Morphic characterizations of languages in Chomsky hierarchy with insertion and locality

Fujioka, Kaoru
Office for Strategic Research Planning, Kyushu University

https://hdl.handle.net/2324/25741

出版情報：Information and Computation. 209 (3), pp.397-408, 2011-03. Elsevier
バージョン：

Morphic Characterizations of Languages in Chomsky Hierarchy with Insertion and Locality

Kaoru Fujioka

Office for Strategic Research Planning, Kyushu University,
6-10-1 Hakozaki Higashi-ku Fukuoka 812-8581, JAPAN

Abstract

This paper concerns new characterizations of regular, context-free, and recursively enumerable languages, using insertion systems with lower complexity. This is achieved by using both strictly locally testable languages and morphisms. The representation is in a similar way to the Chomsky-Schützenberger representation of context-free languages. Specifically, each recursively enumerable language L can be represented in the form $L = h(L(\gamma) \cap R)$, where γ is an insertion system of weight $(3, 3)$, R is a strictly 2-testable language, and h is a projection. A similar representation can be obtained for context-free languages, using insertion systems of weight $(2, 0)$ and strictly 2-testable languages, as well as for regular languages, using insertion systems of weight $(1, 0)$ and strictly 2-testable languages.

Keywords: insertion system, strictly locally testable language,
Chomsky-Schützenberger theorem

1. Introduction

DNA computing theory involves the use of *insertion* and *deletion operations*. It has been shown that by using insertion and deletion operations, any recursively enumerable language can be obtained in [2], [3].

In insertion systems, we can use only insertion operations which is based on insertion rules of the form (u, x, v), where u, x, v are strings over an alphabet, a new string $\alpha uxv\beta$ is produced for a given string $\alpha uv\beta$ with

[☆] A preliminary version of this article appeared in [1].

Email address: kaoru@tcslab.csce.kyushu-u.ac.jp (Kaoru Fujioka) context uv using (u, x, v). From the definition of insertion operations, one would easily imagine that by using only insertion operations, we generate only context-sensitive languages.

On the other hand, the class of strictly locally testable languages is known as a proper subclass of regular language classes [4]. The equivalence relation between a certain type of splicing languages (generated by persistent splicing systems) and strictly locally testable languages is known in [5].

In this paper, we focus on characterizing the classes of languages in Chomsky hierarchy by using insertion systems together with some "additional mechanisms" in the Chomsky-Schützenberger-like form. It has been shown that using insertion systems together with some morphisms, characterizing recursively enumerable languages is accomplished in [3], [6], [7]. For context-free languages, there is a well-known Chomsky-Schützenberger characterization: each context-free language L can be represented in the form $L = h(D \cap R)$, where D is a Dyck language, R is a regular language, and h is a projection. It has been shown that each recursively enumerable language L can be represented in a similar way to the well-known Chomsky-Schützenberger representation of context-free languages, $L = h(L(\gamma) \cap D)$, where γ is an insertion system, h is a projection, and D is a Dyck language [8]. In this paper, we use strictly locally testable languages and morphisms as the additional mechanisms for characterizing languages in Chomsky hierarchy.

In insertion systems, a pair of the maximum length of inserted strings and the one of context-checking strings, called *weight* is an important parameter for generative powers. As for strictly locally testable languages, the length of local testability-checking is considered. The optimality of these two parameters is to be checked.

We prove that each recursively enumerable language can be represented in the form $h(L(\gamma) \cap R)$, where γ is an insertion system of weight $(3, 3)$, h is a morphism, and R is a strictly 2-testable language. Similar characterizations are shown for context-free and regular languages.

2. Preliminaries

In this section, we introduce necessary notation and basic definitions needed in this paper. We assume the reader to be familiar with the rudiments on basic notions in formal language theory (see, e.g., [3], [9]).

2.1. Basic Definitions

For an alphabet V, V^* is the set of all strings of symbols from V which includes the empty string λ. For a string $x \in V^*$, $|x|$ denotes the length of x. For $0 \leq k \leq |x|$, let $Pre_k(x)$ and $Suf_k(x)$ be the prefix and the suffix of x with length k, respectively. For $0 \leq k \leq |x|$, let $Int_k(x)$ be the set of *proper* interior substrings of x with length k, while if $|x| = k$, then $Int_k(x) = \emptyset$.

