (\text{for } a \in S' - \Sigma).
\end{cases}
$$

Then, for an input $w = a_1 \cdots a_n$, there exists $\pi : D_0, D_1, \ldots, D_m \in AIP_{mp}(\mathcal{A}, w)$ which converges on D_m, and $g(w) = b_1 \cdots b_{n'}$, where $(p_1, b_1 \cdots b_t) \in \delta(p_0, a_1)$, if and only if there exists the interactive process π' in $\mathcal{A}'$ such that

$$
\begin{aligned}
D'_0 &\to^{b_1} h(D_0) + cp_0 b_1^2 \to^{b_2} \cdots \\
&\to^{b_{t-1}} h(D_0) + cp_0 + stm(b_1 b_2 \cdots b_t) - b_t \\
&\to^{b_t} h(D_0) + dp_0 a'_1 \\
&\to^{b_{t+1}} h(D_1) + cp_1 b_{t+1}^2 \to^{b_{t+2}} \cdots \\
&(\text{or } \to^{b_{t+1}} h(D_1) + dp_1 a'_2 \to^{b_{t+2}} \cdots \text{ if } (q, b_{t+1}) \in \delta(p_1, a_2)) \\
&\to^{b_{n'}} h(D_{n-1}) + df'' a'_n \\
&\to h(D_n) + f'' \\
&\to h(D_n) - f + f' (= D'_q)
\end{aligned}
$$

and D'_q is a converging state in $\mathcal{A}'$. Hence, it holds that $g(L_{mp}(\mathcal{A})) = L_{mp}(\mathcal{A}')$ and the workspace of $\mathcal{A}'$ is linear-bounded.

[λ-free morphisms] Since $\mathcal{LRA}_{mp}$ is closed under λ-free gsm-mappings, it is also closed under λ-free morphisms. □

In order to prove some of the negative closure properties of $\mathcal{LRA}_{mp}$, the following two lemmas are of crucially importance.

Lemma 3. *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 $L = \{wh(w) \mid w \in \Sigma^*\}$.*

Proof. Assume that there is a PRA $\mathcal{A} = (S, \Sigma, A, D_0, f)$ such that $L_{mp}(\mathcal{A}) = \{wh(w) \mid w \in \Sigma^*\}$. 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 $\pi \in IP_{mp}(\mathcal{A}, wh(w))$, it holds that $|D_i| \leq p(n)$ for some polynomial $p(n)$ from the definition of a PRA.

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 $I_i(w) = \{D_i \in \mathcal{D}_{p(n)} \mid \pi = D_0, \ldots, D_i, \ldots \in IP_{mp}(\mathcal{A}, w)\} \subseteq \mathcal{D}_{p(n)}$, 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 $IP_{mp}(\mathcal{A}, w)$. From the fact that $L_{mp}(\mathcal{A}) = \{wh(w) \mid w \in \Sigma^*\}$ 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 $w_1, w_2 \in \Sigma^n$, $I_n(w_1) \neq I_n(w_2)$. □

Lemma 4. $L_1 = \{w_1 w_2 \mid w_1, w_2 \in \{a, b\}^*,\ w_1 \neq w_2\} \in \mathcal{LRA}_{mp}$.

Proof. Let $L = \{u_1 s u_2 v_1 t v_2 \mid u_1, u_2, v_1, v_2 \in \{a, b\}^*, |u_1| = |v_1|, |u_2| = |v_2|, s, t \in \{a, b\}, s \neq t\}$ and $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be an LRA defined as follows:

$S = \{a, b, a', b', c_1, c_2, p_0, p_1, p_2, p_3, f\}$ with $\Sigma = \{a, b\}$,

$A = \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5, \mathbf{a}_6, \mathbf{a}_7, \mathbf{a}_8, \mathbf{a}_9, \mathbf{a}_{10}, \mathbf{a}_{11}, \mathbf{a}_{12}, \mathbf{a}_{13}, \mathbf{a}_{14}, \mathbf{a}_{15}\}$, where

$\mathbf{a}_1 = (p_0 a, \emptyset, p_0 c_1)$, $\mathbf{a}_2 = (p_0 b, \emptyset, p_0 c_1)$, $\mathbf{a}_3 = (p_0 a, \emptyset, p_1 a')$, $\mathbf{a}_4 = (p_0 b, \emptyset, p_1 b')$,

$\mathbf{a}_5 = (p_1 a, \emptyset, p_1 c_2)$, $\mathbf{a}_6 = (p_1 b, \emptyset, p_1 c_2)$, $\mathbf{a}_7 = (p_1 a, \emptyset, p_2 c_2)$, $\mathbf{a}_8 = (p_1 b, \emptyset, p_2 c_2)$,

$\mathbf{a}_9 = (p_2 a c_1, \emptyset, p_2)$, $\mathbf{a}_{10} = (p_2 b c_1, \emptyset, p_2)$, $\mathbf{a}_{11} = (p_2 a' b, c_1, p_3)$,

$\mathbf{a}_{12} = (p_2 b' a, c_1, p_3)$, $\mathbf{a}_{13} = (p_3 a c_2, \emptyset, p_3)$, $\mathbf{a}_{14} = (p_3 b c_2, \emptyset, p_3)$,

$\mathbf{a}_{15} = (p_3, a b c_2, f)$,

$D_0 = p_0$.

Let $w = u_1 s u_2 v_1 t v_2 \in L$ be an input string. The string w is accepted by $\mathcal{A}$ in the following manner:

1. Applying $\mathbf{a}_1$ and $\mathbf{a}_2$, the length of u_1 is counted by the number of c_1.
2. Applying $\mathbf{a}_3$ or $\mathbf{a}_4$, s is rewritten by s'.
3. Applying $\mathbf{a}_5$, $\mathbf{a}_6$, $\mathbf{a}_7$ and $\mathbf{a}_8$, the length of u_2 is counted by the number of c_2. If $\mathbf{a}_7$ or $\mathbf{a}_8$ is applied, then the interactive process enters the next step.
4. Applying $\mathbf{a}_9$ and $\mathbf{a}_{10}$, it is confirmed that $u_1 = v_1$ by consuming c_1.
5. Applying $\mathbf{a}_{11}$ and $\mathbf{a}_{12}$, it is confirmed that $s \neq t$.
6. Applying $\mathbf{a}_{13}$ and $\mathbf{a}_{14}$, it is confirmed that $u_2 = v_2$ by consuming c_2.

Therefore, it holds that $L = L_{mp}(\mathcal{A})$. Note that $L_1 = L \cup \{w \in \Sigma^* \mid |w| = 2n + 1, n \geq 0\}$. Since $\mathcal{LRA}_{mp}$ is closed under union and includes all regular language, L_1 is in $\mathcal{LRA}_{mp}$. □

Theorem 5. *$\mathcal{LRA}_{mp}$ is not closed under complementation, quotient by regular languages, morphisms or gsm-mappings.*

Proof. From Lemma 4, $L_1 = \{w_1 w_2 \mid w_1, w_2 \in \{a, b\}^*,\ w_1 \neq w_2\} \in \mathcal{LRA}_{mp}$, while from Lemma **??**, $\bar{L_1} = \{w_1 w_2 \mid w_1, w_2 \in \{a, b\}^*,\ w_1 = w_2\} \notin \mathcal{LRA}_{mp}$. Hence, $\mathcal{LRA}_{mp}$ is not closed under complementation. From Corollary 12, it obviously follows that $\mathcal{LRA}_{mp}$ is not closed under quotient by regular languages, morphisms or gsm-mappings. □

3.4.3 The closure properties of $\mathcal{LRA}^{\lambda}_{mp}$

As is seen in the previous section, it remains open whether or not the class $\mathcal{LRA}_{mp}$ is closed under several basic operations such as Kleene closures $(+, *)$ or inverse homomorphism.

In this section, we shall prove that if the λ-move is allowed in the phase of input mode in the transition process of reactions, then the obtained class of languages accepted in that manner shows in turn positive closure properties under those basic operations.

In what follows, we focus on dealing with $\mathcal{LRA}^{\lambda}_{mp}$ and continue investigating the closure properties of the class of languages. As a result, it is shown that the class forms an AFL, i.e., an abstract family of languages.

Theorem 6. *For any LRA $\mathcal{A}$, there exists an LRA $\mathcal{A}'$ such that $L_{mp}(\mathcal{A}) = L^{\lambda}_{mp}(\mathcal{A}')$.*

Proof. Let $\Sigma' = \{a' \mid a \in \Sigma\}$ be a new alphabet. For an LRA $\mathcal{A} = (S, \Sigma, A, D_0, f)$ in normal form, construct an RA $\mathcal{A}' = (S', \Sigma, A', D'_0, f')$ and a mapping $h : S'^{\#} \to S'^{\#}$ as follows:

$$
\begin{aligned}
&S' = S \cup \Sigma' \cup \{p_0, p_1, d, f'\},\\
&A' = \{h(R), h(I) \cup \{f'\}, h(P) \mid (R, I, P) \in A\} \cup \{(a, p_1, a') \mid a \in \Sigma\}\\
&\qquad \cup \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3\}, \text{ where}\\
&\qquad\qquad \mathbf{a}_1 = (d, \Sigma, h(D_0)),\ \ \mathbf{a}_2 = (p_0, \Sigma, p_1),\ \ \mathbf{a}_3 = (f, \Sigma, f'),\\
&D'_0 = dp_0,
\end{aligned}
$$

and

$$
\begin{cases}
h(a) = a' & (\text{for } a \in \Sigma),\\
h(a) = a & (\text{for } a \in S' - \Sigma).
\end{cases}
$$

Note that once λ is inputted before an element $a \in \Sigma$ in an interactive process, a cannot be consumed since p_1 will have to be introduced by $\mathbf{a}_2$ in the next step, which implies that no λ-input is allowed before an element $a \in \Sigma$ in a successful interactive process in $\mathcal{A}'$. □

Definition 12. An $s(n)$-bounded RA $\mathcal{A} = (S, \Sigma, A, D_0, f)$ is said to be in *λ-normal form* if f appears only in a converging state of an interactive process in the λ-input mode.

Lemma 5. *For an $s(n)$-bounded RA $\mathcal{A} = (S, \Sigma, A, D_0, f)$, there exists an $s(n)$-bounded RA $\mathcal{A}' = (S', \Sigma, A', D'_0, f')$ such that $L^{\lambda}_{mp}(\mathcal{A}) = L^{\lambda}_{mp}(\mathcal{A}')$ and f' appears only in a converging state of an interactive process.*

Proof. For an $s(n)$-bounded RA $\mathcal{A} = (S, \Sigma, A, D_0, f)$, construct an RA $\mathcal{A}' = (S', \Sigma, A', D'_0, f')$ and a mapping $h : S'^{\#} \to S'^{\#}$ as follows:

$$
\begin{aligned}
S' &= S \cup \Sigma' \cup \{p_0, p_1, c, d, f'\}, \text{ where } \Sigma' = \{a' | a \in \Sigma\}, \\
A' &= \{(h(R), h(I) \cup f', h(P) + c) \,|\, (R, I, P) \in A\} \\
&\quad \cup \{(a, p_1, a') \,|\, a \in \Sigma\} \cup \{\mathbf{a}_1, \mathbf{a}_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5\}, \text{ where} \\
&\mathbf{a}_1 = (p_0, \Sigma, p_1), \ \ \mathbf{a}_2 = (c, \emptyset, \lambda), \ \ \mathbf{a}_3 = (f, \{c, p_0\} \cup \Sigma, f'), \\
&\mathbf{a}_4 = (d, \emptyset, h(D_0)), \ \ \mathbf{a}_5 = (p_0, \emptyset, p_0), \\
D'_0 &= dp_0,
\end{aligned}
$$

and

$$
\begin{cases}
h(a) = a' & (\text{for } a \in \Sigma), \\
h(a) = a & (\text{for } a \in S' - \Sigma).
\end{cases}
$$

When λ is inputted in an interactive process, $\mathbf{a}_1$ exclusively or $\mathbf{a}_5$ has to be used in the next step. Using $\mathbf{a}_1$ implies that the input of the string terminates, while using

$\mathbf{a}_5$ implies that the input of the string continues. The rest of the key issue is proved in a similar manner to Lemma 2. □

Theorem 7. *$\mathcal{LRA}^{\lambda}_{mp}$ is closed under union, intersection, concatenation, Kleene +, Kleene $*$, derivative, λ-free morphisms, inverse morphisms, λ-free gsm-mappings and shuffle.*

Proof. [union, concatenation and shuffle] Using the same construction as the proof of Theorem 1, the claims are immediately proved.

[intersection] Let $\mathcal{A}_1 = (S_1, \Sigma, A_1, D_0^{(1)}, f_1)$ and $\mathcal{A}_2 = (S_2, \Sigma, A_2, D_0^{(2)}, f_2)$ be LRAs in λ-normal form with $(S_1 - \Sigma) \cap (S_2 - \Sigma) = \emptyset$. Moreover, let $\Sigma_i = \{a^{(i)} \mid a \in \Sigma\}$, $\Sigma_i' = \{a^{(i)'} \mid a^{(i)} \in \Sigma_i\}$ be alphabets and $h_i : {S_i}^{\#} \to {S_i}^{\#}$ be a mapping defined as follows:

$$\begin{cases} h_i(a) = a^{(i)} & (\text{for } a \in \Sigma), \\ h_i(a) = a & (\text{for } a \in S_i - \Sigma), \end{cases}$$

for $i \in \{1, 2\}$.

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

$$\begin{aligned} S &= S_1 \cup S_2 \cup \Sigma_1 \cup \Sigma_2 \cup \Sigma_1' \cup \Sigma_2' \cup \{d, f\}, \\ A &= \{(h_i(R), h_i(I) \cup \Sigma \cup \Sigma_j' \cup \{f\}, h_i(P)) \mid (R, I, P) \in A_i, i \in \{1, 2\}\} \\ &\quad \cup \{(a, \Sigma_1' \cup \Sigma_2', a^{(1)'}a^{(2)'}) \mid a \in \Sigma\} \\ &\quad \cup \{(a^{(i)'}, \Sigma, a^{(i)}) \mid a^{(i)} \in \Sigma_i, i \in \{1, 2\}\} \\ &\quad \cup \{(a^{(i)'}, \Sigma, a^{(i)'}) \mid a^{(i)} \in \Sigma_i, i \in \{1, 2\}\} \\ &\quad \cup \{(d, \emptyset, h_1(D_0^{(1)}) + h_2(D_0^{(1)}))\} \cup \{(f_1 f_2, \Sigma \cup \Sigma_1' \cup \Sigma_2', f)\}, \\ D_0 &= d. \end{aligned}$$

Let $w = a_1 \cdots a_n \in L^{\lambda}_{mp}(\mathcal{A}_1) \cap L^{\lambda}_{mp}(\mathcal{A}_2)$. Moreover, let $D^{(1)}_i \to^{\lambda} \cdots \to^{\lambda} D^{(1)}_j \to^{a_m} D^{(1)}_{j+1}$ be a part of $\pi_1 \in IP^{\lambda}_{mp}(\mathcal{A}_1, w)$ and $D^{(2)}_k \to^{\lambda} \cdots \to^{\lambda} D^{(2)}_l \to^{a_m} D^{(2)}_{l+1}$ be a part of $\pi_2 \in IP^{\lambda}_{mp}(\mathcal{A}_2, w)$ for $1 \leq m \leq n$. We assume that $j - i \leq l - k$. Then, they are imitated in $\pi \in IP^{\lambda}_{mp}(\mathcal{A}, w)$ as follows:

$$
\begin{aligned}
& h_1(D^{(1)}_i) + h_2(D^{(2)}_k) \to^{\lambda} \cdots \to^{\lambda} h_1(D^{(1)}_j) + h_2(D^{(2)}_{k-i+j}) \\
\to^{a_m} & h_1(D^{(1)}_j) + h_2(D^{(2)}_{k-i+j}) + a^{(1)'}_m a^{(2)}_m \\
\to^{\lambda} & \ h_1(D^{(1)}_j) + h_2(D^{(2)}_{k-i+j}) + a^{(1)}_m a^{(2)'}_m \\
\to^{\lambda} & \ h_1(D^{(1)}_j) + h_2(D^{(2)}_{k-i+j+1}) + a^{(1)}_m a^{(2)'}_m \to^{\lambda} \cdots \\
\to^{\lambda} & \ h_1(D^{(1)}_j) + h_2(D^{(2)}_k) + a^{(1)}_m a^{(2)}_m \\
\to^{\lambda} & \ h_1(D^{(1)}_{j+1}) + h_2(D^{(2)}_{k+1}).
\end{aligned}
$$

The other direction of the proof is shown in the similar manner. Hence, it holds that $L^{\lambda}_{mp}(\mathcal{A}) = L^{\lambda}_{mp}(\mathcal{A}_1) \cap L^{\lambda}_{mp}(\mathcal{A}_2)$ and the workspace of $\mathcal{A}$ is linear-bounded.