2.2. Normal Forms of Grammars

A *phrase structure grammar* is a quadruple $G = (N, T, P, S)$, where N is a set of *nonterminal symbols*, T is a set of *terminal symbols*, P is a set of *production rules*, and S in N is the *initial symbol.* A rule in P is of the form $r : \alpha \rightarrow \beta$, where $\alpha \in (N \cup T)^* N (N \cup T)^*$, $\beta \in (N \cup T)^*$, and r is a label from a given set $Lab(P)$ such that there are no production rules with the same label. For any x and y in $(N \cup T)^*$, if $x = u\alpha v$, $y = u\beta v$, and $r : \alpha \rightarrow \beta \in P$, then we write

$$x \overset{r}{\Longrightarrow}_G y.$$

We say that x *directly derives* y with respect to G. If there is no confusion, we write $x \Longrightarrow y$. The n-th power of $\Longrightarrow$, denoted as $\Longrightarrow^n$, is defined by $x \Longrightarrow x$ with $k = 0$ for any x in $(N \cup T)^*$. For any $n > 0$ and $x, z \in (N \cup T)^*$, $x \Longrightarrow^n z$ holds if there is $y \in (N \cup T)^*$ such that $x \Longrightarrow^{n-1} y$ and $y \Longrightarrow z$. The reflexive and transitive closure of $\Longrightarrow$ is denoted by $\Longrightarrow^*$.

We define a *language* $L(G)$ generated by a grammar G as follows:

$$L(G) = \{w \in T^* \mid S \Longrightarrow_G^* w\}.$$

It is well known that the class of languages generated by the phrase structure grammars is equal to the class of *recursively enumerable languages RE* [9].

A grammar $G = (N, T, P, S)$ is *context-free* if P is a finite set of *context-free rules* of the form $A \rightarrow \alpha$, where $A \in N$ and $\alpha \in (N \cup T)^*$. A language L is a *context-free language* if there is a context-free grammar G such that $L = L(G)$. Let CF be the class of context-free languages.

A context-free grammar $G = (N, T, P, S)$ is in *Chomsky normal form* if each production rule in P is of one of the following forms:

1. $X \rightarrow YZ$, where $X, Y, Z \in N$.

2. $X \to a$, where $X \in N$, $a \in T$.
3. $S \to \lambda$ (only if S does not appear in right-hand sides of production rules).

It is well known that, for each context-free language L, there is a context-free grammar in Chomsky normal form generating L [9].

A grammar $G = (N, T, P, S)$ is *regular* if P is a finite set of production rules of the form $X \to \alpha$, where $X \in N$ and $\alpha \in TN \cup T \cup \{\lambda\}$. A language L is a *regular language* if there is a regular grammar G such that $L = L(G)$. Let REG be the class of regular languages.

We are going to define a strictly locally testable language, which is one of the main objectives of the present work.

Let k be a positive integer. A language L over T is *strictly k-testable* if there is a triplet $S_k = (A, B, C)$ with sets of strings over T of length k $A, B, C \subseteq T^k$ such that for any w with $|w| \geq k$, w is in L iff $Pre_k(w) \in A$, $Suf_k(w) \in B$, $Int_k(w) \subseteq C$.

Note that if L is strictly k-testable, then L is strictly k'-testable for all $k' > k$. Further, the definition of strictly k-testable says nothing about the strings of "length $k-1$ or less".

A language L is *strictly locally testable* iff there exists an integer $k \geq 1$ such that L is strictly k-testable. Let $LOC(k)$ be the class of strictly k-testable languages. Then one can prove the following theorem.

Theorem 1. [10] *$LOC(1) \subset LOC(2) \subset \cdots \subset LOC(k) \subset \cdots \subset REG$.*

We are now going to define an insertion system. An *insertion system* is a triple $\gamma = (T, P, A_X)$, where T is an alphabet, P is a finite set of *insertion rules* of the form (u, x, v) with $u, x, v \in T^*$, and A_X is a finite set of strings over T called axioms.

We write $\alpha \overset{r}{\Longrightarrow}_\gamma \beta$ if $\alpha = \alpha_1 uv\alpha_2$ and $\beta = \alpha_1 uxv\alpha_2$ for some insertion rule $r : (u, x, v) \in P$ with $\alpha_1, \alpha_2 \in T^*$. If there is no confusion, we write $\alpha \Longrightarrow \beta$. As usual, $\Longrightarrow^n$ denotes the n-th power of $\Longrightarrow$. The reflexive and transitive closure of $\Longrightarrow$ is denoted by $\Longrightarrow^*$.

A language generated by γ is defined by

$$L(\gamma) = \{w \in T^* \mid s \Longrightarrow_\gamma^* w, \text{ for some } s \in A_X\}.$$

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

$$\begin{array}{rcl} m & = & max\{\ |x| \mid (u, x, v) \in P\}, \\ n & = & max\{\ |u| \mid (u, x, v) \in P \text{ or } (v, x, u) \in P\}. \end{array}$$

For $m, n \geq 0$, INS_m^n denotes the class of all languages generated by insertion systems of weight (m', n') with $m' \leq m$ and $n' \leq n$. When the parameter is not bounded, we replace m or n with $*$.

For insertion systems, there exist the following results.

Theorem 2 ([3])**.** 1. *For the class of finite languages FIN and the one of context-sensitive languages CS, $FIN \subset INS_*^0 \subset INS_*^1 \cdots \subset INS_*^* \subset CS$.*

2. *$REG \subset INS_*^*$.*
3. *$INS_*^1 \subset CF$.*
4. *CF is incomparable with all INS_*^n $(n \geq 2)$, and INS_*^*.*
5. *INS_2^2 contains non-semilinear languages.*

From the definition of insertion systems, we can easily prove the following lemma.

Lemma 1. *$INS_1^0 \subset REG$.*

We are now going to introduce some notations concerning morphisms, which help to express the class of languages represented in the Chomsky-Schützenberger-like form. A mapping $h : V^* \to T^*$ is called *morphism* if $h(\lambda) = \lambda$ and $h(xy) = h(x)h(y)$ for any $x, y \in V^*$. For languages L_1, L_2, and a morphism h, we introduce the following notation: $h(L_1 \cap L_2) = \{h(w) \mid w \in L_1 \cap L_2\}$. For language classes $\mathcal{L}_1$ and $\mathcal{L}_2$, we introduce the following class of languages:

$$H(\mathcal{L}_1 \cap \mathcal{L}_2) = \{h(L_1 \cap L_2) \mid h \text{ is a morphism}, L_i \in \mathcal{L}_i\ (i = 1, 2)\}.$$

3. Characterizations of Regular Languages

In this section, we will characterize regular languages in terms of insertion languages and strictly locally testable languages both of which form proper subclasses of regular languages.

Lemma 2. *$REG \subseteq H(INS_1^0 \cap LOC(2))$.*

Proof. For a regular language L, let $G = (N, T, P, S)$ be a regular grammar such that $L = L(G)$. Using the new symbol F, we construct the insertion system $\gamma = (V, P', \{\lambda\})$ of weight $(1, 0)$, where

$$\begin{aligned} V = & \ \{X_r \mid r : X \to \alpha \in P,\ \alpha \in TN \cup T \cup \{\lambda\}\} \cup \{F\}, \\ P' = & \ \{(\lambda, X, \lambda) \mid X \in V\}. \end{aligned}$$

Then, $L(\gamma) = V^*$.