[Kleene $*$] Let $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be an LRA in λ-normal form and $\Sigma' = \{a' \mid a \in \Sigma\}$. Construct an RA $\mathcal{A}' = (S', \Sigma, A', D'_0, f')$ and a mapping $h : S'^{\#} \to S'^{\#}$ as follows:

$$
\begin{aligned}
S' &= S \cup \Sigma' \cup \{p_0, p_1, d, e, f'', f'\}, \\
A' &= \{(h(R), h(I) \cup \{f', f''\}, h(P)) \mid (R, I, P) \in A\} \\
&\quad \cup \{(h(R), h(I) \cup \Sigma \cup \{f', f''\}, h(P)) \mid (R, I, P) \in A,\ f \subseteq P\} \\
&\quad \cup \{(a, p_1, a') \mid a \in \Sigma\} \\
&\quad \cup \{(a, e, \lambda) \mid a \in (S \cup \Sigma') - \Sigma\} \\
&\quad \cup \{(d, \emptyset, h(D_0) + e)\} \cup \{(p_0, \Sigma, p_1)\} \cup \{(p_0, \emptyset, p_0)\} \\
&\quad \cup \{(ef, \Sigma, f'')\} \cup \{(f'', \{p_1\}, h(D_0) + e)\} \cup \{(f'', \Sigma \cup \{p_0\}, f')\},
\end{aligned}
$$

$$D'_0 = dp_0,$$

and

$$\begin{cases} h(a) = a' & (\text{for } a \in \Sigma), \\ h(a) = a & (\text{for } a \in S' - \Sigma). \end{cases}$$

Let $w_1, w_2 \in \Sigma^*$ with $w_1 = a_1^{(1)} \cdots a_n^{(1)}, w_2 = a_1^{(2)} \cdots a_m^{(2)}$. Then, we can easily see that there exist the interactive processes $D_0, D_1^{(1)}, \ldots D_i^{(1)} \in AIP_{mp}^{\lambda}(\mathcal{A}, w_1)$ and $D_0, D_1^{(2)} \ldots D_j^{(2)} \in AIP_{mp}^{\lambda}(\mathcal{A}, w_2)$ which converge on $D_i^{(1)}$ and $D_j^{(2)}$, respectively, if and only if there exists the interactive process $D'_0, D'_1, \ldots D'_{i+j+4} \in AIP_{mp}^{\lambda}(\mathcal{A}', w_1 w_2)$ such that

$$\begin{cases} D'_{k+1} = h(D_k^{(1)}) + ep_0 & (\text{for } 0 \le k \le i), \\ D'_{i+2} = h(D_i^{(1)}) - f + f'' p_0, & \\ D'_{i+3} = h(D_0) + ep_0, & \\ D'_{k+i+4} = h(D_k^{(2)}) + ep_0 & (\text{for } 0 \le k \le j), \\ D'_{i+j+5} = h(D_j^{(2)}) - f + f'' p_1, & \\ D'_{i+j+6} = fp_1 & \end{cases}$$

Hence, it holds that $w_1 w_2 \in L_{mp}^{\lambda}(\mathcal{A}')$. In a similar manner, we can prove that $w_1 \cdots w_l \in L_{mp}^{\lambda}(\mathcal{A}')$ for $w_1, \ldots, w_l \in L_{mp}^{\lambda}(\mathcal{A})$ and $l \ge 0$. Then, it holds that $L_{mp}^{\lambda}(\mathcal{A})^* = L_{mp}^{\lambda}(\mathcal{A}')$ and the workspace of $\mathcal{A}'$ is linear-bounded.

[Kleene +] For LRAs $\mathcal{A}$ and $\mathcal{A}'$ in the proof of "Kleene $*$" part, it holds that $L_{mp}^{\lambda}(\mathcal{A})^+ = L_{mp}^{\lambda}(\mathcal{A}') \cap \Sigma^+$. Since $\mathcal{LRA}_{mp}^{\lambda}$ is closed under intersection with regular languages, it is also closed under Kleene +.

[right derivative] For an LRA $\mathcal{A} = (S, \Sigma, A, D_0, f)$ in λ-normal form and $x = a_1 \cdots a_n \in \Sigma^+$, construct an RA $\mathcal{A}' = (S', \Sigma, A', D'_0, f')$ and a mapping $h : S'^{\#} \to S'^{\#}$ as follows:

$$S' = S \cup \Sigma' \cup Q \cup \{f'\}, \text{ where } \Sigma' = \{a' | a \in \Sigma\}, Q = \{q_i \,|\, 0 \le i \le n\},$$

$$
\begin{aligned}
A' = &\{(h(R), h(I) \cup \{f'\}, h(P)) \,|\, (R, I, P) \in A\} \\
&\cup \{(a, Q - \{q_0\}, a') \,|\, a \in \Sigma\} \\
&\cup \{(q_i, \Sigma, a'_{i+1} q_{i+1}) \,|\, 0 \leq i \leq n-1\} \\
&\cup \{(q_i, \Sigma, q_i) \,|\, 0 \leq i \leq n\} \\
&\cup \{(f q_n, \Sigma, f')\}, \\
D'_0 = &\, h(D_0) + q_0,
\end{aligned}
$$

and

$$
\begin{cases}
h(a) = a' & (\text{for } a \in \Sigma), \\
h(a) = a & (\text{for } a \in S' - \Sigma).
\end{cases}
$$

Note that because of the inhibitor of a reaction in $\{(a, Q - \{q_0\}, a') \,|\, a \in \Sigma\}$, a reaction in $\{(q_i, \Sigma, a'_{i+1} q_{i+1}) \,|\, 0 \leq i \leq n-1\}$ must be used after feeding the input. Hence, the rest of the proof is similar to the case for the ordinary input mode.

[left derivative] For an LRA $\mathcal{A} = (S, \Sigma, A, D_0, f)$ in λ-normal form and $x = a_1 \cdots a_n \in \Sigma^+$, construct an RA $\mathcal{A}' = (S', \Sigma, A', D'_0, f')$ and a mapping $h : S'^{\#} \to S'^{\#}$ as follows:

$$
\begin{aligned}
S' = &\, S \cup \Sigma' \cup Q \cup \{f'\}, \text{ where } \Sigma' = \{a' | a \in \Sigma\}, Q = \{q_i \,|\, 0 \leq i \leq n\}, \\
A' = &\{(h(R), h(I) \cup \{f'\}, h(P)) \,|\, (R, I, P) \in A\} \\
&\cup \{(a, Q - \{q_n\}, a') \,|\, a \in \Sigma\} \\
&\cup \{(q_i, \Sigma, a'_{i+1} q_{i+1}) \,|\, 0 \leq i \leq n-1\} \\
&\cup \{(q_i, \Sigma, q_i) \,|\, 0 \leq i \leq n\} \\
&\cup \{(f q_n, \Sigma, f')\}, \\
D'_0 = &\, h(D_0) + q_0,
\end{aligned}
$$

and

$$\begin{cases} h(a) = a' & (\text{for } a \in \Sigma), \\ h(a) = a & (\text{for } a \in S' - \Sigma). \end{cases}$$

Note that because of the inhibitor of a reaction in $\{(a, Q - \{q_n\}, a') \mid a \in \Sigma\}$, each reaction in $\{(q_i, \Sigma, a'_{i+1}q_{i+1}) \mid 0 \leq i \leq n-1\}$ must be used before starting the input except λ.

Let $xw \in \Sigma^*$ with $w = b_1 \cdots b_l$. Then, there exists an interactive process $\pi = D_0, \ldots, D_m \in AIP^{\lambda}_{mp}(\mathcal{A}, xw)$ which converges on D_m if and only if there exists $\pi' = D'_0, \ldots, D'_{m+2} \in AIP^{\lambda}_{mp}(\mathcal{A}', w)$ such that

$$\begin{cases} D'_{i+1} = h(D_i) + q_{k+1}a'_{k+1} & (\text{for } 0 \leq k \leq n-1), \\ D'_{i+1} = h(D_i) + q_n b'_{k-n+1} & (\text{for } n \leq k \leq l+n-1), \\ D'_{i+1} = h(D_i) + q_n, & \end{cases}$$

for some $0 \leq i \leq m$. Hence, it holds that $x \backslash L^{\lambda}_{mp}(\mathcal{A}) = L^{\lambda}_{mp}(\mathcal{A}')$ and the workspace of $\mathcal{A}'$ is linear-bounded.

[inverse morphisms] Let $\mathcal{A} = (S, \Delta, A, D_0, f)$ be an LRA in normal form and $h : \Sigma^* \to \Delta^*$ be a morphism defined as $h(a) = b_{(a,1)} \cdots b_{(a,l)} \in \Delta^*$ or $h(a) = \lambda$, for $a \in \Sigma$ and $|h(a)| = l$. Moreover, let $\Delta' = \{b' \mid b \in \Delta\}$ and $Q = \{q_{(a,i)} \mid a \in \Sigma,\ |h(a)| \geq 2,\ 1 \leq i \leq |h(a)| - 1\}$. Construct an RA $\mathcal{A}' = (S', \Sigma, A', D'_0, f')$ and a mapping $g : S'^{\#} \to S'^{\#}$ as follows:

$$\begin{aligned} S' &= S \cup \Sigma \cup \Delta' \cup Q \cup \{d, f'\}, \\ A' &= \{(g(R), g(I) \cup \{a \in \Sigma \mid |h(a)| = 0\} \cup \{f'\}, g(P)) \mid (R, I, P) \in A\} \\ &\quad \cup \{(a, \emptyset, \lambda) \mid |h(a)| = 0, a \in \Sigma\} \\ &\quad \cup \{(a, \emptyset, b'_{(a,1)}) \mid |h(a)| = 1, a \in \Sigma\} \\ &\quad \cup \{(a, \emptyset, q_{(a,1)}b'_{(a,1)}), (q_{(a,1)}, \Sigma, b'_{(a,2)}) \mid |h(a)| = 2, a \in \Sigma\} \end{aligned}$$

$$\cup \{(a, \emptyset, q_{(a,1)}b'_{(a,1)}), (q_{(a,i)}, \Sigma, q_{(a,i+1)}b'_{(a,i+1)}), (q_{(a,|h(a)|-1)}, \Sigma, b'_{(a,|h(a)|)})$$
$$| \, |h(a)| \geq 3, \ 1 \leq i \leq |h(a)| - 2, a \in \Sigma\}$$
$$\cup \{(q, \emptyset, q) \mid q \in Q\}$$
$$\cup \{(d, \emptyset, g(D_0))\} \cup \{(f, Q \cup \Sigma, f')\},$$
$$D'_0 = d,$$

and

$$\begin{cases} g(a) = a' & (\text{for } a \in \Delta), \\ g(a) = a & (\text{for } a \in S' - \Delta). \end{cases}$$

Let $w = b_{(a_1,1)} \cdots b_{(a_1,|h(a_1)|)} \cdots b_{(a_n,1)} \cdots b_{(a_n,|h(a_n)|)} \in L^{\lambda}_{mp}(\mathcal{A})$. Hence, $a_1 \cdots a_n$ is included in $h^{-1}(w)$. Moreover, let $D_i \rightarrow^{b_{(a_m,1)}} \cdots \rightarrow^{b_{(a_m,|h(a_m)|)}} D_j$ be a part of $\pi \in IP^{\lambda}_{mp}(\mathcal{A}, w)$. For $|h(a_m)| \geq 3$, it is imitated in $\pi' \in IP^{\lambda}_{mp}(\mathcal{A}', w)$ as follows:

$$\begin{aligned} & g(D_{i-1}) \\ \rightarrow^{a_m} & g(D_i)b'_{(a_m,1)}q_{(a_m,1)} \\ \rightarrow^{\lambda} & g(D_{i+1})b'_{(a_m,2)}q_{(a_m,2)} \rightarrow^{\lambda} \cdots \\ & (\text{or } \rightarrow^{\lambda} \ g(D_{i+1})q_{(a_m,1)} \rightarrow^{\lambda} \cdots, \text{ for a } \lambda\text{-input in } \mathcal{A}) \\ \rightarrow^{\lambda} & g(D_{i-1})b'_{(a_m,h(a_m))} \\ \rightarrow^{\lambda} & g(D_i). \end{aligned}$$

The other direction of the proof is shown in the similar manner. Hence, it holds that $h^{-1}(L^{\lambda}_{mp}(\mathcal{A})) = L^{\lambda}_{mp}(\mathcal{A}')$ and the workspace of $\mathcal{A}'$ is linear-bounded.

[λ-free morprhisms] We first show that $\mathcal{LRA}^{\lambda}_{mp}$ is closed under codings. For an LRA $\mathcal{A} = (S, \Sigma, A, D_0, f)$ in λ-normal form and a coding $h : \Sigma^* \rightarrow \Delta^*$, con-

struct an RA $\mathcal{A}' = (S', \Delta, A', D'_0, f')$ and a mapping $h : S'^{\#} \to S'^{\#}$ as follows:

$$
\begin{aligned}
S' &= S \cup \Sigma' \cup \Delta \cup \{d\}, \text{ where } \Sigma' = \{a' \mid a \in \Sigma\}, \\
A' &= \{(h(R), h(I), h(P)) \mid (R, I, P) \in A\} \\
&\quad \cup \{(h(a), \emptyset, a') \mid a \in \Sigma\} \\
&\quad \cup \{(d, \emptyset, h(D_0))\}, \\
D'_0 &= d,
\end{aligned}
$$

and

$$
\begin{cases}
h(a) = a' & (\text{for } a \in \Sigma), \\
h(a) = a & (\text{for } a \in S' - \Sigma).
\end{cases}
$$

Then, it holds that $h(L^{\lambda}_{mp}(\mathcal{A})) = L^{\lambda}_{mp}(\mathcal{A}')$ and the workspace of $\mathcal{A}'$ is linear-bounded.

In Theorem 3.7.1 of [12], it is shown that each family closed under inverse morphisms, intersection with regular languages and codings is also closed under λ-free morprhisms. Hence, $\mathcal{LRA}^{\lambda}_{mp}$ is closed under λ-free morprhisms.

[λ-free gsm-mappings] Since every trio is closed under λ-free gsm-mappings ([40]), $\mathcal{LRA}^{\lambda}_{mp}$ is closed under λ-free gsm-mappings. □

We shall show that $\mathcal{LRA}^{\lambda}_{mp}$ shares common negative closure properties with $\mathcal{LRA}_{mp}$. The manner of proving those results is almost parallel to that of proofs for $\mathcal{LRA}_{mp}$ presented in the previous section. In order to make this chapter self-contained, below we give the proof of the following lemma that is a λ-version of Lemma 2.

Lemma 6. *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 PRA $\mathcal{A}$ such that $L^{\lambda}_{mp}(\mathcal{A}) = \{wh(w) \mid w \in \Sigma^*\}$.*

Proof. Assume that there is a PRA $\mathcal{A} = (S, \Sigma, A, D_0, f)$ such that $L^{\lambda}_{mp}(\mathcal{A}) = \{wh(w) \mid w \in \Sigma^*\}$. 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 $\pi \in IP^{\lambda}_{mp}(\mathcal{A}, wh(w))$, it holds that $|D_i| \leq p(n)$ for some polynomial $p(n)$ from the definition of a PRA.

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

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

where ${}_{m_1}\mathrm{H}_k$ 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 $w = a_1 \cdots a_n \in \Sigma^*$, let $I(w) = \{D \in \mathcal{D}_{p(n)} \mid \pi = D_0 \to^{a_1} \cdots \to^{a_n} D \to \cdots \in IP^{\lambda}_{mp}(\mathcal{A}, w)\} \subseteq \mathcal{D}_{p(n)}$, i.e., $I(w)$ is the set of multisets in $\mathcal{D}_{p(n)}$ which appear immediately after inputing w in $IP^{\lambda}_{mp}(\mathcal{A}, w)$. From the fact that $L^{\lambda}_{mp}(\mathcal{A}) = \{wh(w) \mid w \in \Sigma^*\}$ and h is an injection, we can show that for any two distinct strings $w_1, w_2 \in \Sigma^n$, $I(w_1)$ and $I(w_2)$ are incomparable. This is because if $I(w_1) \subseteq I(w_2)$, then the string $w_2 h(w_1)$ is in $L^{\lambda}_{mp}(\mathcal{A})$, which means that $h(w_1) = h(w_2)$ and contradicts that h is an injection.

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

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

However, from the pigeonhole principle, the inequality $|\{I(w) \mid w \in \Sigma^n\}| < |\Sigma^n|$ contradicts that for any two distinct strings $w_1, w_2 \in \Sigma^n$, $I(w_1) \neq I(w_2)$. Hence, there is no LRA $\mathcal{A}$ such that $L^{\lambda}_{mp}(\mathcal{A}) = \{wh(w) \mid w \in \Sigma^*\}$. □

Theorem 8. *$\mathcal{LRA}^{\lambda}_{mp}$ is not closed under complementation, quotient by regular languages, morphisms or gsm-mappings.*

Corollary 8. *$\mathcal{LRA}^{\lambda}_{mp}$ is an AFL, but not a full AFL.*

Remark: We note the class $\mathcal{PRA}^{\lambda}_{mp}$ could be proved to be an AFL in the same manner as $\mathcal{LRA}^{\lambda}_{mp}$.

3.4.4 The hierarchy of language classes by reaction automata

In this section, we develop further characterizations concerning the language classes defined by bounded *RA*s in relation to the Chomsky hierarchy.

Theorem 9. *For a language L, L is accepted by an $s(n)$-bounded RA in maximally parallel manner if and only if L is accepted by a* log *$s(n)$-bounded one-way TM.*

Proof. ("if" part) For simulating a log $s(n)$-bounded one-way TM M, we can use the same way as the proof of Theorem 1. Hence, it is enough to consider the workspace of an RA $\mathcal{A}_M$.

From the proof of Proposition 1 in [14], any $f(n)$-bounded 1-way Turing machine can be simulated by an $f(n)$-bounded two-stack machine. Assume that the workspace of 1-way Turing machine M' with $L(M') = L(M)$ is bounded by some function $\log_2 s(n)$. Then, the maximum number of the sum of the workspace of stack-1 and stack-2 of M is also bounded $\log_2 s(n)$. Hence, the workspace of $\mathcal{A}_M$

is at most $1 + 2 + \cdots + 2^{\log_2 s(n)-1} + c = s(n) + c - 1$ for some constant c, which is bounded by $s(n)$.

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

1. At first, Tape-1 has the input $w \in \Sigma^*$ and Tape-$(i + 1)$ has the number of s_i in D_0 (for $1 \leq i \leq k$) represented by the binary number. Tape-$(k + 2)$ is used to count the number of computation step of $M_{\mathcal{A}}$.

2. Let D be the current multiset in π. When $M_{\mathcal{A}}$ reads the symbol s_i in the input, add one to the Tape-$(i + 1)$. Then, by checking all tapes except Tape-1, compute an element of $Res_A^{mp}(s_i + D)$ in the nondeterministic way and rewrite the contents in the tapes. After reading through the input w, $M_{\mathcal{A}}$ computes an element of $Res_A^{mp}(D)$ in the nondeterministic way and rewrite the contents in the tapes.

3. After reading through the input w, if $Res_A^{mp}(D) = \{D\}$ and $f \subseteq D$, then $M_{\mathcal{A}}$ accepts w. In the case where (i) $Res_A^{mp}(D) = \{D\}$ and $f \nsubseteq D$, (ii) $|D|$ exceeds $s(n)$ or (iii) the number of computation step exceeds $c(s(n))^k$ for $k(= |S|)$ and some constant c, $M_{\mathcal{A}}$ rejects w.

Since we use the binary number for counting the number of symbols, the maximum length of each tape to memorize D is $\log_2 s(n)$. In the case where $M_{\mathcal{A}}$ never stops with the input w, there exists a cycle of configurations in the computation of

$M_{\mathcal{A}}$. Since the number of all possible Ds during the computation is bounded by $c(s(n))^k$ for k and some constant c (see the equation $(*)$ in the proof of Lemma 6), the length of Tape-$(k+2)$ to count the number of steps of computation is bounded by $\log_2 c + k \log_2 s(n)$. Therefore, it holds that $L(M_{\mathcal{A}}) = L_{mp}(\mathcal{A})$ and the workspace of $M_{\mathcal{A}}$ is bounded by $\log_2 s(n)$. □

The similar theorem for RAs in sequential manner is easily derived from Theorem 3 and Theorem 9.

Theorem 10. *For a language L, L is accepted by an $s(n)$-bounded RA with λ-input mode in sequential manner if and only if L is accepted by a* $\log s(n)$*-bounded one-way TM.*

Corollary 9. $\mathcal{CS} = \mathcal{ERA}_{mp} = \mathcal{ERA}^{\lambda}_{sq}$.

Next, we consider a representation theorem for the class $\mathcal{RE}$ in terms of $\mathcal{LRA}_{mp}$. For the purpose, we use the following result in [34].

Theorem 11 (Theorem 3.12 in [34])**.** *For any recursively enumerable language $L \subseteq \Sigma^*$, context-sensitive language L' such that $w \in L$ if and only if ${c_2}^i c_1 w \in L'$ (or $w c_1 {c_2}^i \in L'$) for some $i \geq 0$ and $c_1, c_2 \notin \Sigma$.*

Lemma 7. *For any context-sensitive language $L \subseteq \Sigma^*$, there exists an LRA $\mathcal{A}$ such that $w \in L$ if and only if $c^{2^n} w \in L_{mp}(\mathcal{A})$ (or $w c^{2^n} \in L_{mp}(\mathcal{A})$) with $|w| = n$ and $c \notin \Sigma$.*

Proof. From Corollary 9, let $\mathcal{A} = (S, \Sigma, A, D_0, f)$ be an ERA which accepts L.

Then, construct an RA $\mathcal{A}' = (S', \Sigma \cup c, A', D'_0, f')$ as follows:

$$
\begin{aligned}
S' &= S \cup \{p_0, p_1, p_2, p_3, p_4, n_1, n_2, c, c_1, c_2, d, e, f', f''\},\\
A' &= \{R, I \cup \{c, f'\}, P) \mid (R, I, P) \in A\}\\
&\quad \cup \{\mathbf{a}'_1, \mathbf{a}'_2, \mathbf{a}_3, \mathbf{a}_4, \mathbf{a}_5, \mathbf{a}_6, \mathbf{a}_7, \mathbf{a}'_8, \mathbf{a}_9, \mathbf{a}_{10}, \mathbf{a}_{11}, \mathbf{a}_{12}\}, \text{ where}\\
&\mathbf{a}'_1 = (p_0, c, p_1p_2p_3n_2),\ \mathbf{a}'_2 = (p_1, eff'f'', p_1n_1),\ \mathbf{a}_3 = (c, p_1, c_1),\\
&\mathbf{a}_4 = (c_1^2, p_0c_2e, c_2),\ \mathbf{a}_5 = (c_2^2, p_0c_1e, c_1),\ \mathbf{a}_6 = (c_1d, p_0c_2, e),\\
&\mathbf{a}_7 = (c_2d, p_0c_1, e),\ \mathbf{a}'_8 = (e, p_0cc_1c_2, f''),\ \mathbf{a}_9 = (p_2, cp_4, p_2n_2)\\
&\mathbf{a}_{10} = (p_3, \Sigma, p_4),\ \mathbf{a}_{11} = (n_1n_2, \emptyset, \lambda),\ \mathbf{a}_{12} = (ff'', p_3n_1n_2, f'),\\
D'_0 &= D_0 + dp_0.
\end{aligned}
$$

We note that the reactions $\mathbf{a}'_1$-$\mathbf{a}'_8$ are almost the same as the ones of Example 4 and the total number of n_1 appearing in a interactive process of $IP_{mp}(\mathcal{A}', c^{2^n}w)$ is $n + 1$ (see Example 4 and Figure 3.4). On the other hand, the total number of n_2 appearing in a interactive process of $IP_{mp}(\mathcal{A}', c^{2^n}w)$ is $|w| + 1$, which is derived by the reactions $\mathbf{a}'_1, \mathbf{a}_9, \mathbf{a}_{10}$. Using the reaction $\mathbf{a}_{11}$, it is confirmed that if $c^{2^n}w$ is accepted by $\mathcal{A}'$, then $n + 1 = |w| + 1$. Hence, it holds that $w \in L_{mp}(\mathcal{A})$ if and only if $c^{2^n}w \in L_{mp}(\mathcal{A}')$ with $|w| = n$.

Since the workspace of $\mathcal{A}$ for w is bounded by an exponential function with respect to the length $|w| = n$, the workspace of $\mathcal{A}'$ for $c^{2^n}w$ is bounded by a linear function with respect to the length $|c^{2^n}w| = 2^n + n$.

For the case wc^{2^n}, we can prove in a similar manner. □

Lemma 8. *For any recursively enumerable language $L \subseteq \Sigma^*$, there exists an LRA $\mathcal{A}$ such that $w \in L$ if and only if ${c_3}^j{c_2}^ic_1w \in L_{mp}(\mathcal{A})$ (or $wc_1{c_2}^i{c_3}^j \in L_{mp}(\mathcal{A})$) for some $i, j \geq 0$ and $c_1, c_2, c_3 \notin \Sigma$.*

Theorem 12. *(i) 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.*
(ii) For any recursively enumerable language L, there exists an LRA $\mathcal{A}$ *such that* $L = h(L_{mp}(\mathcal{A}))$ *for some projection h.*

There are many related works on language acceptors based on multiset rewriting, such as a variant of P systems, called P automata investigated in the literature (e.g., [4],[35],[5]). A P automaton is a finite automata-like computing model in which a configuration comprises a tuple of multisets each of which consists of objects from each membrane region. On receiving an input (a multiset) from the environment at each step of computation, it changes its configuration by making region-wise applications of the equipped rules. An input sequence of multisets is accepted if the transition halts in all regions after reading the whole input, and the language accepted by a P automaton is defined as a mapping image of those accepted multiset sequences. In this sense, reaction automata may also be regarded as a simplified variants of P automata with no membrane structure.

Let us denote the class of languages accepted by P automata with sequential rule applications by $\mathcal{PA}_{sq}$. In [4], it is proved that $\mathcal{L}_1(LOG, LOG) = \mathcal{PA}_{sq}$. Hence, the following corollary holds from Corollary 5.

Corollary 10. $\mathcal{LRA}_{sq} \subseteq \mathcal{L}_1(LOG, LOG) = \mathcal{PA}_{sq}$.

Then, we consider the relation between the language classes accepted by RAs in maximally parallel manner and ones in sequential manner. For the sake of comparing the classes of languages $\mathcal{LRA}_{mp}$ and $\mathcal{RA}_{sq}$, remind Lemma 7.

Theorem 13. $\mathcal{LRA}_{mp}$, $\mathcal{RA}_{sq}$ *and* $\mathcal{CF}$ *are incomparable with one another.*

Proof. ($\mathcal{LRA}_{mp}-(\mathcal{RA}_{sq}\cup\mathcal{CF}) \neq \emptyset$) From Lemma 7, it holds that $L = \{wwc^{2^{2n}} \mid w \in \Sigma^*, |w| = n\} \in \mathcal{LRA}_{mp}$. Let h be an injection such that $h(w) = wc^{2^{2n}}$ with $|w| = n$. On the other hand, from Corollary 6 it is obviously holds that $L \notin \mathcal{RA}_{sq} \cup \mathcal{CF}$.

($\mathcal{RA}_{sq} - (\mathcal{LRA}_{mp} \cup \mathcal{CF}) \neq \emptyset$) Let $\mathcal{CM}$ be the class of all commutative languages. Then, it holds that $\mathcal{CM} \subset \mathcal{RA}_{sq}$ because after counting the number of each symbol appearing in the input, an RA can simulate a TM which accepts a vector of natural numbers. On the other hand, $\mathcal{CM}$ and $\mathcal{CS}(\supset \mathcal{LRA}_{mp})$ is obviously incomparable. Hence, it holds that $\mathcal{RA}_{sq} - (\mathcal{LRA}_{mp} \cup \mathcal{CF}) \neq \emptyset$.

($\mathcal{CF} - (\mathcal{LRA}_{mp} \cup \mathcal{RA}_{sq}) \neq \emptyset$) From Corollary 6 and Lemma 4, $\{ww^R \mid w \in \{a, b\}^*\} \in \mathcal{CF} - (\mathcal{LRA}_{mp} \cup \mathcal{RA}_{sq})$. □

Corollary 11. *It holds that $\mathcal{LRA}_{sq} \subset \mathcal{RA}_{sq}$ and $\mathcal{LRA}_{sq} \subset \mathcal{LRA}_{mp}$.*

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

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

Then, it holds that $L_{sq}(\mathcal{A}) = L_{mp}(\mathcal{A}')$ and $\mathcal{LRA}_{sq} \subseteq \mathcal{LRA}_{mp}$. Using Theorem 13, it is shown that $\mathcal{LRA}_{sq} \subset \mathcal{RA}_{sq}$ and $\mathcal{LRA}_{sq} \subset \mathcal{LRA}_{mp}$. □

Lastly, we consider the hierarchy of the language classes accepted by RAs in maximally parallel manner.

Theorem 14. *The following inclusions hold* :

$$\mathcal{REG} = \mathcal{CRA}_{mp} \subset \mathcal{LRA}_{mp} \subseteq \mathcal{PRA}_{mp} \subset \mathcal{ERA}_{mp} = \mathcal{CS} \subset \mathcal{RA}_{mp} = \mathcal{RE}.$$

Proof. From the definitions, the inclusion $\mathcal{REG} \subseteq \mathcal{CRA}_{mp}$ is straightforward. Conversely, for a given k-bounded RA $\mathcal{A} = (S, \Sigma, A, D_0, f)$ and for $w \in L_{mp}(\mathcal{A})$, there exists a π in $IP_{mp}(\mathcal{A}, w)$ 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 $F = \{D \mid D \in Q, f \subseteq D, Res_A^{mp}(D) = \{D\}\}$, and construct an NFA $M = (Q, \Sigma, \delta, D_0, F)$, where δ is defined by $\delta(D, a) \ni D'$ if $D \rightarrow^a D'$ for $a \in \Sigma \cup \{\lambda\}$. Then, it is seen that $L_{mp}(\mathcal{A}) = L(M)$, and $\mathcal{CRA}_{mp} \subseteq \mathcal{REG}$, thus we obtain that $\mathcal{REG} = \mathcal{CRA}_{mp}$. The other inclusions are all obvious from the definitions. The language $\{a^n b^n \mid n \geq 0\}$ proves the proper inclusion : $\mathcal{REG} \subset \mathcal{LRA}_{mp}$. A proper inclusion $\mathcal{PRA}_{mp} \subset \mathcal{ERA}_{mp}$ is due to that $\{ww^R \mid w \in \{a, b\}^*\} \in \mathcal{ERA}_{mp} - \mathcal{PRA}_{mp}$, which follows from Lemma 3. □

Note that we can prove $\mathcal{REG} = \mathcal{CRA}_{sq}$ in a similar way.

3.5 Discussion

Based on the formal framework presented in a series of papers [6, 7, 8, 9, 10], we have introduced the notion of reaction automata and investigated the language accepting powers of the automata. Roughly, a reaction automaton may be characterized as a *language accepting device* based on the *multiset rewriting*.

We have investigated RAs with a focus on the two ways of rule applications, maximally parallel manner and sequential manner. Considering some restrictions on the workspace and λ-input mode, we have introduced the classes of languages accepted by the variants of RAs, and investigated the computational powers and the closure properties of them. In order to explore Turing machines (TMs) corresponding to those classes of RAs, we have also introduced a new variant of TMs with restricted workspace, called $s(n)$-restricted TMs.

Table 3.1: Closure properties of $\mathcal{LRA}_{mp}$ and $\mathcal{LRA}^{\lambda}_{mp}$.

language operations	$\mathcal{LRA}_{mp}$	$\mathcal{LRA}^{\lambda}_{mp}$
union	Y	Y
intersection	Y	Y
complementation	N	N
concatenation	Y	Y
Kleene +	?	Y
Kleene $*$	?	Y
(right & left) derivative	Y	Y
(right & left) quotient by regular languages	N	N
λ-free morphisms	Y	Y
morphisms	N	N
inverse morphisms	?	Y
λ-free gsm-mappings	Y	Y
gsm-mappings	N	N
shuffle	Y	Y

Table 1 summarizes the results of closure properties of both $\mathcal{LRA}_{mp}$ and $\mathcal{LRA}^{\lambda}_{mp}$, while Figure 3.8 illustrates the relationship between language classes defined by a various types of reaction automata and the Chomsky hierarchy.

Specifically, we have shown that in a computing schema with one-pot solution and a finite number of molecular species, reaction automata can perform the Turing universal computation. The idea behind its computing principle is to simulate the behavior of two pushdown stacks in terms of multiset rewriting with the help of an encoding technique, where the role of the inhibitors in each reaction is effectively utilized.

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

- $\mathcal{LRA}_{mp} \subset \mathcal{LRA}^{\lambda}_{mp}$,

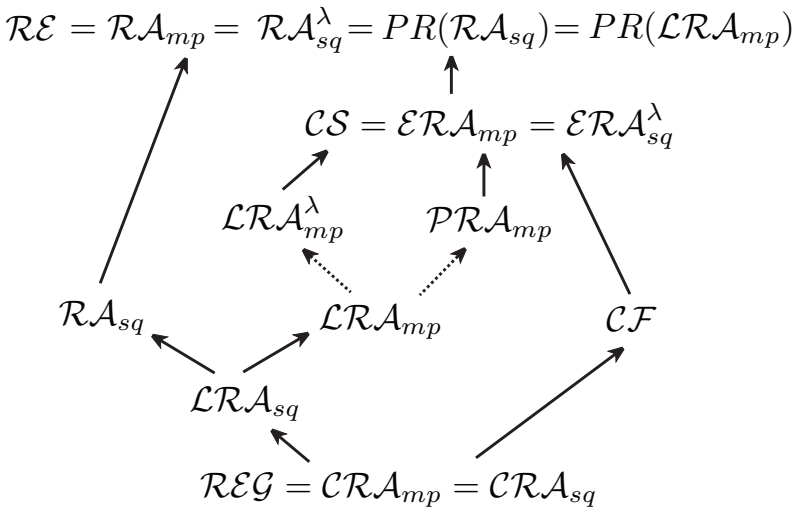

Figure 3.8: The diagram of the relation between the language classes regarding *RA*. A proper inclusion relation is denoted by a solid line and an inclusion relation is denoted by a broken line.

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

Secondly, to explore the computation powers of deterministic reaction automata and time-bounded reaction automata is open and important issues. Lastly, from the viewpoint of designing chemical reactions, it is useful to explore a methodology for "chemical reaction programming" in terms of reaction automata. Further, interesting is to evaluate/simulate a variety of chemical reactions in the real world by the use of the framework of reaction automata.

Chapter 4

Hairpin incompletion

Hairpin completion and its variant called bounded hairpin completion are operations on formal languages, inspired by a hairpin formation in molecular biology. Another variant called hairpin lengthening has been recently introduced, and the related closure properties and algorithmic problems concerning several families of languages have been studied.

In this chapter, we introduce a new operation of this kind, called *hairpin incompletion* which is not only an extension of bounded hairpin completion, but also a restricted (bounded) variant of hairpin lengthening. Further, the hairpin incompletion operation provides a formal language theoretic framework that models a bio-molecular technique nowadays known as Whiplash PCR. We study the closure properties of language families under both the operation and its iterated version.

We show that a family of languages closed under intersection with regular sets, concatenation with regular sets, and finite union is closed under one-sided iterated hairpin incompletion, and that a family of languages containing all linear languages and closed under circular permutation, left derivative and substitution is also closed under iterated hairpin incompletion.

4.1 Introduction

In these years there has been introduced and much investigated an operation called *hairpin completion* in formal language theory, inspired by intra molecular phenomena in molecular biology. A hairpin structure is well-known as one of the most popular secondary structures for a single stranded DNA (or RNA) molecule to form, with the help of so-called *Watson-Crick complementarity* and *annealing*, under a certain biochemical condition in a solution.

This chapter continues research directed by a series of works started in [3] where the hairpin completion operation was introduced, followed by several other related papers ([22, 24, 25]), where both the hairpin completion and its inverse operation (the hairpin reduction) were investigated.

Inspired by threefold motivations, we will introduce the notion of *hairpin incompletion* in this chapter. Firstly, the hairpin incompletion is a natural extension of the notion of *bounded hairpin completion* introduced and studied in [15] which is a restricted variant of the hairpin completion with the property that the length of the prefix (suffix) prolongation is constantly bounded. Thus, the bounded hairpin completion involves the lengthening of prefix (suffix) with a constant length of the strand at the end, which implies that the resulting strand always bears a specific property that its prefix and suffix *always form complementary sub-strands* of a certain constant length. In contrast, our notion of hairpin incompletion can produce a resulting strand with more complexity, due to the nature of its prolongation, which will be formally explained later.