Further, we define the morphism $h : V^* \to T^*$ by

$$\begin{array}{lcll} h(X_r) & = & a & \text{if } \ r : X \to aY \in P \text{ or } r : X \to a \in P, \\ h(X_r) & = & \lambda & \text{if } \ r : X \to \lambda \in P, \\ h(F) & = & \lambda. & \end{array}$$

Finally, consider $R = AV^* \cap V^*B - V^+C'V^+$ with $C' = V^2 - C$, where

$$\begin{array}{lcl} A & = & \{S_r X_{r_1} \mid r : S \to aX \in P,\ r_1 : X \to \alpha \in P,\ \alpha \in T \cup TN \cup \{\lambda\}\} \cup \\ & & \{S_r F \mid r : S \to \alpha \in P,\ \alpha \in T \cup \{\lambda\}\}, \\ B & = & \{X_r F \mid r : X \to a \in P \ \text{ or } \ r : X \to \lambda \in P\}, \\ C & = & \{X_r Y_{r_1} \mid r : X \to aY \in P,\ r_1 : Y \to \alpha \in P,\ \alpha \in T \cup TN \cup \{\lambda\}\}. \end{array}$$

Then R is a strictly 2-testable language prescribed by $S_2 = (A, B, C)$.

We will show that, for any $X \in N$, $X \xrightarrow{r_1}_G \cdots \xrightarrow{r_{n-1}}_G w'Y \xrightarrow{r_n}_G w'y = w \in T^*$ iff $X_{r_1} \cdots Y_{r_n} F \in V^*B - V^*C'V^+$ with $h(X_{r_1} \cdots Y_{r_n} F) = w$ by the induction on n.

Base step: For a nonterminal symbol X in N, there is a derivation $X \xrightarrow{r}_G w$ with $w \in \{\lambda\} \cup T$ iff from the definitions of P' and R, $X_r F$ is hence in $V^*B - V^*C'V^+$. Furthermore, from the definition of h, $h(X_r F) = w$.

Induction step: Suppose that the claim holds for any $n \leq k$. Consider a derivation $X \xrightarrow{r}_G aY \overset{r_1 \cdots r_{k-1}}{\Longrightarrow}_G aw'Z \xrightarrow{r_k}_G aw$, where $a \in T$, $w, w' \in T^*$, $Y, Z \in N$.

For the rules r and r_1, by the constructions of V and R, r and r_1 are in P iff $X_r Y_{r_1}$ is in $V^* \cap C$. From the definition of h, $h(X_r) = a$. By the induction hypothesis, $Y \Longrightarrow_G^* w$ iff a string $Y_{r_1} \cdots Z_{r_k} F$ is in $V^*B - V^*C'V^+$ with $h(Y_{r_1} \cdots Z_{r_k} F) = w$. Therefore, $X \Longrightarrow^* aw$ iff $X_r Y_{r_1} \cdots Z_{r_k} F \in V^*B - V^*C'V^+$ with $h(X_r Y_{r_1} \cdots Z_{r_k} F) = aw$.

Note that, for the special case where $X = S$, $S_r Y_{r_1}$ is in A, which implies that $S_r Y_{r_1} \cdots Z_{r_k} F \in AV^* \cap V^*B - V^*C'V^+$. Then, $S_r Y_{r_1} \cdots Z_{r_k} F$ is in $AV^* \cap V^*B - V^+C'V^+$ with $h(S_r Y_{r_1} \cdots Z_{r_k} F) = aw$. Therefore, for any w in L, w is in $L(G)$ iff w is in $h(L(\gamma) \cap R)$. □

Lemma 3. $H(INS_1^0 \cap LOC(2)) \subseteq REG$.

Proof. Since the class of regular languages is closed under intersection with regular languages and morphisms, the result follows from the facts that $INS_1^0 \subset REG$ in Lemma 1 and $LOC(2) \subset REG$ in Theorem 1. □

From Lemma 2 and Lemma 3, we have the following theorem.

Theorem 3. $REG = H(INS_1^0 \cap LOC(2))$.

Since for arbitrary k with $k \geq 2$, the class of regular languages includes the class of strictly k-testable languages, the next result follows from Theorem 3 and Theorem 1.

Corollary 1. For all $k \geq 2$, $REG = H(INS_1^0 \cap LOC(k))$.

The value of parameter $k = 2$ in the strictly k-testable languages in Theorem 3 is necessary for expressing regular languages in the following sense.