Secondly, the hairpin incompletion is also regarded as a restricted variant of the notion of *hairpin lengthening* recently introduced in [23] which is an extension

of the (original) notion of the hairpin completion. More specifically, the hairpin lengthening concerns the prolongation of a strand that allows to stop itself at any position in the process of completing a hairpin structure. From the practical and molecular implementation point of view, here we are interested in the case where the prolongation in the hairpin lengthening is bounded by a constant, which leads to our notion of the hairpin incompletion. In this respect, one may take the hairpin incompletion as the *bounded* variant of the hairpin lengthening.

Thirdly, the hairpin incompletion can provide a purely formal framework that models a bio-molecular technique called *Whiplash PCR* that has nowadays been recognized as a promising experimental technique and has been proposed in an ingenious paper [13] by Hagiya et al. They developed an experimental technique called polymerization stop and theoretically showed in terms of thermal cycling how DNA molecules can solve the learning problem of μ-formulas (i.e., Boolean formulas with each variable appearing only once) from given data. Suppose that a DNA sequence is designed as given in (a) of Figure 1, where a sequence of transition (program) is delimited by a special sequence (called *stopper sequence*) and α and its reversal complementarity $\bar{\alpha}^R$ may hybridize, leading to a hairpin structure (b). Then, the head $\bar{\alpha}^R$ (current state) is extended by polymerization (with a primer $\bar{\alpha}^R$ and a template γ) up to $\bar{\gamma}^R$, where the stopper sequence is specifically designed to act as the stopper. In this way, this cycle can execute one process of state transition and be repeatedly performed[1]. Following the work of [13], Sakamoto et al. have shown how some NP-complete problems can be solved with Whiplash PCR (or Whiplash machines) ([38]). Recently, Komiya et al. have demonstrated the applicability of Whiplash PCR to the experimental validation of signal dependent

[1] Adleman has named this experimental technique whiplash PCR

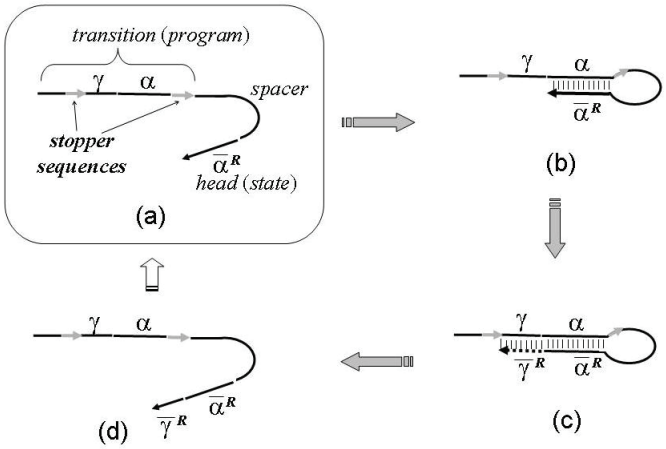

Figure 4.1: (a)The structural design of Whiplash PCR molecule ; (b) hairpin formation with stem part α ; (c) polymerization extension of γ ; (d) simulation of one state transition.

operation ([19]).

The chapter is organized as follows. In Section 4.2, we define the central notion of *hairpin incompletion* (as an extension of the bounded hairpin completion and also as a bounded variant of the hairpin lengthening). We first show in Section 4.3 that any family of languages with certain closure properties is closed under the hairpin incompletion. We then consider the case of applying the *iterated* hairpin incompletion operations, and show that every AFL is closed under the iterated one-sided hairpin incompletion. This result is further extended to the general case of the iterated hairpin incompletion, and it is shown that any family of languages including all linear languages and with certain closure properties is also closed under the iterated hairpin incompletion, and as a corollary that the family of context-free languages is closed under the iterated hairpin incompletion, followed by a brief discussion with concluding remarks in Section 4.4.

4.2 Hairpin incompletion–A bounded variant of hairpin lengthening

For the original definitions of the (unbounded) k-hairpin completion, the reader is referred to precedent papers (for example, [2, 3, 25]). A variant of the notion called bounded k-hairpin completion and its modified operation were introduced and investigated in [15] and [21], respectively, while a recent paper [23] introduces and studies an extended version of the hairpin completion, called hairpin lengthening.

In this chapter, we are interested in a new variant of both the bounded k-hairpin completion and the hairpin lengthening which will be introduced as follows.

An *involution* over V is a bijection $\sigma\colon V \to V$ such that $\sigma = \sigma^{-1}$. In particular, an involution σ over V such that $\sigma(a) \neq a$ for all $a \in V$ is called *Watson-Crick involution*. In this dissertation, we fix an involution $\bar{\cdot}$ over V such that $\overline{\overline{a}} = a$ for $a \in V$ and extend it to V^* in the usual way. Note that for all $x, y \in V^*$, it holds that $(\overline{x})^R = \overline{x^R}$.

Let $m, k \geq 1$. For any $w \in V^*$, we define the *m-bounded k-hairpin incompletion* of w, denoted by $HI_{m,k}(w)$, as follows:

$$rHI_{m,k}(w) = \{w\overline{\gamma}^R \mid w = \delta\gamma\alpha\beta\overline{\alpha}^R,\ |\alpha| = k,\ |\gamma| \leq m,\ \alpha,\beta,\gamma,\delta \in V^*\},$$
$$lHI_{m,k}(w) = \{\overline{\gamma}^R w \mid w = \alpha\beta\overline{\alpha}^R\gamma\delta,\ |\alpha| = k,\ |\gamma| \leq m,\ \alpha,\beta,\gamma,\delta \in V^*\},$$
$$HI_{m,k}(w) = rHI_{m,k}(w) \cup lHI_{m,k}(w).$$

where $rHI_{m,k}$ (or $lHI_{m,k}$) is called *m-bounded right (or left) k-hairpin incompletion*. Moreover, m-bounded right (or left) k-hairpin incompletion is also called m-bounded *one-sided* k-hairpin incompletion. (See Figure 4.2, for pictorial illus-

tration of the operations $rHI_{m,k}$ and $lHI_{m,k}$.) Thus, from a mathematical viewpoint, we consider the hairpin incompletion operations whose prolongations take place at both ends in a hypothetical (and ideal) molecular biological setting.

Note. For $w \in V^*$ not satisfying the condition to apply the m-bounded k-hairpin incompletion, here we assume $r\,HI_{m,k}(w) = l\,HI_{m,k}(w) = \{w\}$.

The iterated version of the m-bounded right k-hairpin incompletion is defined in a usual manner:

$$\begin{cases} rHI^0_{m,k}(w) &= \{w\}, \\ rHI^{n+1}_{m,k}(w) &= rHI_{m,k}(rHI^n_{m,k}(w)) \text{ for } n \geq 0, \\ rHI^*_{m,k}(w) &= \bigcup_{n\geq 0} rHI^n_{m,k}(w). \end{cases}$$

The "left" counterpart of the iterated version of this operation is defined in an obvious and similar manner and is denoted by $lHI^*_{m,k}(w)$.

Further, the iterated version of the m-bounded k-hairpin incompletion operation is defined in a similar manner as follows:

$$\begin{cases} HI^0_{m,k}(w) &= \{w\}, \\ HI^{n+1}_{m,k}(w) &= HI_{m,k}(HI^n_{m,k}(w)) \text{ for } n \geq 0, \\ HI^*_{m,k}(w) &= \bigcup_{n\geq 0} HI^n_{m,k}(w). \end{cases}$$

Finally, the iterated version of the m-bounded (right or left) k-hairpin incompletion operation is naturally extended to languages as follows:

$$\begin{cases} rHI^*_{m,k}(L) &= \bigcup_{w\in L} rHI^*_{m,k}(w), \\ lHI^*_{m,k}(L) &= \bigcup_{w\in L} lHI^*_{m,k}(w), \\ \text{and } HI^*_{m,k}(L) &= \bigcup_{w\in L} HI^*_{m,k}(w). \end{cases}$$

Note that the hairpin *in*completion in this chapter is an extension of bounded hairpin completion in the sense that $HI_{m,k}(w)$ is exactly the same as $mHC_k(w)$ in [15] when the prefix (suffix) δ of w is empty. Further, the hairpin lengthening $HL_k(w)$ in [23] corresponds to the union of all $HI_{m,k}(w)$, where m is arbitrary.

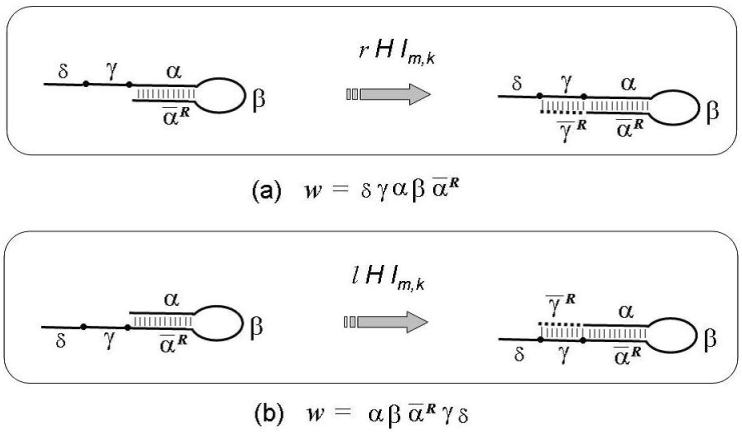

Figure 4.2: (a) m-bounded right k-hairpin incompletion operation ; (b) m-bounded left k-hairpin incompletion operation, where $|\alpha| = k$ and $|\gamma| \leq m$.

4.3 Main Results

4.3.1 Non-iterated hairpin incompletion

As is expected from the definitions, non-iterated hairpin incompletion operation behaves as the bounded hairpin completion operation does.

Theorem 15. *Let $\mathcal{L}$ be a class of languages and $m, k \geq 1$. If $\mathcal{L}$ is closed under gsm-mappings, $\mathcal{L}$ is also closed under m-bounded k-hairpin incompletion.*

Proof. For any $m, k \geq 1$, consider a generalized sequential machine (*gsm*) $g_{m,k}$ which adds a suffix (or prefix) $\overline{\gamma}^R$ of length at most m to the input word w if w is of the form $\delta\gamma\alpha\beta\overline{\alpha}^R$ (or $\alpha\beta\overline{\alpha}^R\gamma\delta$) with $|\alpha| = k$, $|\gamma| \leq m$. It is easily shown that this *gsm* simulates m-bounded k-hairpin incompletion $HI_{m,k}(w)$. □

Since every trio is closed under *gsm* mapping ([**?**]), the following is straightforwardly obtained.

Corollary 12. *Every trio is closed under m-bounded k-hairpin incompletion for any $m, k \geq 1$.*

This result extends the corresponding one (i.e., Proposition 1 in [15]), while it is in contrast to the result that neither the class of regular languages nor of the context-free languages is closed under hairpin lengthening (see [23]).

4.3.2 Iterated one-sided hairpin incompletion

In this section, we consider the closure properties of iterated one-sided hairpin incompletion. Especially, we show that every AFL is closed under this operation. To this aim, we start by introducing some notions required in the proof of the main result. A key idea of the proof is to construct a certain equivalence relation which is right invariant and of finite index.

First, we consider the iterated m-bounded right k-hairpin incompletion operation: $rHI^{*}_{m,k}$.

Definition 13. *Given $m, k \geq 1$ and a word $w \in V^{\geq 2k}$, we define:*

$$C_{m,k}(w) = \{(xy, z) \mid xy \in \bigcup_{0 \leq i \leq m} Inf_{i+k}(w),\ |y| = k,\ w = w_1 xy w_2,\ z \in Suf_{\leq k}(w_2) \cap Pref_{\leq k}(\overline{y}^R)\},$$

$$D_{m,k}(w) = (C_{m,k}(w), \bigcup_{0 \leq i \leq m} \{suf_{i+k-1}(w)\}).$$

We also define a binary relation $\equiv_{D_{m,k}}$ as follows: For $w_1, w_2 \in V^{\geq 2k}$,

$$w_1 \equiv_{D_{m,k}} w_2 \text{ iff } D_{m,k}(w_1) = D_{m,k}(w_2).$$

Intuitively, a pair (xy, z) in $C_{m,k}(w)$ implies that it is a candidate of $(\gamma\alpha, \overline{\alpha}^R)$ where α and γ satisfy the conditions to apply m-bounded right k-hairpin incompletion to w, producing a word in $rHI^{i}_{m,k}(w)$.

From the definition, it holds that $(\gamma\alpha, \overline{\alpha}^R)$ is in $C_{m,k}(w)$ with $|\alpha| = k$ if and only if $w\overline{\gamma}^R$ is in $rHI_{m,k}(w)$.

The binary relation $\equiv_{D_{m,k}}$ is clearly an equivalence relation and of finite index, that is, the number of equivalence classes $|V^{\geq 2k}/\equiv_{D_{m,k}}|$ is finite. Moreover, the following claim holds.

Claim 1. *The equivalence relation $\equiv_{D_{m,k}}$ is right invariant, that is, for $w_1, w_2 \in V^{\geq 2k}$, $w_1 \equiv_{D_{m,k}} w_2$ implies that for any $r \in V^*$, $w_1 r \equiv_{D_{m,k}} w_2 r$.*

Proof. We prove it by induction on the length of r. If $|r| = 0$, then the claim trivially holds. Assume that $w_1 \equiv_{D_{m,k}} w_2$ implies that $w_1 r \equiv_{D_{m,k}} w_2 r$ with $|r| \geq 0$. Then, it suffices to show that for any $a \in V$, $D_{m,k}(w_1 ra) = D_{m,k}(w_2 ra)$.

We observe that $D_{m,k}(w_1 ra)$ is constructed from *only* $D_{m,k}(w_1 r)$ as follows:

$$\begin{aligned}
&\bigcup_{0\leq i\leq m} \{suf_{i+k-1}(w_1 ra)\} \\
&\qquad = \{suf_{i+k-2}(w_1 r)\cdot a \mid 0 \leq i \leq m,\ i+k \geq 2,\ |w_1 r| \geq i+k-2\} \\
&\qquad\quad (\cup\{\lambda\} \text{ if } k = 1),
\end{aligned}$$

$$\begin{aligned}
C_{m,k}(w_1 ra) =& \{(x,\lambda) \mid (x,\lambda) \in C_{m,k}(w_1 r)\} \\
&\cup \{(suf_{i+k-1}(w_1 r)\cdot a, \lambda) \mid 0 \leq i \leq m,\ |w_1 r| \geq i+k-1\} \\
&\cup \{(xy, za) \mid (xy, z) \in C_{m,k}(w_1 r), |y| = k, za \in Pref_{\leq k}(\overline{y}^R)\}.
\end{aligned}$$

Note that if $(xy, z) \in C_{m,k}(w_1 r)$, then $w_1 r = w_1' xy w_1'' z$ for some w', w" $\in V^*$, so that $w_1 ra$ can be rewritten as $w_1' xy w_1'' za$. Therefore, $\{(xy, za) \mid (xy, z) \in C_{m,k}(w_1 r), |y| = k, za \in Pref_{\leq k}(\overline{y}^R)\}$ is contained in $C_{m,k}(w_1 ra)$.

From the induction hypothesis, since $D_{m,k}(w_1 r) = D_{m,k}(w_2 r)$, we can construct $D_{m,k}(w_2 ra)$ from *only* $D_{m,k}(w_1 r)$ in the same way. Thus, it holds that $D_{m,k}(w_1 ra) = D_{m,k}(w_2 ra)$. □

We first show that the language obtained by applying the iterated right hairpin incompletion to a singleton is regular.

[Regular grammar G_w]

Let us consider the equivalence classes:

$$V^{\geq 2k}/\equiv_{D_{m,k}} = \{[w_1], [w_2], \ldots, [w_t] \mid w_i \in V^{\geq 2k}, 1 \leq i \leq t\},$$

where w_i is the representative of $[w_i]$. For $w \in V^{\geq 2k}$, the regular grammar $G_w = (N, V, P, S)$ is constructed as follows:

$$\begin{aligned}
N =& \{S\} \cup \{D_i \mid 1 \leq i \leq t\},\\
P =& \{S \to wD_i \mid w \in V^{\geq 2k}, w \equiv_{D_{m,k}} w_i\}\\
&\cup \{D_i \to rD_j \mid (\gamma\alpha, \overline{\alpha}^R) \in C_{m,k}(w_i), |\alpha| = k,\\
&\qquad r = \overline{\gamma}^R,\ w_i r \equiv_{D_{m,k}} w_j,\ 1 \leq i, j \leq t\}\\
&\cup \{D_i \to \lambda \mid 1 \leq i \leq t\}.
\end{aligned}$$

We need the following two claims.

Claim 2. *Let w be in $V^{\geq 2k}$, and $D_i, D_j \in N$. Then, for $n \geq 0$, if a derivation of G_w is of the form $wD_i \Rightarrow^n wrD_j$ for some $r \in V^*$, then $wr \equiv_{D_{m,k}} w_j$.*

Proof. The proof is by induction on n. If $n = 0$, then $i = j$ and from the manner of constructing P, it holds $w \equiv_{D_{m,k}} w_j$, thus, the claim holds. Assume that the claim holds for $n \geq 0$ and consider a derivation of the form $wD_i \Rightarrow wrD_j \Rightarrow^n wrr'D_h$

for some $D_h \in N$, $r' \in V^*$. From the assumption and the form of P, it holds that $wr \equiv_{D_{m,k}} w_j$ and $w_j r' \equiv_{D_{m,k}} w_h$. By Claim 1, we obtain that $wrr' \equiv_{D_{m,k}} w_j r' \equiv_{D_{m,k}} w_h$. □

Claim 3. *For $n \geq 0$ and $r \in V^*$, there exists a derivation of G_w of the form $S \Rightarrow wD_i \Rightarrow^n wrD_j \Rightarrow wr$ if and only if wr is in $rHI^n_{m,k}(w)$.*

Proof. The proof is by induction on n. If $n = 0$, it obviously holds that $S \Rightarrow wD_i \Rightarrow w$ if and only if w is in $rHI^0_{m,k}(w)$. Assume that the claim holds for n and consider the case for $n + 1$.