Lemma 4. *There exists a regular language which cannot be written in the form* $h(L(\gamma) \cap R)$*, for any insertion system* γ *of weight* $(i, 0)$ $(\forall i \geq 1)$*, strictly* 1*-testable language* R*, and morphism* h*.*

Proof. Consider the regular language $L = \{a^l \mid l \geq 0\} \cup \{b^l \mid l \geq 0\}$. Suppose that there is an insertion system $\gamma = (V, P, A_X)$ of weight $(i, 0)$ with $i \geq 1$, a strictly 1-testable language R prescribed by $S_1 = (A, B, C)$, and a morphism h such that $L = h(L(\gamma) \cap R)$.

Then, for any $l \geq 0$, there exists the set of strings $D_l = \{x \mid h(x) = a^l\} \cup \{y \mid h(y) = b^l\}$ such that $D_l \subset L(\gamma) \cap R$. Let $D = \bigcup_{l \geq 0} D_l$, then D is an infinite set. Since $D \subset L(\gamma) \cap R$ holds, $L(\gamma) \cap R$ is also an infinite set. Then P includes both (λ, u_a, λ) and (λ, u_b, λ), where $u_a, u_b \in C^i$, $h(u_a) = a^{i_a}$, $h(u_b) = b^{i_b}$ for some $i_a, i_b > 0$. Let t_1xt_2 and t_3yt_4 be in $L(\gamma) \cap R$ with $t_1, t_3 \in A$, $t_2, t_4 \in B$, $u_a \in Int_i(t_1xt_2)$, $u_b \in Int_i(t_3yt_4)$.

Then, the string $t_1u_bxt_2$ is in $L(\gamma) \cap R$ satisfying $|h(t_1u_bxt_2)|_a \geq i_a > 0$ and $|h(t_1u_bxt_2)|_b \geq i_b > 0$, which contradicts to the fact that $L = \{a^l \mid l \geq 0\} \cup \{b^l \mid l \geq 0\}$. □

From Lemma 4, Theorem 1, and Theorem 3, we have the following theorem.

Theorem 4. $H(INS_1^0 \cap LOC(1)) \subset REG$.

The value of weight $(1, 0)$ in insertion systems in Theorem 3 is optimal for expressing regular languages in the following sense.

Lemma 5. *There exist an insertion system γ of weight $(2, 0)$, a strictly 1-testable language R, and a morphism h such that $h(L(\gamma) \cap R)$ is non-regular.*

Proof. Consider an insertion system $\gamma = (T, \{\lambda\}, \{(\lambda, ab, \lambda)\})$ with $T = \{a, b\}$. Then, for any w in $L(\gamma)$, $|w|_a = |w|_b$ holds.

Consider $R = AT^* \cap T^*B - T^+C'T^+$ with $C' = T - C$, where $A = B = C = T$. Then $R = T^+$ is a strictly 1-testable language prescribed by $S_1 = (T, T, T)$. Further, we define a morphism $h : T^* \to T^*$ by $h(c) = c$ for any $c \in T$. Then, we have $L(\gamma) \cap R = h(L(\gamma) \cap R) = \{w \mid w \in L(\gamma),\ w \neq \lambda\}$.

For a regular language $R_* = \{a^i b^j \mid i, j \geq 1\}$, $h(L(\gamma) \cap R) \cap R_* = \{a^i b^i \mid i \geq 1\}$ is not regular. From the fact that the class of regular languages is closed under intersection with regular languages, $h(L(\gamma) \cap R)$ is not regular. □

From Lemma 4, Lemma 5, and the fact that $INS_i^0 \subseteq INS_{i+1}^0$ with $i \geq 1$, we have the following corollary.

Corollary 2. REG and $H(INS_i^0 \cap LOC(1))$ are incomparable ($i \geq 2$).

From Lemma 5, Theorem 2, and Theorem 1, we have the following corollary.

Corollary 3. $REG \subset H(INS_i^0 \cap LOC(k))$ ($i \geq 2$, $k \geq 2$).

4. Characterizations of Context-Free Languages

We will show how context-free languages can be characterized by insertion systems and strictly locally testable languages. In [8], each context-free language L can be written in the form $L = h(L(\gamma) \cap R)$, where γ is an insertion system of weight $(3, 0)$, R is a *star language*, i.e., $R = F^*$ for a finite set of strings F, and h is a projection. Let us start by showing the relationships between the class of star languages *Star* and the one of strictly k-testable languages for $k \geq 1$.

Lemma 6. *For all $k \geq 1$, $LOC(k)$ and Star are incomparable.*

Proof. For a given k with $k \geq 1$, consider a star language $R = \{a^{(k+1)i} \mid i \geq 1\}$. Suppose that there is a triplet (A, B, C) with $A, B, C \subseteq T^k$ prescribing R. Since $a^{k+1} \in R$ holds, we have $a^k \in A$ and $a^k \in B$. Then a string $a^k \in A \cap B$ is in the strictly k-testable language prescribed by (A, B, C), which contradicts to the fact that $a^k \notin R = \{a^{(k+1)i} \mid i \geq 1\}$.

Conversely, consider a strictly 1-testable language $L_1 = \{a^n b \mid n > 0\}$ prescribed by $S_1 = (\{a\}, \{b\}, \{a\})$. Then L_1 is prefix-free, i.e., no string in L_1 is a prefix of another string in L_1. Suppose that there is a finite set F such that $L_1 = F^*$, which contradicts to the fact that L_1 is prefix-free. From Theorem 1, for any $k \geq 1$, L_1 is in $LOC(k)$. Therefore, for any $k \geq 1$, $LOC(k)$ and $Star$ are incomparable. □

Let us consider the following theorem which is essential for this section.

Theorem 5. $CF \subseteq H(INS_2^0 \cap LOC(2))$.

Construction of an insertion system γ: Consider a context-free grammar $G = (N, T, P, S)$ in Chomsky normal form. We construct an insertion system $\gamma = (\Sigma, P_\gamma, \{S\})$, where

$$\begin{aligned} \Sigma = \; & N_\gamma \cup N \cup T \cup \{S_r \mid r : S \to \lambda \in P\} \quad \text{with} \\ N_\gamma = \; & \{X_{r_1}, X_{r_2} \mid r : X \to YZ \in P,\ X, Y, Z \in N\} \cup \\ & \{X_{r_3} \mid r : X \to a \in P,\ X \in N,\ a \in T\}, \end{aligned}$$

and P_γ contains the following insertion rules:

- for each production rule $r : X \to YZ \in P$ with $X, Y, Z \in N$, we construct the following *r-pair* insertion rules

$$\begin{array}{ll} \text{form-(1)} & (\lambda, X_{r_1} Z, \lambda), \\ \text{form-(2)} & (\lambda, X_{r_2} Y, \lambda). \end{array}$$

- for each production rule $r : X \to a \in P$ with $X \in N$ and $a \in T$, we construct the following insertion rules

$$\text{form-(3)} \quad (\lambda, X_{r_3} a, \lambda).$$

- for the production rule $r : S \to \lambda \in P$, we construct the following insertion rule

$$\text{form-(4)} \quad (\lambda, S_r, \lambda).$$

We define the projection $h : \Sigma^* \to T^*$ by

$$\begin{array}{ll} h(a) = a & \text{for all } a \in T, \\ h(a) = \lambda & \text{otherwise.} \end{array}$$

Consider $R = A\Sigma^* \cap \Sigma^* B - \Sigma^+ C' \Sigma^+$ with $C' = \Sigma^2 - C$, where

$$\begin{array}{rl} A = & \{SS_{r_1} \mid r : S \to YZ \in P\} \cup \{SS_{r_3} \mid r : S \to a \in P\} \cup \\ & \{SS_r \mid r : S \to \lambda \in P\}, \\ B = & \{X_{r_3}a \mid r : X \to a \in P\} \cup \\ & \{SS_r \mid r : S \to \lambda \in P\}, \\ C = & \{XX_{r_1}, X_{r_1}X_{r_2}, X_{r_2}Y \mid r : X \to YZ \in P\} \cup \\ & \{XX_{r_3}, X_{r_3}a \mid r : X \to a \in P\} \cup \\ & \{aX \mid a \in T, X \in N\}. \end{array}$$