(If Part) Let $wr' \in rHI^{n+1}_{m,k}(w)$. Then there exist $r, \gamma \in V^*$ such that $wr' = wr\overline{\gamma}^R \in rHI_{m,k}(wr)$ with $wr \in rHI^n_{m,k}(w)$. From the definition of $C_{m,k}$, $(\gamma\alpha, \overline{\alpha}^R)$ is in $C_{m,k}(wr)$ with $|\alpha| = k$. From the induction hypothesis and Claim 2, there exists a derivation: $S \Rightarrow wD_i \Rightarrow^n wrD_j$ with $wr \equiv_{D_{m,k}} w_j$. Since $(\gamma\alpha, \overline{\alpha}^R)$ is in $C_{m,k}(wr) = C_{m,k}(w_j)$, there exists the derivation $S \Rightarrow wD_i \Rightarrow^n wrD_j \Rightarrow wr\overline{\gamma}^R D_h \Rightarrow wr\overline{\gamma}^R = wr'$ for some $D_h \in N$.

(Only If Part) If there exists the derivation $S \Rightarrow wD_i \Rightarrow^n wrD_j \Rightarrow wr\overline{\gamma}^R D_h \Rightarrow wr\overline{\gamma}^R$ for some $D_h \in N$, it holds that $wr \equiv_{D_{m,k}} w_j$ from Claim 2. Moreover, from the form of P, there exists $(\gamma\alpha, \overline{\alpha}^R) \in C_{m,k}(w_j) = C_{m,k}(wr)$. Hence, $wr\overline{\gamma}^R$ is in $rHI_{m,k}(wr)$. From the induction hypothesis, $wr \in rHI^n_{m,k}(w)$ so that $wr\overline{\gamma}^R \in rHI^{n+1}_{m,k}(w)$. □

It follows from the claim that the language obtained by applying iterated right hairpin incompletion to a singleton is regular.

Lemma 9. *For any word $w \in V^*$ and $m, k \geq 1$, the language $rHI^*_{m,k}(w)$ is regular.*

Proof. In the case of $w \in V^* - V^{\geq 2k}$, from the definition, $rHI^*_{m,k}(w) = \{w\}$ is regular. For $w \in V^{\geq 2k}$ it follows from Claim 3 that there exists a derivation of G_w which derives a terminal string w' if and only if $w' \in rHI^*_{m,k}(w)$. Thus, we have that $L(G_w) = rHI^*_{m,k}(w)$ which is regular. □

In order to show more general results, we need to prove the claims regarding the language $rHI^*_{m,k}(w)$.

Claim 4. *For $w_1, w_2 \in V^{\geq 2k}$ and $n \geq 0$, if $w_1 \equiv_{D_{m,k}} w_2$ then there exists a finite language $F \subseteq V^*$ such that $rHI^n_{m,k}(w_1) = w_1F$ and $rHI^n_{m,k}(w_2) = w_2F$.*

Proof. The proof is by induction on n. If $n = 0$, it obviously holds that $rHI^0_{m,k}(w_1) = w_1F$ and $rHI^0_{m,k}(w_2) = w_2F$, where $F = \{\lambda\}$. We assume that the claim holds for up to n. Let $rHI^n_{m,k}(w_1) = w_1F$ and $rHI^n_{m,k}(w_2) = w_2F$ for some finite language F. For any $r \in F$, it holds that $w_1r \equiv_{D_{m,k}} w_2r$ from Claim 1. Hence, from the induction hypothesis, there exists a finite language F_r such that

$$rHI_{m,k}(w_1r) = w_1rF_r \text{ and } rHI_{m,k}(w_2r) = w_2rF_r.$$

Therefore, it holds that

$$rHI^{n+1}_{m,k}(w_1) = rHI_{m,k}(w_1F) = \bigcup_{r\in F} w_1rF_r = w_1 \bigcup_{r\in F} rF_r = w_1F',$$
$$rHI^{n+1}_{m,k}(w_2) = rHI_{m,k}(w_2F) = \bigcup_{r\in F} w_2rF_r = w_2 \bigcup_{r\in F} rF_r = w_2F',$$

where $F' = \bigcup_{r\in F} rF_r$. □

Claim 5. *For $w_1, w_2 \in V^{\geq 2k}$, if $w_1 \equiv_{D_{m,k}} w_2$ then there exists a regular language $R \subseteq V^*$ such that $rHI^*_{m,k}(w_1) = w_1R$ and $rHI^*_{m,k}(w_2) = w_2R$.*

Proof. From Claim 4, if $w_1 \equiv_{D_{m,k}} w_2$, then there exists a sequence of finite languages: $F_0, F_1, F_2, \cdots$, where $F_n \subseteq V^*(n \geq 0)$, with the property that for $n \geq 0$, $rHI^n_{m,k}(w_1) = w_1F_n$ and $rHI^n_{m,k}(w_2) = w_2F_n$. Then it holds that

$$rHI^*_{m,k}(w_1) = \bigcup_{n\geq 0} rHI^n_{m,k}(w_1) = \bigcup_{n\geq 0} w_1F_n = w_1 \bigcup_{n\geq 0} F_n,$$
$$rHI^*_{m,k}(w_2) = \bigcup_{n\geq 0} rHI^n_{m,k}(w_2) = \bigcup_{n\geq 0} w_2F_n = w_2 \bigcup_{n\geq 0} F_n.$$

Let $R = \bigcup_{n\geq 0} F_n$. Then, we obtain $rHI^*_{m,k}(w_1) = w_1R$ and $rHI^*_{m,k}(w_2) = w_2R$. Recall that w_1R and w_2R are regular from Lemma 9. The class of regular languages is closed under left derivative, so that R is also regular. □

We are now in a position to show the main theorem of this section. It is shown that iterated one-sided hairpin incompletion can be simulated by several basic language operations, which leads to the following theorem.

Theorem 16. *Let $\mathcal{L}$ be a class of languages and $m, k \geq 1$. If $\mathcal{L}$ is closed under intersection with regular languages, concatenation with regular languages and finite union, then $\mathcal{L}$ is also closed under iterated m-bounded right (left) k-hairpin incompletion.*

Proof. Let $L \in \mathcal{L}$ be a language over V. We can write $L = L_1 \cup L_2$ where

$$L_1 = \{w \in L \,|\, |w| \geq 2k\},$$
$$L_2 = \{w \in L \,|\, |w| < 2k\}.$$

Note that $rHI^*_{m,k}(L) = rHI^*_{m,k}(L_1) \cup rHI^*_{m,k}(L_2) = rHI^*_{m,k}(L_1) \cup L_2$. Since the number of the elements in $L_1/\equiv_{D_{m,k}}$ is finite from the definition of $\equiv_{D_{m,k}}$, we can set $L_1/\equiv_{D_{m,k}} = \{[w_1], [w_2], \ldots, [w_s] \,|\, w_i \in L_1 \text{ for } 1 \leq i \leq s\}$ for some $s \geq 0$. From

the way of construction of $D_{m,k}(w_i)$, it holds that for $1 \leq i \leq s$,

$$[w_i] = L_1 \cap (\bigcap_{(xy,z)\in C_{m,k}(w_i)} V^* xyV^* z) \cap (\bigcap_{0\leq j\leq m} V^* \cdot suf_{j+k-1}(w_i)),$$

For $1 \leq i \leq s$, since all words in $[w_i]$ are equivalent, it follows from Claim 5 that there exists a regular language R_i such that $rHI^*_{m,k}([w_i]) = [w_i]R_i$. Moreover, it holds that $rHI^*_{m,k}(L_1) = \bigcup_{1\leq i\leq s}[w_i]R_i$. Thus, $rHI^*_{m,k}(L)$ can be constructed from L by intersection with regular languages, concatenation with regular languages and finite union, which completes the proof. □

As a corollary, we immediately obtain the following.

Corollary 13. *Every AFL is closed under iterated m-bounded right (left) k-hairpin incompletion for any* $m, k \geq 1$.

It is known that there exists *no universal* regular grammar $G_u(x) = (V, \Sigma, P, x)$ with the property that for any regular grammar G, there exists a coding w_G of G such that $L(G) = L(G_u(w_G))$ (see [**?**]). This can be strengthened in the form that no morphism h can help to satisfy the equation $L(G) = h(L(G_u(w_G)))$.

In this context, the next lemma shows that the hairpin incompletion operation can play the role of the universal-like grammar for all regular languages.

Lemma 10. *A language* $L \subseteq V^*$ *is regular if and only if there exists a word* $w \in (V')^*$ *and a weak coding* $h : V' \to V$ *such that* $L = h(rHI^*_{1,1}(w) \cap (V' - \{\#\})^* V'')$, *where* $\# \in V'$ *and* $V'' \subseteq V'$.

Proof. (If Part) This clearly holds, because the class of the regular languages is closed under iterated right hairpin incompletion, intersection and weak codings.

(Only If Part) For a regular grammar $G = (N, V, P, S)$, we construct V',V'', $w \in V$ and $h : V' \to V$ as follows:

- $V' = \{[a, X] \mid a \in V, X \in N \cup \{\lambda\}\} \cup \{\overline{[a, X]} \mid a \in V, X \in N\} \cup \{\#, \overline{\#}\}$,
- $V'' = \{[a, \lambda] \mid a \in V\}$,
- $w = (\prod_{X_i \to aX_j \in P,\, b \in V} \overline{\#}\,\overline{[a, X_j]}\,\overline{[b, X_i]}) \cdot [\lambda, S]$,
- $h(A) = a$ for $A = [a, X] \in \{[a, X] \mid a \in V, X \in N \cup \{\lambda\}\}$, $h(A) = \lambda$ otherwise.

Note that for any $n \geq 0$ and $w' = \delta\gamma\alpha\beta\overline{\alpha}^R \in rHI^n_{1,1}(w) \cap (V' - \{\#\})^*$ with $|\alpha| = |\gamma| = 0$, if $w\overline{\gamma}^R \in rHI^{n+1}_{1,1}(w) \cap (V' - \{\#\})^*$, then γ is the symbol just right of $\overline{\#}$. Then, from the way of construction of w, it holds that there exists a derivation of G,

$$S \Rightarrow a_1X_1 \Rightarrow a_1a_2X_2 \Rightarrow \cdots \Rightarrow a_1a_2 \ldots a_{n-1}X_{n-1} \Rightarrow a_1a_2 \ldots a_{n-1}a_n,$$

if and only if

$$w' = (\prod_{X_i \to aX_j \in P,\, b \in V} \overline{\#}\,\overline{[a, X_j]}\,\overline{[b, X_i]})[\lambda, S][a_1, X_1][a_2, X_2] \ldots [a_{n-1}, X_{n-1}][a_n, \lambda]$$

is in $rHI^n_{1,1}(w) \cap (V' - \{\#\})^*$, which can be shown by induction on n. By applying h, we obtain $L(G) = h(rHI^*_{1,1}(w) \cap (V' - \{\#\})^*V'')$. □

We note that Theorem 3 in [23] proves the *only if* part of this lemma for the iterated (unbounded) hairpin lengthening. Thus, Lemma 10 complements the result for the case of bounded hairpin lengthening.

4.3.3 Iterated hairpin incompletion

In this section, we consider the closure properties of iterated hairpin incompletion. For the (unbounded) hairpin lengthening operation, the paper [23] has proved that the family of context-free languages is closed under iterated hairpin lengthening

in Theorem 4. We will show that the result also holds for the case of iterated bounded hairpin lengthening, in a more general setting of AFL-like formulation.

The proof is based on the similar idea to the previous section and Claims 1, 2, 3 are corresponding to Claims 6, 7, 8 (below), respectively.

In order to consider both-sided hairpin incompletion, we modify the equivalence relation.

Definition 14. *For $m, k \geq 1$ and the word $w \in V^{\geq 2k}$, $C'_{m,k}(w)$, $D'_{m,k}(w)$ and $E_{m,k}(w)$ are defined by*

$$C'_{m,k}(w) = \{(z, yx) \mid yx \in \bigcup_{0 \leq i \leq m} Inf_{i+k}(w),\ |y| = k,\ w = w_1 yx w_2,\ z \in Pref_{\leq k}(w_1) \cap Suf_{\leq k}(\overline{y}^R)\},$$

$$D'_{m,k}(w) = (C'_{m,k}(w), \bigcup_{0 \leq i \leq m} \{pref_{i+k-1}(w)\}),$$

$$E_{m,k}(w) =< D_{m,k}(w), D'_{m,k}(w) >,$$

where $D_{m,k}(w)$ is the relation defined in Definition 13.

The binary relation $\equiv_{E_{m,k}}$ is defined as $w_1 \equiv_{E_{m,k}} w_2$ if $E_{m,k}(w_1) = E_{m,k}(w_2)$ for $w_1, w_2 \in V^{\geq 2k}$.

The binary relation $\equiv_{E_{m,k}}$ is clearly an equivalence relation and of finite index. Note that $D_{m,k}$ and $D'_{m,k}$ are symmetrically defined.

We show that the equivalence relation $\equiv_{E_{m,k}}$ is right invariant and left invariant.

Claim 6. *The equivalence relation $\equiv_{E_{m,k}}$ is right invariant and left invariant, that is, for $w_1, w_2 \in V^{\geq 2k}$, if $w_1 \equiv_{E_{m,k}} w_2$ then for any $r, l \in V^*$, $w_1 r \equiv_{E_{m,k}} w_2 r$ and $l w_1 \equiv_{E_{m,k}} l w_2$ hold.*

Proof. We firstly show that for $r \in V^*$, $w_1 r \equiv_{E_{m,k}} w_2 r$. The proof is by induction on the length of r. If $|r| = 0$, it clearly holds. We assume that the claim holds for n, i.e., $w_1 r \equiv_{E_{m,k}} w_2 r$ with $|r| = n$. Let a be a symbol in V.

[Proof of $D_{m,k}(w_1 ra) = D_{m,k}(w_2 ra)$] It can be shown in the same way as Claim 1.

[Proof of $D'_{m,k}(w_1 ra) = D'_{m,k}(w_2 ra)$] We construct $D'_{m,k}(w_1 ra)$ from *only* $E_{m,k}(w_1 r)$ as follows:

$$\bigcup_{0 \le i \le m} \{pref_{i+k-1}(w_1 ra)\} = \{pref_{i+k-1}(w_1 r) \mid 0 \le i \le m,\ |w_1 r| \ge i + k - 1\} \\ (\cup \{w_1 ra\} \text{ if } |w_1 r| < m + k - 1),$$

$$\begin{aligned} C'_{m,k}(w_1 ra) = C'_{m,k}(w_1 r) \\ \cup\ \{(\lambda, suf_{i+k-1}(w_1 r) \cdot a) \mid 0 \le i \le m,\ |w_1 r| \ge i + k - 1\} \\ \cup\ \{(z, suf_{i+k-1}(w_1 r) \cdot a) \mid 0 \le i \le m,\ |w_1 r| \ge |z| + i + k - 1, \\ z \in Pref_{\le k}(w_1 r) \cap Suf_{\le k}(\overline{suf_{i+k-1}(w_1 r) \cdot a}^R)\}. \end{aligned}$$

Note that for $0 \le i \le m$, $z \in Pref_{\le k}(w_1 r) \cap Suf_{\le k}(\overline{suf_{i+k-1}(w_1 r) \cdot a}^R)$ with $|w_1 r| \ge |z| + i + k - 1$ and some $z' \in V^*$, $w_1 ra$ can be represented as $w_1 ra = z \cdot z' \cdot suf_{i+k-1}(w_1 r) \cdot a$. Hence, $(z, suf_{i+k-1}(w_1 r) \cdot a)$ is in $C'_{m,k}(w_1 ra)$.

Since $E_{m,k}(w_1 r) = E_{m,k}(w_2 r)$, we can construct $D'_{m,k}(w_2 ra)$ from *only* $E_{m,k}(w_1 r)$ in the same way. Therefore, it holds that $D'_{m,k}(w_1 ra) = D'_{m,k}(w_2 ra)$. From $D_{m,k}(w_1 ra) = D_{m,k}(w_2 ra)$ and $D'_{m,k}(w_1 ra) = D'_{m,k}(w_2 ra)$, we eventually get $w_1 ra \equiv_{E_{m,k}} w_2 ra$.

For the left invariance of $\equiv_{E_{m,k}}$, we can proceed in the symmetrical manner. □

[Linear grammar G_L]

For the proof of Theorem 17 (below) regarding m-bounded k-hairpin incompletion, we need to construct a linear grammar. For $L \subseteq V^*$, let $L/\equiv_{E_{m,k}} = \{A_1, A_2, \ldots, A_u\}$ for some $u \geq 1$ and $V^*/\equiv_{E_{m,k}} = \{[w_1], [w_2], \ldots, [w_s]\}$ for some $s \geq 1$, where w_i is the representative of $[w_i]$. A linear grammar $G_L = (N, T, P, S)$ is constructed as follows:

$$
\begin{aligned}
N =& \{S\} \cup \{E_i \mid 0 \leq i \leq s\},\\
T =& V \cup \{a_i \mid 0 \leq i \leq u\} \cup \{\$\},\\
P =& \{S \rightarrow E_i a_j \mid \text{For any } w \in A_j, w \equiv_{E_{m,k}} w_i\}\\
&\cup \{E_i \rightarrow rE_j \mid (\gamma\alpha, \overline{\alpha}^R) \in C_{m,k}(w_i), |\alpha| = k, r = \overline{\gamma}^R, w_i r \equiv_{E_{m,k}} w_j\}\\
&\cup \{E_i \rightarrow E_j l \mid (\overline{\alpha}^R, \alpha\gamma) \in C'_{m,k}(w_i), |\alpha| = k, l = \overline{\gamma}^R, l w_i \equiv_{E_{m,k}} w_j\}\\
&\cup \{E_i \rightarrow \$ \mid 0 \leq i \leq s\}.
\end{aligned}
$$

We set $R_P = \{r \mid E_i \rightarrow rE_j \in P\} \cup \{\lambda\}$ and $L_P = \{l \mid E_i \rightarrow E_j l \in P\} \cup \{\lambda\}$.