Then R is a strictly 2-testable language prescribed by $S_2 = (A, B, C)$. The language R can be characterized by using

$$\begin{array}{rl} \Omega_1 = & \{XX_{r_1}X_{r_2} \mid r : X \to YZ \in P\}, \\ \Omega_2 = & \{XX_{r_3}a \mid r : X \to a \in P\}, \end{array}$$

such that $R \subset (\Omega_1 \cup \Omega_2)^* \Omega_2 \cup \{SS_r \mid r : S \to \lambda \in P\}$.

A nonterminal symbol X in $XX_{r_1}X_{r_2} \in \Omega_1$ or $XX_{r_3}a \in \Omega_2$ is said to be Ω-*blocked*. A symbol in N which is not Ω-blocked is said to be *unblocked*. We call a string in $(\Omega_1 \cup \Omega_2 \cup N)^*$ a *legal string*.

Intuitively, an Ω-blocked nonterminal symbol X in $XX_{r_1}X_{r_2}$ or $XX_{r_3}a$ means that X has been used for the rule r. In γ at each step a string consisting of unblocked symbols and terminal symbols of a legal string indicates a sentential form of G.

Further, based on γ and R, we define the followings: for each $X \in N$, let

$$\gamma_X = (\Sigma, P_\gamma, \{X\})$$

be an insertion grammar, and let

$$R_X = A_X \Sigma^* \cap \Sigma^* B_X - \Sigma^* C' \Sigma^*$$

be a strictly 2-testable language, where

$$\begin{array}{rl} A_X = & \{XX_{r_1} \mid r : X \to YZ \in P\} \cup \{XX_{r_3} \mid r : X \to a \in P\}, \\ B_X = & B - \{SS_r \mid r : S \to \lambda \in P\}. \end{array}$$

Then R_X is a strictly 2-testable language prescribed by $S_X = (A_X, B_X, C)$. The language R_X satisfies $R_X \subset (\Omega_1 \cup \Omega_2)^*\Omega_2$.

From the above definitions, for any $X \in N$, $A_X \cup B_X \subset C$ holds.

Lemma 7. *For any γ_W and a legal string w with $W \Longrightarrow^*_{\gamma_W} w$, a form-(1) rule $(\lambda, X_{r_1}Z, \lambda)$ is applied if and only if the form-(2) rule $(\lambda, X_{r_2}Y, \lambda)$ inserts the string $X_{r_2}Y$ just right to X_{r_1}.*

Proof. If part: Since the symbol X_{r_2} always follows the symbol X_{r_1} in a legal string, from the definition of P_γ, the form-(2) rule $(\lambda, X_{r_2}Y, \lambda)$ can insert the string $X_{r_2}Y$ in the presence of the symbol X_{r_1}. Therefore, the form-(1) rule $(\lambda, X_{r_1}Z, \lambda)$ has been applied before applying $(\lambda, X_{r_2}Y, \lambda)$.

Only if part: Consider an insertion rule $(\lambda, X_{r_1}Z, \lambda)$ in form-(1). For any legal string in $(\Omega_1 \cup \Omega_2 \cup N)^*$, the symbol X_{r_1} is always followed by the symbol X_{r_2}. From the definition of P_γ, the form-(2) rule $(\lambda, X_{r_2}Y, \lambda)$ should insert the string $X_{r_2}Y$ just right to X_{r_1}, then we obtain $X_{r_1}X_{r_2}Y$. □

From Lemma 7, without loss of generality, for r-pair rules r_1 in form-(1) and r_2 in form-(2), we may apply r_2 immediately after applying the rule r_1 to obtain a legal string.

Lemma 8. *For any γ_W, a legal string w with $W \overset{\sigma}{\Longrightarrow}_{\gamma_W} w$, and a substring $X_{r_3}a$ in $(\lambda, X_{r_3}a, \lambda) \in P_\gamma$, no insertion rule inserts a string in between X_{r_3} and a in σ.*

Proof. For any legal string in $(\Omega_1 \cup \Omega_2 \cup N)^*$, a symbol X_{r_3} is always followed by a terminal symbol in T. The claim is almost obvious from the definition of insertion rules in P_γ. □

From Lemma 8, without loss of generality, we may consider the derivation which satisfies the property that once a form-(3) rule is applied then no rule in form-(1),(2),(4) is applied.