Claim 7. *Let $0 \leq p \leq u$ and $E_i, E_j \in N$. For $n \geq 0$, if a derivation of G_L is of the form $E_i a_p \Rightarrow^n r_1 \ldots r_n E_j l_n \ldots l_1 a_p$, then for any $w \in A_p$, it holds that $l_n \ldots l_1 w r_1 \ldots r_n \equiv_{E_{m,k}} w_j$, where for each $1 \leq h \leq n$, $r_h \in R_P$, $l_h \in L_P$, one of r_h and l_h is λ and the other is not λ.*

Proof. The proof is by induction on n. If $n = 0$, then $i = j$ and from the manner of constructing P, for any $w \in A_p$, it holds that $w \equiv_{E_{m,k}} w_j$, thus the claim holds. Assume that the claim holds for $n \geq 0$ and consider a derivation of the form

$$
\begin{aligned}
&E_i a_p \Rightarrow r' E_j a_p \Rightarrow^n r' r_1 \ldots r_n E_h l_n \ldots l_1 a_p\\
&(\, E_i a_p \Rightarrow E_j l' a_p \Rightarrow^n r_1 \ldots r_n E_h l_n \ldots l_1 l' a_p \,)
\end{aligned}
$$

for some $E_h \in N$, $r' \in R_P$ ($l' \in L_P$). From the assumption and the form of P, for any $w \in A_p$, it holds that $wr' \equiv_{E_{m,k}} w_j$ ($l'w \equiv_{E_{m,k}} w_j$) and $l_n \dots l_1 w_j r_1 \dots r_n \equiv_{E_{m,k}} w_h$. By Claim 6, we obtain that

$$l_n \dots l_1 w r' r_1 \dots r_n \equiv_{E_{m,k}} l_n \dots l_1 w_j r_1 \dots r_n \equiv_{E_{m,k}} w_h$$

$$(\, l_n \dots l_1 l' w r_1 \dots r_n \equiv_{E_{m,k}} l_n \dots l_1 w_j r_1 \dots r_n \equiv_{E_{m,k}} w_h \,).$$

□

Claim 8. *A word $r_1 \dots r_n \$ l_n \dots l_1 a_i$ is generated by G_L if and only if for any $w \in A_i$, $l_n \dots l_1 w r_1 \dots r_n$ is in $HI^n_{m,k}(L)$, where for each $1 \le h \le n$, $r_h \in R_P$, $l_h \in L_P$, one of r_h and l_h is λ and the other is not λ.*

Proof. The proof is by induction on n. If $n = 0$, it obviously holds that $S \Rightarrow E_i a_j \Rightarrow \$ a_j$ if and only if for any $w \in A_j$, w is in $HI^0_{m,k}(L)$. Assume that the claim holds for n and consider the case for $n + 1$.

(If Part) Let $l_{n+1} l_n \dots l_1 w r_1 \dots r_n r_{n+1} \in HI^{n+1}_{m,k}(w)$, where for each $1 \le h \le n+1$, $r_h \in R_P$, $l_h \in L_P$, one of r_h and l_h is λ and the other is not λ. From the definition of $C_{m,k}$ and $C'_{m,k}$, either $(\overline{r_{n+1}}^R \cdot \alpha, \overline{\alpha}^R)$ is in $C_{m,k}(l_n \dots l_1 w r_1 \dots r_n)$ or $(\overline{\alpha}^R, \alpha \cdot \overline{l_{n+1}}^R)$ is in $C'_{m,k}(l_n \dots l_1 w r_1 \dots r_n)$ with $|\alpha| = k$. From the induction hypothesis and Claim 7, there exists a derivation:

$$S \Rightarrow E_i a_p \Rightarrow^n r_1 \dots r_n E_j l_n \dots l_1 a_p$$

with $l_n \dots l_1 w r_1 \dots r_n \equiv_{E_{m,k}} w_j$. Therefore, it holds that either $(\overline{r_{n+1}}^R \cdot \alpha, \overline{\alpha}^R) \in C_{m,k}(w_j)$ or $(\overline{\alpha}^R, \alpha \cdot \overline{l_{n+1}}^R) \in C'_{m,k}(w_j)$, from which there exists the derivation either

$$S \Rightarrow E_i a_p \Rightarrow^n r_1 \dots r_n E_j l_n \dots l_1 a_p \Rightarrow r_1 \dots r_n r_{n+1} E_h l_n \dots l_1 a_p$$
$$\Rightarrow r_1 \dots r_n r_{n+1} \$ l_n \dots l_1 a_p$$

or

$$S \Rightarrow E_i a_p \Rightarrow^n r_1 \dots r_n E_j l_n \dots l_1 a_p \Rightarrow r_1 \dots r_n E_h l_{n+1} l_n \dots l_1 a_p$$
$$\Rightarrow r_1 \dots r_n \$ l_{n+1} l_n \dots l_1 a_p$$

for some $E_h \in N$.

(Only If Part) Consider the case where there exists a derivation $S \Rightarrow E_i a_p \Rightarrow^n r_1 \dots r_n E_j l_n \dots l_1 a_p \Rightarrow r_1 \dots r_n r_{n+1} E_h l_n \dots l_1 a_p \Rightarrow r_1 \dots r_n r_{n+1} \$ l_n \dots l_1 a_p$ for some $E_h \in N$. Then, it holds that for any $w \in A_p$, $l_n \dots l_1 w r_1 \dots r_n \equiv_{E_{m,k}} w_j$ from Claim 7. Moreover, from the way of construction of P, there exists $(\overline{r_{n+1}}^R \cdot \alpha, \overline{\alpha}^R) \in C_{m,k}(w_j) = C_{m,k}(l_n \dots l_1 w r_1 \dots r_n)$. Hence, $l_n \dots l_1 w r_1 \dots r_n r_{n+1}$ is in $HI_{m,k}(l_n \dots l_1 w r_1 \dots r_n)$. From the induction hypothesis, $l_n \dots l_1 w r_1 \dots r_n \in HI^n_{m,k}(w)$ so that $l_n \dots l_1 w r_1 \dots r_n r_{n+1} \in HI^{n+1}_{m,k}(w)$.

For the other case, there exists a derivation $S \Rightarrow E_i a_p \Rightarrow^n r_1 \dots r_n E_j l_n \dots l_1 a_p \Rightarrow r_1 \dots r_n E_h l_{n+1} l_n \dots l_1 a_p \Rightarrow r_1 \dots r_n \$ l_{n+1} l_n \dots l_1 a_p$ for some $E_h \in N$. Then we can show in a similar way that for any $w \in A_p$, $l_{n+1} l_n \dots l_1 w r_1 \dots r_n \in HI^{n+1}_{m,k}(w)$.

□

In order to prove the next result, we need a language operation called *circular permutation cp* which maps every word to the set of all its circular permutations and every language to the set of all circular permutations of its words. The proof is due to an idea similar to the one in [15].

Theorem 17. *Let $\mathcal{L}$ be a class of languages which includes all linear languages and let $m, k \geq 1$. If $\mathcal{L}$ is closed under circular permutation, left derivative and substitution, then $\mathcal{L}$ is also closed under iterated m-bounded k-hairpin incompletion.*

Proof. Recall the construction of the linear grammar G_L. Let L be in $\mathcal{L}$ and f be a substitution over T defined by $f(a_i) = A_i$ for $\{a_i \mid 0 \leq i \leq u\}$ and $f(a) = \{a\}$ otherwise. From Claim 8, it holds that

$$L_G = \{r_1 \dots r_n \$ l_n \dots l_1 a_i \mid a_i \in T,\ 1 \leq j \leq n,\ r_j \in R_P,\ l_j \in L_P,\ \text{for any } w \in A_i,\ l_n \dots l_1 w r_1 \dots r_n \in HI^*_{m,k}(L)\},$$

where $L_G = L(G_L)$. Hence, it is easily seen that $HI^*_{m,k}(L) = f(\$ \backslash cp(L_G))$. □

Since the family of context-free languages meets all of preconditions in Theorem 17, the following corollary holds.

Corollary 14. *The family of context-free languages is closed under iterated m-bounded k-hairpin incompletion for any $m, k \geq 1$.*

4.4 Discussion

In many works on DNA-based computing and the related areas, DNA hairpin structures have numerous applications to develop novel computing mechanisms in molecular computing. Among others, these molecules of hairpin formation called Whiplash PCR have been successfully employed as the basic feature of new computational models to solve an instance of the 3-SAT problem ([39]), to execute (and simulate) state transition systems ([38]), to explore the feasibility of parallel computing for solving DHPP ([20]), and so forth. On the other hand, different types of hairpin and hairpin-free languages are defined in [36] and more recently in [16], where they are studied from a language theoretical point of view.

We have proposed a new variant of hairpin completion called hairpin incompletion, and investigated its closure properties of the language families. The hair-

pin incompletion is in fact a bounded variant of the hairpin lengthening in [23] where not only closure properties of language families but also the algorithmic aspects of the hairpin lengthening operations are investigated. The hairpin incompletion is also an extended version of the bounded hairpin completion recently studied in [15] that has been more recently followed up by slightly modified operations in [21] where two open problems from [15] have been solved.

We have shown that every AFL is closed under the iterated one-sided hairpin incompletion, and therefore, the family of regular languages is closed under the operation. Further, it has been shown that the family of context-free languages is closed under the iterated hairpin incompletion. These complement some of the corresponding results for (unbounded) hairpin lengthening operations in [23]. Moreover, since the hairpin incompletion nicely models a bio-molecular technique (Whiplash PCR), the obtained results in this chapter may provide new insight into the computational analysis of the experimental technique.

It remains as an interesting open problem if the family of regular languages is closed under iterated hairpin incompletion.

Chapter 5

Insertion systems

Insertion systems have a unique feature in that only string insertions are allowed, which is in marked contrast to a variety of the conventional computing devices based on string rewriting. This chapter will mainly focus on those systems whose insertion operations are performed in a context-free fashion, called *context-free insertion systems*, and obtain several characterizations of language families with the help of other primitive languages (like star languages) as well as simple operations (like projections, weak-codings). For each $k \geq 1$, a language L is a k-star language if $L = F^+$ for some finite set F with the length of each string in F is no more than k. The results of this kind have already been presented in [33] by Păun et al., while the purpose of this chapter is to prove enhanced versions of them.

5.1 Introduction

In the theory of computing, computation may be considered as regulated rewriting of strings and there exist numerous works investigated in formal language theory that were devoted to string rewriting systems. In contrast, there are several classes of computing devices whose basic operations are based on adjoining and remov-

ing, such as the tree adjoining grammars (see, e.g., [37]), the contextual grammars ([31]) and the insertion-deletion systems ([17]). Among others, research on insertion and deletion operations has a rather old history in both linguistics and formal language theory, and computing models based on insertion-deletion have been recently drawing renewed attention in relation to the theory of DNA computing.

Fortunately, most of those models are shown to be able to characterize the Turing computability (that is, recursively enumerable languages) in a general (unrestricted) framework of computing systems. From the viewpoint of biochemically implementing those computing models, however, it is of crucial importance to investigate the computing power of *context-free* operations of insertion-deletion, because of their simplicity in comparison to the context-dependent counterparts. In fact, recent contributions have been made to explore the computing capability of context-free operations in which inserting and deleting strings are performed independently of the context ([26], [42]).

On the other hand, there are a number of works which have been devoted to characterization/representation theorems of context-free languages. Among others, a well-known Chomsky and Schützenberger characterization (e.g., [37]) states that each context-free language L can be expressed as $h(D \cap R)$ for some projection h, a Dyck set D, and a regular set R. This insight has been recently reformulated as $L = h(L(\gamma) \cap R')$, by using a context-free insertion system γ (instead of a Dyck set) and some simpler regular language R' called "star language", where a star language is given in the form F^*, for some finite set F ([33]). The latter (of star languages) is of interest and simple enough to employ as a member of components to simulate a given computing mechanism based on the context-free rewriting. It should be also noted that a star language is a natural extension of a "finitely gener-

ated free monoid". Returning back to the computing power of insertion-deletion systems, one question arises : how can we achieve a given rewriting mechanism in terms of "context-free insertion" and "free generating monoid", therefore, totally within the framework of the *context-freeness*.

In this chapter we shall provide an answer to the above question, by showing the following characterization of context-free languages that are based on only insertion operations applied in a context-free manner and as small as possible in the length of the inserted string involved. Specifically, it is proved that for each λ-free context-free language L there exist a projection h, a context-free insertion system γ, and a star language F^+ such that $L = h(L(\gamma) \cap F^+)$, where γ only allows inserting at most three symbols in a context-free manner, and the length of each string in F is no more than two. Further, we shall show that a manner of construction used in the proof can be applied to characterize recursively enumerable languages in a similar form of $h(L(\gamma) \cap F^+)$, for some insertion system γ and the same type of F. All of these refine and improve the results for the language families in [33].

5.2 Preliminaries

5.2.1 Insertion systems

Without loss of the essential properties, *we may assume that all of the languages dealt in this chapter are λ-free*.

Definition 1. An *insertion system* is a triple $\gamma = (V, A, P)$, where

- V is an alphabet,
- A is a finite set of strings over V called axioms,

- P is a finite set of triples of the form (u, w, v), for $u, w, v \in V^*$.

A derivation step of an insertion system $\gamma = (V, A, P)$ is defined by the binary relation $\Rightarrow_\gamma$ on V^* such that

$$\alpha \Rightarrow_\gamma \beta \text{ iff } \alpha = \alpha_1 uv\alpha_2,\ \beta = \alpha_1 uwv\alpha_2, \text{ for some } (u, w, v) \in P,\ \alpha_1, \alpha_2 \in V^*.$$

When γ is clear from the context, we simply write $\alpha \Rightarrow \beta$.

The language generated by an insertion system $\gamma = (V, A, P)$ is defined in the usual manner as the set

$$L(\gamma) = \{w \in V^* \mid z \Rightarrow^* w,\ z \in A\},$$

where $\Rightarrow^*$ is the reflexive and transitive closure of $\Rightarrow$.

An insertion system $\gamma = (V, A, P)$ is said to be of *weight* (m, n) if

$$m = max\{|w| \mid (u, w, v) \in P\},$$
$$n = max\{|u| \mid (u, w, v) \in P \text{ or } (v, w, u) \in P\}.$$

By INS^n_m, we denote the family of languages generated by insertion systems of weight (m', n') with $m' \leq m$, $n' \leq n$. When the parameter is not bounded, we replace m or n with $*$.

As for the generating powers of insertion systems, we recall the following results [34]:

- $\mathcal{FIN} \subset INS^0_* \subset INS^1_* \subset \cdots \subset INS^*_* \subset \mathcal{CS}$.
- $\mathcal{REG}$ is incomparable with all INS^n_*, for $n \geq 0$, but $\mathcal{REG} \subset INS^*_*$.
- $\mathcal{CF}$ is incomparable with all INS^n_*, for $n \geq 2$, and INS^*_*.

- $INS^1_* \subseteq \mathcal{CF}$.

- Each regular language is the coding of a language in INS^1_*.

5.2.2 Strictly locally testable languages and star languages

We are going to define strictly locally testable languages and star languages.

For $k \geq 1$, a language over V is *strictly k-testable* if there is a triple $S_k = (A, B, C)$ with $A, B, C \subseteq V^k$ such that for any w with $|w| \geq k$, $w \in L$ iff $pref_k(w) \in A$, $suf_k(w) \in B$, $pInf_k(w) \subseteq C$.

A language L is strictly locally testable iff there exists $k \geq 1$ such that L is strictly k-testable. We denote the class of strictly k-testable languages by $LOC(k)$. In [28], the following theorem is proved.

Theorem 18 ([28])**.** $LOC(1) \subset LOC(2) \subset \cdots \subset LOC(k) \subset \cdots \subset \mathcal{REG}$.

Next, we define a star language. A language L is a *star language*[1] if L is of the form F^+, where F is a finite set of strings. Moreover, for $k \geq 1$ if the maximum length of the string in F is bounded by k, we call L a k-star language. We denote the class of k-star languages by $STAR(k)$.

From the definition of k-star languages, a result analogous to Theorem 1 holds.

Theorem 19. $STAR(1) \subset STAR(2) \subset \cdots \subset STAR(k) \subset \cdots \subset \mathcal{REG}$.

Proof. It is clear from the definition that for $k \geq 1$, $STAR(k) \subseteq STAR(k+1)$ and $STAR(k) \subset \mathcal{REG}$. Then, consider $L = \{a^{k+1}\}^+$ which is in $STAR(k+1)$. L is not in $STAR(k)$, because L contains no strings whose length is less than or equal to k.

□

[1] In the original definition [33], L is a star language if $L = F^*$ for some finite set F.

5.2.3 Labelled derivation trees of context-free grammars

Derivations of a context-free grammar can be represented by trees, called derivation trees. We make a modification on a derivation tree by concatenating the label of the applied context-free rule to each interior node. We call this modified derivation tree a *labelled derivation tree* (*LDT*, in short).

Definition 2. For a context-free grammar $G = (N, T, S, P)$, a *labelled derivation tree* of G is a tree which satisfies the following conditions:

1. The root is labelled by S or Sr, where $r \in Lab(P)$.
2. Each interior node is labelled by Ar, where $A \in N$ and $r \in Lab(P)$.
3. Each leaf is labelled by X, where $X \in N \cup T$.
4. If a interior node labelled by Ar has children $X'_1, X'_2, \ldots, X'_k$ from left to right, then there is a rule $r : A \to X_1 X_2 \ldots X_k \in P$, where $X'_i = X_i$ (if X'_i is a leaf node) or $X'_i = X_i r'$ for some r' in $Lab(P)$ (otherwise) with $1 \leq i \leq k$.

For a context-free grammar $G = (N, T, S, P)$, we denote the set of all LDTs of G by $LD(G)$. An LDT $t \in LD(G)$ is called *complete*, if each leaf of t is labelled by an element of T. The set of all complete LDTs (CLDTs, in short) of G is denoted by $CLD(G)$. For $t \in LD(G)$, *yield* of t, denoted by $yield(t)$, is defined as a label sequence of the leaves of t, in order from left to right. The notion of yield is extended to a set as $yield(LD(G)) = \{yield(t) \mid t \in LD(G)\}$.

Note that $L(G)$, the context-free language generated by G, is nothing but $yield(CLD(G))$.

We also consider a relaxation of Definition 2 and define a *pseudo* LDT (PLDT, in short) as follows :

Definition 3. For a context-free grammar $G = (N, T, S, P)$, a *pseudo* LDT of G is the tree which satisfies the conditions (1), (2), (3) of Definition 2 and (4').

(4') If the interior node labelled by Ar has children $X'_1, X'_2, \ldots, X'_k$ from left to right, then there is a rule $r : B \to X_1 X_2 \ldots X_k \in P$, where $X'_i = X_i$ (if X'_i is a leaf node) or $X'_i = X_i r'$ for some r' in $Lab(P)$ (otherwise) with $1 \leq i \leq k$ (Thus, it is not necessarily the case that $A = B$.).

For a context-free grammar $G = (N, T, S, P)$, we denote by $PLD(G)$ the set of all pseudo LDTs of G.

Finally, we introduce a *preorder traverse sequence* of a binary tree:

Definition 4. A *preorder traverse sequence* of a binary tree t is defined by the following procedure:

Procedure *preorder*(t);
begin
set *preorder*(t) the label of the root of t;
if the root of t has a left subtree t_L,
then substitute $preorder(t) \cdot preorder(t_L)$ for $preorder(t)$;
if the root of t has a right subtree t_R,
then substitute $preorder(t) \cdot preorder(t_R)$ for $preorder(t)$;
end

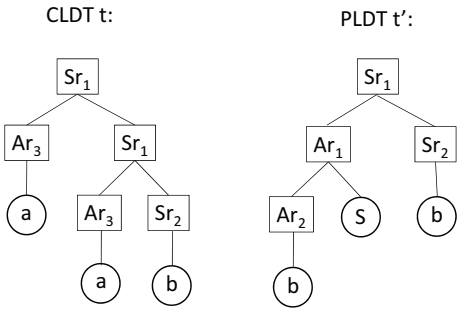

Figure 5.1: Examples of CLDT and PLDT

The notion of a preorder traverse sequence is extended to a set of trees $\mathcal{T}$ in a usual manner, that is, $preorder(\mathcal{T}) = \{preorder(t) \mid t \in \mathcal{T}\}$.

We consider a context-free grammar $G = (N, T, S, P)$ in Chomsky normal form, that is, with rules of the forms $A \to BC$ for $A, B, C \in N$, and $A \to a$ for $A \in N$, $a \in T$. Since an LDT of G is a binary tree, we can consider $preorder(LD(G))$. Note that for a projection h_G defined by $h_G(a) = a$ for $a \in T$, $h_G(a) = \lambda$ otherwise, it holds that $L(G) = h_G(preorder(CLD(G)))$. This is easily seen from the fact that a preorder traverse sequence of CLDT preserves the order of appearance of terminal symbols in the yield of CLDT.

Example. For $G = (\{S, A\}, \{a, b\}, S, \{r_1 : S \to AS, r_2 : S \to b, r_3 : A \to a\})$, we illustrate examples of CLDT t and PLDT t' in Figure 5.1. Here, it holds that

- $preorder(t) = S\, r_1 A r_3 a S\, r_1 A r_3 a S\, r_2 b$,
- $h_G(preorder(t)) = aab$,
- $preorder(t') = S\, r_1 A r_1 A r_2 b S S\, r_2 b$.

5.3 Morphic characterizations of $\mathcal{CF}$

In Theorem 5 of [33], it is proved that $\mathcal{CF} = PR(INS_3^0 \cap STAR(4))$. We now improve this result by reducing $STAR(4)$ to $STAR(2)$.

Lemma 11. $\mathcal{CF} \subseteq PR(INS_3^0 \cap STAR(2))$.

Proof. We need two claims (Claim 1 and Claim 2) to derive the conclusion. For a context-free grammar $G = (N, T, S, P)$ in Chomsky normal form, we construct the insertion system $\gamma = (V, \{S\}, P')$ of weight $(3, 0)$, where

$$V = N \cup T \cup Lab(P),$$

$$P' = \{(\lambda, rBC, \lambda) \mid r : A \to BC \in P\} \cup \{(\lambda, ra, \lambda) \mid r : A \to a \in P\}.$$

Moreover, we construct the 2-star language F^+, where $F = \{Ar \mid r : A \to \alpha \in P\} \cup T$, and the projection h, where $h(a) = a$ for $a \in T$, $h(a) = \lambda$ otherwise.

Claim 9. *It holds that* $preorder(CLD(G)) \subseteq L(\gamma) \cap F^+$.

Proof of Claim 1. Let $w_0(= S) \Rightarrow^{n-1} w_{n-1}(= uAv) \Rightarrow w_n(= u\alpha v)$ in G, where $r : A \to \alpha$ is used in the last step. Moreover, for each $i = 0, 1, \ldots, n$, we denote by t_{w_i} the LDT corresponding to the derivation from S up to w_i.

First, by induction on n, we show that for all $n \geq 0$, $preorder(t_{w_n})$ is derived by γ. If $n = 0$, $preorder(t_S) = S$ is obviously derived by γ. Suppose that the claim holds for up to $(n-1)$. By the induction hypothesis, $preorder(t_{w_{n-1}}) = xAy$ is derived by γ. If $r : A \to \alpha$ is applied to a leaf A in $t_{w_{n-1}}$, A is relabelled with Ar, and Ar has children which are leaves composing α from left to right in t_{w_n}. This implies $preorder(t_{w_n}) = xAr\alpha y$. Since γ has a rule $(\lambda, r\alpha, \lambda)$, $preorder(t_{w_n})$ can be derived by γ.

If t_{w_n} is a CLDT of G, each interior node is of the form Ar, where $r : A \to \alpha \in P$, and each leaf is an element of T. This implies $preorder(t_{w_n}) \in F^+ = \{\{Ar \mid A \to \alpha \in P\} \cup T\}^+$.

Thus, we obtain $preorder(t_{w_n}) \in L(\gamma) \cap F^+$, where $t_{w_n} \in CLD(G)$. □

Before starting the proof of the next claim, we note two observations. A derivation $z_0(= S) \Rightarrow^{n-1} z_{n-1} \Rightarrow z_n$ in γ is said to be *successful*, if $z_n \in L(\gamma) \cap F^+$.

Observation 1. For a successful derivation in γ, any rule of the form $(\lambda, r\alpha, \lambda)$ in P' is only applicable to immediately after (right of) a nonterminal in a sentential form.

This is easily seen as follows:

(1) Once $r\alpha$ is inserted immediately after r' (in $Lab(P)$) in a sentential form, any of the subsequent sentential form always contains a substring "$r'r''$" (for some r'' in $Lab(P)$).

(2) Once $r\alpha$ is inserted immediately after a (in T) in a sentential form, any of the subsequent sentential form always contains a substring "ar''" (for some r'' in $Lab(P)$).

(3) Once $r\alpha$ is inserted (appended) to the top of a sentential form, any of the subsequent sentential form always starts with r'' (for some r'' in $Lab(P)$).

Thus, any of these three cases eventually contradicts the property (of being in F^+) of a successful derivation in γ.

Observation 2. For a successful derivation in γ, no rule of the form (λ, ra, λ) in

P' is applicable to immediately before (left of) r' (of $Lab(P)$) in a sentential form.

This is seen as follows: From the form of rules in P', once ra is inserted immediately before r' (for some r' in $Lab(P)$), any of the subsequent sentential form always contains a substring "rar''" (for some r'' in $Lab(P)$), which eventually contradicts the property of a successful derivation.

We are going to prove Claim 2.

Claim 10. *It holds that* $L(\gamma) \cap F^+ \subseteq preorder(CLD(G))$.

Proof of Claim 2. Let $z_0(= S) \Rightarrow^{n-1} z_{n-1}(= xAy) \Rightarrow z_n(= xAr\alpha y)$ in γ, where $(\lambda, r\alpha, \lambda)$ is used in the last step (From Observation 1, it is sufficient to consider the case where the insertion rule is used immediately after a nonterminal.).

First, by induction on n, we show that for all $n \geq 0$, z_n is in $preorder(PLD(G))$ (Note that here we are dealing with *pseudo* LDTs.). If $n = 0$, S is obviously in $preorder(PLD(G))$. Suppose that the claim holds for up to $(n-1)$. By the induction hypothesis, there exists $t_{n-1} \in PLD(G)$ such that $z_{n-1} = preorder(t_{n-1})$ $(= xAy$, where $x, y \in V^*)$.

(Case 1.) $z_{n-1} = xAy$, $z_n = xArBCy$ with $r : A' \to BC \in P$, and this "A" is a leaf in t_{n-1} (note that it is possible that $A \neq A'$). We construct $t_n \in PLD(G)$ from t_{n-1} by relabelling a node "A" with "Ar" and adding the left and right children of "Ar", "B" and "C", respectively. Here, "B" and "C" are leaves, so that $z_n = preorder(t_n)$ holds. (See Figure 2.)

(Case 2.) $z_{n-1} = xAr'y'$, $z_n = xArBCr'y'$ with $r : A' \to BC \in P$, $r' \in Lab(P)$,

and for this "A" and "r'", "Ar'" is an interior node in t_{n-1}. We construct $t_n \in PLD(G)$ from t_{n-1} by the following steps. (1)Relabel "Ar'" with "Ar". (2)Replace the children of "Ar" with new left child "B" and new right child "Cr'". (3)Add the children of "Cr'", so that its new children may be former children of "Ar'" in t_{n-1}. Here, $z_n = preorder(t_n)$ holds.

(Case 3.) $z_{n-1} = xAy$, $z_n = xAray$ with $r : A' \to a \in P$, and this "A" is a leaf in t_{n-1}. We construct $t_n \in PLD(G)$ from t_{n-1} by relabelling a node "A" with "Ar" and adding the child of "Ar", "a". Here, "a" is a leaf, so that $z_n = preorder(t_n)$ holds.

(Case 4.) $z_{n-1} = xAr'y'$, $z_n = xArar'y'$ with $r : A' \to a \in P$, $r' \in Lab(P)$, and for this "A" and "r'", "Ar'" is an interior node in t_{n-1}. This is not the case to examine because of Observation 2.

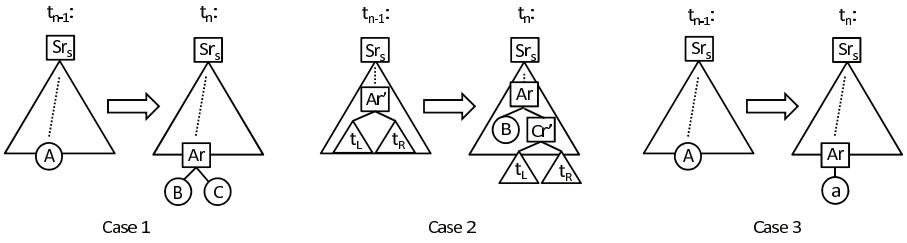

Figure 5.2: A pictorial transformation in Case 1, Case 2 and Case 3

In each case, it holds that $z_n = preorder(t_n)$ for $t_n \in PLD(G)$, which completes the induction.

Because z_n is in F^+, for each nonterminal A which appears in z_n, A must be followed by r with $r : A \to \alpha \in P$. This means that all leaves of t_n are terminals and $t_n \in CLD(G)$. Thus, we obtain $z_n \in preorder(CLD(G))$, where $z_n \in L(\gamma) \cap F^+$. □

Continuation of Proof for Lemma 1.

From Claims 1 and 2, it holds that $preorder(CLD(G)) = L(\gamma) \cap F^+$. Further, the discussion in Section 2.3 reminds us that $L(G) = h(preorder(CLD(G)))$, which proves that $L(G) = h(L(\gamma) \cap F^+)$. □

Next, we prove the inverse inclusion.

Lemma 12. $PR(INS_3^0 \cap STAR(2)) \subseteq \mathcal{CF}$.

Proof. It is known that $\mathcal{CF}$ includes INS_3^0 and $\mathcal{CF}$ is closed under intersection with regular languages and arbitrary morphisms. Therefore, any language in $PR(INS_3^0 \cap STAR(2))$ is context-free. □

From Lemma 11 and Lemma 12, we obtain the main theorem.

Theorem 20. $\mathcal{CF} = PR(INS_3^0 \cap STAR(2))$.

The result above can be reinforced by showing that $STAR(2)$ is optimally small within the families $STAR(k)$ $(k = 1, 2 \dots)$ for the representation of Theorem 3.

Theorem 21. $PR(INS_*^0 \cap STAR(1)) \subset \mathcal{CF}$.

Proof. Note that $INS_*^0 \cap STAR(1) = INS_*^0$. Since $\mathcal{CF}$ includes INS_*^0 and $\mathcal{CF}$ is closed under arbitrary morphisms, we get $PR(INS_*^0 \cap STAR(1)) \subseteq \mathcal{CF}$.

We consider $L = \{a^n b^n \mid n \geq 1\} \in \mathcal{CF}$ and assume that for an insertion system $\gamma = (V, A, P)$ and a projection h, $L = h(L(\gamma))$. From the assumption, $(\lambda, x_1 a x_2, \lambda)$, $(\lambda, y_1 b y_2, \lambda)$ is in P with $x_1, x_2, y_1, y_2 \in V^*$ and neither a nor b is deleted by the projection h. Because $a^k b^k$ is in L for $k \geq 1$, there exists $x = uv$ in $L(\gamma)$ such that $h(x) = a^k b^k$ for some $u, v \in V^*$, where $h(u) = a^k$ and $h(v) = b^k$. Applying $(\lambda, y_1 b y_2, \lambda)$ to x can derive a string $y = y_1 b y_2 uv$ in $L(\gamma)$, and $h(y) = h(y_1 b y_2 uv) =$

$h(y_1)b\,h(y_2)a^kb^k$ must be in L, which is a contradiction. Thus, we have that L is not in $PR(INS^0_*)$. □

Corollary 15. $PR(INS^0_*) \subset \mathcal{CF}$.

In [30], it is shown that context-free languages can be characterized using strictly k-testable languages, that is, $\mathcal{CF} = PR(INS^0_3 \cap LOC(4))$. This result is improved in [11] by $PR(INS^0_2 \cap LOC(3))$. We improve the result in [30] by $PR(INS^0_3 \cap LOC(2))$ and show that $LOC(1)$ is not sufficient to characterize context-free languages.

Theorem 22. $\mathcal{CF} = PR(INS^0_3 \cap LOC(2))$.

Proof. (Proof of $\subseteq$) For a context-free grammar $G = (N, T, S, P)$ in Chomsky normal form, we construct the insertion system $\gamma = (V, \{S\}, P')$ of weight $(3, 0)$ in the same way as the one in the proof of Theorem 3.

Moreover, we construct the strictly 2-testable language R, where

$$\begin{aligned}
A\,(= pref_2(R)) =& \{Sr \mid r : S \to \alpha \in P\},\\
B\,(= suf_2(R)) =& \{ra \mid r : A \to a \in P\},\\
C\,(= pInf_2(R)) =& \{Ar \mid r : A \to \alpha \in P\} \cup \{ra \mid r : A \to a \in P\} \cup\\
& \{rB \mid r : A \to BC \in P\} \cup \{aA \mid A \in N, a \in T\},
\end{aligned}$$

and the projection h, where $h(a) = a$ for $a \in T$, $h(a) = \lambda$ otherwise.

We note that it holds that $R \subset F^+$, where F^+ is defined in the proof of Lemma 1. Thus, it holds that $h(L(\gamma) \cap R) \subseteq h(L(\gamma) \cap F^+) = L(G)$. Similar to the proof of Lemma 1, it can be shown that $h(L(\gamma) \cap R) \supseteq L(G)$. We have $\mathcal{CF} \subseteq PR(INS^0_3 \cap LOC(2))$.

(Proof of $\supseteq$) It is known that $\mathcal{CF}$ includes INS_3^0 and $\mathcal{CF}$ is closed under intersection with regular languages and arbitrary morphisms. Therefore, any language in $PR(INS_3^0 \cap LOC(2))$ is context-free. □

Theorem 23. $PR(INS_*^0 \cap LOC(1)) \subset \mathcal{CF}$.

Proof. Since $\mathcal{CF}$ includes INS_*^0 and $\mathcal{CF}$ is closed under intersection with regular languages and arbitrary morphisms, we get $PR(INS_*^0 \cap LOC(1)) \subseteq \mathcal{CF}$.

We consider $L = \{a^n b^n \mid n \geq 1\} \in \mathcal{CF}$. Assume that for an insertion system $\gamma = (V, A, P)$, a strictly 1-testable language R and a projection h, $L = h(L(\gamma) \cap R)$.

From the assumption, $(\lambda, x_1 a x_2, \lambda)$, $(\lambda, y_1 b y_2, \lambda)$ is in P, where $x_1, x_2, y_1, y_2 \in V^*$ and $pInf_1(R)$ contains the all letters composing $x_1 a x_2$ and $y_1 b y_2$. Here, neither a nor b is not deleted by the projection h. Because $a^k b^k$ is in L for $k \geq 1$, there exists $x = uvw$ in $L(\gamma) \cap R$ such that $h(x) = a^k b^k$ for some $u, v, w \in V^*$, where $h(u) = a^{k-1}$, $h(v) = a$ and $h(w) = b^k$. Applying $(\lambda, y_1 b y_2, \lambda)$ to x can derive a string $y = u y_1 b y_2 v w$ in $L(\gamma) \cap R$, and $h(y) = h(u y_1 b y_2 v w) = a^{k-1} h(y_1) b\, h(y_2) a b^k$ must be in L, which is a contradiction. Thus, L is not in $PR(INS_*^0 \cap LOC(1))$. □

In [27], it is proved that each recursively enumerable language can be represented by an insertion system, an inverse morphism and a projection, that is, $\mathcal{RE} = PR(H^{-1}(INS_4^7))$. This result is improved in [18], [29], where it is proved that $\mathcal{RE} = PR(H^{-1}(INS_3^3))$. We show that $\mathcal{CF}$ can be characterized in a similar way.

Theorem 24. $\mathcal{CF} = PR(H^{-1}(INS_3^0))$.

Proof. (Proof of $\subseteq$) For a context-free grammar $G = (N, T, S, P)$ in Chomsky normal form, we construct the insertion system $\gamma = (V, \{S\}, P')$ of weight $(3, 0)$ in the same way as the one in the proof of Theorem 3.

Then, we consider the new alphabet $U = \{A_r \mid r : A \to \alpha \in P\}$ and construct the morphism $h : (U \cup T)^* \to (N \cup T \cup Lab(P))^*$ which is defined by $h(A_r) = Ar$ for $A_r \in U$, $h(a) = a$ for $a \in T$, and the projection $g : (U \cup T)^* \to T^*$, where $g(a) = a$ for $a \in T$, $g(a) = \lambda$ otherwise. Note that h^{-1} plays a similar role to F^+ in the proof of Lemma 1, because undesired strings (not in F^+) are filtered out by h^{-1}. The rest of the proof is almost similar to the proof of Lemma 1, so that we omit it.

(Proof of $\supseteq$) It is known that $\mathcal{CF}$ includes INS_3^0 and it is closed under inverse morphisms and projections. Thus, any language in $PR(H^{-1}(INS_3^0))$ is context-free.

□

As the class of context-free languages includes INS_3^1, it is easy to derive the following corollary.

Corollary 16. $\mathcal{CF} = PR(H^{-1}(INS_3^1))$.

5.4 A morphic characterization of $\mathcal{RE}$

We can easily characterize $\mathcal{RE}$ by using a star language instead of an inverse morphism. The idea of the proof depends upon the similar results in [18, 27, 30].

Theorem 25. $\mathcal{RE} = PR(INS_3^3 \cap STAR(2))$.

Proof. For a phrase structure grammar $G = (N, T, P, S)$ in Penttonen normal form, construct the insertion system $\gamma = (V, \{Sccc\}, P')$ of weight $(3, 3)$, where $V = N \cup T \cup \{\#, c\}$, and $\#$, c are not in $N \cup T$. The set of rules P' is constructed as follows:

1. For each rule $A \to a$ in P ($a \in T \cup \{\lambda\}$), there are rules $(A, \#a, \alpha_1\alpha_2\alpha_3)$, where $\alpha_1 \in V - \{\#\}$, $\alpha_2, \alpha_3 \in V$ and $\alpha_2\alpha_3 \neq \#\#$.

2. For each rule $A \to BC$ in P, there are rules $(A, \#BC, \alpha_1\alpha_2\alpha_3)$, where $\alpha_1 \in V - \{\#\}$, $\alpha_2, \alpha_3 \in V$ and $\alpha_2\alpha_3 \neq \#\#$.

3. For each rule $AB \to AC$ in P, there are rules $(AB, \#C, \alpha_1\alpha_2\alpha_3)$, where $\alpha_1 \in V - \{\#\}$, $\alpha_2, \alpha_3 \in V$ and $\alpha_2\alpha_3 \neq \#\#$.

4. For each $X, Y \in N$, there are rules $(XY\#, \#X, \alpha)$, where $\alpha \in N \cup T \cup \{c\}$.

5. For each $X, Y \in N$, there are rules $(X, \#, Y\#\#)$.

6. For each $X, Y \in N$, there are rules $(\#Y\#, Y, \#X)$.

We construct the 2-star language F^+, where $F = \{A\# \mid A \in N\} \cup T \cup \{c\}$, and the projection h is defined as $h(a) = a$ for $a \in T$, $h(a) = \lambda$ otherwise.

Without loss of generality, we may assume that in every derivation in G, the rules of the form $A \to \alpha$(corresponding to (1)) are applied only in the final steps.

The symbol # is said to be a *marker*. A nonterminal in N followed by # is said to be *marked*.

By using the rules (1), (2), (3), we can simulate derivations of G. In a derivation of γ, a consumed nonterminal is marked by #, instead of being rewritten.

In the case where the rule (3) is used, pairs of unmarked nonterminals can be separated by one or more marked nonterminals. By using the rules (4), (5), (6) in this order, we can move an unmarked nonterminal across a marked nonterminal as follows:

$$\underline{X}Y\#\underline{Z} \overset{(4)}{\Rightarrow} XY\#\#XZ \overset{(5)}{\Rightarrow} X\#Y\#\#XZ \overset{(6)}{\Rightarrow} X\#Y\#Y\#\underline{XZ}.$$

Iterating the above derivation enables an unmarked nonterminal (X) to move across more than one marked nonterminal.

(Proof sketch of $L(G) \subseteq h(L(\gamma) \cap F^+)$)

We can easily verify if the rules of γ are used in a manner described above, then γ correctly simulates all the derivations of G. During such a correct simulation, the auxiliary substrings of the form $A\#$ are inserted into the sentential form. In the string in $L(\gamma) \cap F^+$, all the nonterminals are followed by #. This means that all the nonterminals have been consumed. Finally, h removes all the symbols but terminals in T. Hence, $L(G) \subseteq h(L(\gamma) \cap F^+)$.

(Proof sketch of $L(G) \supseteq h(L(\gamma) \cap F^+)$)

We need to show that γ can produce only the sentential forms which correspond to derivations in G. For a sentential form w of γ, consider the rules (4), (5), (6).

- For a substring of the form $XY\#Z$ of w, where $X, Y, Z \in N$, the rule (4) can be applied only once to it, producing $XY\#\#XZ$. Observe that the substring ## cannot be produced by an application of any rule other than (4). Note that (5) and (6) are applicable only if the substring ## appears in w.

- Following an application of rule (4), only (5) can be applied to the substring $XY\#\#$, producing $X\#Y\#\#$.

- Similarly, following an application of rule (5), only (6) can be applied to $X\#Y\#\#$, producing $X\#Y\#Y\#$.

Hence, after an application of rule (4), rules (5) and (6) must be applied in this order to an adequate position of the sentential form for each. Unmarked nonterminals and their order in w are preserved by the rules (4), (5), (6).

Consequently, unmarked nonterminals in a sentential form of γ can only be changed (and consumed) by the rules (1), (2), (3), and their applications can be clearly simulated by G.

Moreover, taking the intersection of $L(\gamma)$ with F^+ filters only the sentential forms whose nonterminals are all marked. Therefore, $L(G) \supseteq h(L(\gamma) \cap F^+)$. □

5.5 Discussion

It is clear that star languages are conceptually simpler than strictly locally testable languages, because a star language F^+ is a finitely generated monoid obtained from a generator set F by concatenating arbitrary number of elements of F in an arbitrary order. In fact, one can show that a star language with some property (called k-parsability, see [28]) is strictly locally testable. We note that the 2-star languages used in the proofs in this chapter are 1-parsable and, therefore, strictly locally testable.

We have shown that $\mathcal{CF} = PR(INS_3^0 \cap STAR(2))$; namely, a language L is in $\mathcal{CF}$ iff $L = h(L(\gamma) \cap F^+)$, where γ is a context-free insertion systems of weight (3,0), F^+ is a 2-star language, and h is a projection. A morphic characterization of $\mathcal{RE}$ was also presented in the form $\mathcal{RE} = PR(INS_3^3 \cap STAR(2))$. (We remark that a similar representation for $\mathcal{REG}$ could be obtained by particularizing the proof construction used for the former result.)

In comparison to the well-known Chomsky-Schützenberger characterization

of context-free languages, our result benefits from a great simplicity by reducing $\mathcal{REG}$ to $STAR(2)$, at the price of enhancing *Dyck* up to INS_3^0. It may also be interestingly compared to the result $\mathcal{CF} = PR(INS_2^0 \cap LOC(3))$ in [11], where another trade-off relation is found in the parameters on INS and $LOC(STAR)$, while $STAR$ is conceptually much simpler than LOC.

Lastly, the following questions remain open :

- $\mathcal{CF} = PR(INS_2^0 \cap STAR(k))$, for some $k \geq 1$?
- How large is the class $PR(H^{-1}(INS_3^2))$?, while we only know that it must be between $\mathcal{CF}$ and $\mathcal{RE}$.
- $\mathcal{RE} = PR(INS_i^2 \cap STAR(k))$, for some $i, k \geq 1$?
- $\mathcal{RE} = PR(INS_2^j \cap STAR(k))$, for some $j, k \geq 1$?

Bibliography

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

[2] J. Castellanos, V. Mitrana, Some remarks on hairpin and loop languages, in *Words, Semigroups, and Translations*, World Scientific, Singapore, pp.47-59, 2001.

[3] D. Cheptea, C. Martin-Vide, V. Mitrana, A new operation on words suggested by DNA biochemistry: hairpin completion, in *Proc. Transgressive Computing*, pp.216-228, 2006.

[4] E. Csuhaj-Varju, O.H.Ibarra, Gy. Vaszil, On the computational complexity of P automata, *Natural Computing* vol.5, pp.109-126, 2006.

[5] E. Csuhaj-Varju, Gy. Vaszil, P automata or purely communicating accepting P systems, LNCS 2597, Springer, pp.219-233, 2003.

[6] A. Ehrenfeucht, G. Rozenberg, Reaction systems, *Fundamenta Informaticae* vol.75, pp.263-280, 2007.

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

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

[9] A. Ehrenfeucht, M. Main, G. Rozenberg, Combinatorics of life and death in reaction systems, *Intern. J. of Foundations of Computer Science* vol.21, pp.345-356, 2010.

[10] A. Ehrenfeucht, M. Main, G. Rozenberg, Functions defined by reaction systems, *Intern. J. of Foundations of Computer Science* vol.22, pp.167-178, 2011.

[11] K. Fujioka, Refinement of representation theorems for context-free languages, *IEICE Transactions*, E93-D(2):227–232, 2010.

[12] S. Ginsburg, *Algebraic and automata-theoretic properties of formal languages*, North-Holland, Amsterdam, 1975.

[13] M. Hagiya, M. Arita, D. Kiga, K. Sakamoto, S. Yokoyama, Towards parallel evaluation and learning of Boolean μ-formulas with molecules, *DNA Based Computers III* (Rubin, H. and Wood, D. eds.), *DIMACS Series in Discrete Mathematics*, vol. 48, pp. 57-72, 2000.

[14] J.E. Hopcroft, T. Motwani, J.D. Ullman, *Introduction to automata theory, language and computation* - 2nd ed, Addison-Wesley, 2003.

[15] M. Ito, P. Leupold, F. Manea, V. Mitrana, Bounded hairpin completion, *Information and Computation*, vol.209, pp.471-485, 2011.

[16] L. Kari, S. Konstantinidis, P. Sosik, G. Thierrin, On hairpin-free words and languages, in *Proc. Developments in Language Theory 2005*, LNCS 3572, Springer, pp.296-307, 2005.

[17] L. Kari, Gh. Păun, G. Thierrin, S. Yu, At the crossroads of dna computing and formal languages: Characterizing re using insertion-deletion systems, In *Proc. 3rd DIMACS Workshop on DNA Based Computing*, pages 318–333, 1997.

[18] L. Kari and P. Sosík, On the weight of universal insertion grammars, *Theor. Comput. Sci.*, 396(1-3):264–270, 2008.

[19] K. Komiya, A. Rose, Experimental validation of signal dependent operation in Whiplash PCR, *DNA Computing. 14th International Workshop on DNA-Based Computers* (Goel, A. and Simmel, F.C., eds.), LNCS 5347, Springer, pp.1-10, 2009.

[20] K. Komiya, K. Sakamoto, A. Kameda, M. Yamamoto, A. Ohuchi, D. Kiga, S. Yokoyama, M. Hagiya, DNA polymerase programmed with a hairpin DNA incorporates a multiple-instruction architecture into molecular computing, *Biosystems*, vol. 83, pp. 18-25, 2006.

[21] S. Kopecki, On the iterated hairpin completion. In Y. Gao, H. Lu, S. Seki, and S. Yu (editors), *14th Developments in Language Theory*, LNCS 6224, Springer, pp.438-439, 2010. Also, in http://arxiv.org/abs/1010.3640.

[22] F. Manea, C. Martín-Vide, V. Mitrana, On some algorithmic problems regarding the hairpin completion, *Discr. Appl. Math.*, vol.157, pp.2143-2152, 2009.

[23] F. Manea, C. Martín-Vide, V. Mitrana, Hairpin Lengthening and Shortening of Regular Languages, In H. Bordihn, M. Kutrib and B. Truthe, editors, *Languages Alive*, volume 7300 of *Lecture Notes in Computer Science*, pages 145-159. Springer, 2012.

[24] F. Manea, V. Mitrana, Hairpin completion versus hairpin reduction, in *Computation in Europe CiE 2007*, LNCS 4497, Springer, pp.532-541, 2007.

[25] F. Manea, V. Mitrana, T. Yokomori, Two complementary operations inspired by the DNA hairpin formation: completion and reduction, *Theor. Comput. Sci.*, vol.410, pp.41-425, 2009.

[26] M. Margenstern, Gh. Păun, Y. Rogozhin, S. Verlan, Context-free insertion-deletion systems, *Theor. Comput. Sci.*, 330:339–348, 2005.

[27] C. Martín-Vide, Gh. Păun, A. Salomaa, Characterizations of recursively enumerable languages by means of insertion grammars, *Theor. Comput. Sci.*, 205(1-2):195–205, 1998.

[28] R. McNaughton, S. Papert, *Counter-free automata*, M.I.T. Press Cambridge, Mass., 1971.

[29] K. Onodera, A note on homomorphic representation of recursively enumerable languages with insertion grammars, *IPSJ Journal*, 44(5):1424–1427, 2003.

[30] K. Onodera, New morphic characterizations of languages in chomsky hierarchy using insertion and locality, In A.H. Dediu, A.-M. Ionescu, and

C. Martín-Vide, editors, *LATA*, volume 5457 of *Lecture Notes in Computer Science*, pages 648–659. Springer, 2009.

[31] Gh. Păun, *Marcus Contextual Grammars*, Kluwer, Dordrecht, Boston, 1998.

[32] Gh. Păun, Computing with membranes, *Journal of Computer and System Sciences* vol.61, pp.108-143, 2000.

[33] Gh. Păun, M.J. Pérez-Jiménez, T. Yokomori, Representations and characterizations of languages in chomsky hierarchy by means of insertion-deletion systems, *Int. J. Found. Comput. Sci.*, 19(4):859–871, 2008.

[34] Gh. Păun, G. Rozenberg, A. Salomaa, *DNA Computing: New Computing Paradigms*, Springer-Verlag Berlin, Inc., 1998.

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

[36] G. Păun, G. Rozenberg, T. Yokomori, Hairpin languages, *Intern. J. Found. Comp. Sci.*, vol. 12, pp.837-847, 2001.

[37] G. Rozenberg, A. Salomaa (Eds.), *Handbook of Formal Languages*, 3 volumes, Springer-Verlag, Berlin, Heidelberg (1997).

[38] K. Sakamoto, D. Kiga, K. Komiya, H. Gouzu, S. Yokoyama, S. Ikeda, H. Sugiyama, M. Hagiya, State transitions by molecules, *BioSystems*, vol.52, no.1-3, pp.81-91, 1999.

[39] K. Sakamoto, H. Gouzu, K. Komiya, D. Kiga, S. Yokoyama, T. Yokomori, M. Hagiya, Molecular computation by DNA hairpin formation, *Science*, vol. 288, pp.1223-1226, 2000.

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

[41] Y. Suzuki, Y. Fujiwara, J. Takabayashi, H. Tanaka, Artificial Life Applications 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.

[42] S. Verlan, On minimal context-free insertion-deletion systems, *Autom. Lang. Combin.*, 12(1):317–328, 2007.

List of papers by Fumiya Okubo

Refereed papers:

1. F. Okubo, A note on the descriptional complexity of semi-conditional grammars, *Information Processing Letters*, 110(1), pp.36-40, 2009.

2. F. Okubo, T. Yokomori, Morphic Characterizations of Language Families in terms of Insertion Systems and Star Languages, *International Journal of Foundations of Computer Science*, 22(1), pp.247-260, 2011.
   (Contribution to Chapter 5)

3. F. Okubo, T. Yokomori, On the Hairpin Incompletion, *Fundamenta Informaticae*, 110, pp.255-269, 2011.
   (Contribution to Chapter 4)

4. F. Okubo, S. Kobayashi, T. Yokomori, Reaction Automata, *Theoretical Computer Science*, 429, pp.247-257, 2012.
   (Contribution to Chapter 3)

5. F. Okubo, S. Kobayashi, T. Yokomori, On the Properties of Language Classes Defined by Bounded Reaction Automata, *Theoretical Computer Science*, 454, pp.206-221, 2012.
   (Contribution to Chapter 3)

6. F. Okubo, On the Computational Power of Reaction Automata Working in Sequential Manner, *Proceedings of 4th Workshop on Non-Classical Models for Automata and Applications*, book@ocg.at series, Öesterreichische Computer Gesellschaft, 290, pp.149-164, 2012.
   (Contribution to Chapter 3)

7. F. Okubo, On language classes defined by reaction automata, *Academic Studies and Scientific Research*, Faculty of Education and Integrated Arts and Sciences, Waseda University, 61, pp.39-46, 2013.
   (Contribution to Chapter 3)

8. F. Okubo, Reaction Automata Working in Sequential Manner, submitted.
   (Contribution to Chapter 3)

Unrefereed papers:

1. F. Okubo, S. Kobayashi, T. Yokomori, Automata inspired by biochemical reaction, *RIMS Kōkyūroku*, Kyoto University, 1779, pp.179-182, 2012.
   (Contribution to Chapter 3)