Definition 1. For any X in N and w in R_X, a derivation $X = \alpha_0 \Longrightarrow^*_{\gamma_X} \alpha_1 \Longrightarrow^*_{\gamma_X} \cdots \Longrightarrow^*_{\gamma_X} \alpha_n = w$ is called a *standard derivation*, if it satisfies the following conditions:

1. α_i is a legal string $(1 \leq \forall i \leq n)$.
2. No intermediate string appearing between $\alpha_i \Longrightarrow^*_{\gamma_X} \alpha_{i+1}$ is legal $(0 \leq i \leq n-1)$.

3. For each derivation $\alpha_i \overset{\sigma_i}{\Longrightarrow}_{\gamma_X} \alpha_{i+1}$ $(0 \leq \forall i \leq n-1)$, σ_i is one of the following forms;
   - $\sigma_i = p_1 p_2$, where p_1 and p_2 are r-pair insertion rules such that p_1 is in form-(1) and p_2 is in form-(2),
   - $\sigma_i = p_3$, where p_3 is a form-(3) rule.
4. Once a form-(3) rule is applied in $\alpha_i \Longrightarrow^*_{\gamma_X} \alpha_{i+1}$ $(0 \leq i \leq n-1)$, then no rule in form-(1) or form-(2) is applied in $\alpha_{i+1} \Longrightarrow^*_{\gamma_X} \alpha_n$.
5. No insertion rule splits any string in $\Omega_1 \cup \Omega_2$.

Lemma 9. *For any γ_X and w in $L(\gamma_X) \cap (\Omega_1 \cup \Omega_2)^*$, there is a standard derivation for w.*

Proof. Consider γ_X and a string w in $L(\gamma_X) \cap (\Omega_1 \cup \Omega_2)^*$ such that $X \overset{\sigma}{\Longrightarrow}_{\gamma_X} w$. From Lemma 7 and Lemma 8, we prove that no insertion rule inserts a string across the string in $\Omega_1 \cup \Omega_2$ by the induction on the number n of r-pair insertion rules in the derivation σ.

Base step: Since there are no r-pair insertion rules in σ, w is in $L(\gamma_X) \cap \Omega_2^*$. From the definition of P_γ, a form-(3) rule inserts a string in $N_\gamma T$. Further $\Omega_2 \subset (NN_\gamma T)^*$ holds. Then we have $w = XX_{r_3}a$ for a form-(3) rule $(\lambda, X_{r_3}a, \lambda)$, with $X \Longrightarrow_{\gamma_X} XX_{r_3}a$, which gives a standard derivation for w.

Induction step: Suppose that the claim holds for any $n \leq k$. Consider a derivation

$$X = \alpha_0 \overset{\sigma_1}{\Longrightarrow}_{\gamma_X} \alpha_1 \cdots \overset{\sigma_k}{\Longrightarrow}_{\gamma_X} \alpha_k \overset{\sigma_{k+1}}{\Longrightarrow}_{\gamma_X} w,$$

where σ_i consists of r-pair insertion rules for $1 \leq i \leq k$, σ_{k+1} consists of form-(3) rules, and $\alpha_1 = XW_{r_1}W_{r_2}YZ$.

There are the following two cases for α_k:

1. $W = X$ and $\alpha_k = XX_{r_1}X_{r_2}YyZz$, where $Yy, Zz \in (\Omega_1 \cup N)^*$.
   In this case, we have a derivation

   $$\alpha_k = XX_{r_1}X_{r_2}YyZz \Longrightarrow_{\gamma_X} XX_{r_1}X_{r_2}Yy'Zz' = w,$$

   where $Yy', Zz' \in (\Omega_1 \cup \Omega_2)^*$. From the induction hypothesis for $Y \Longrightarrow^*_{\gamma_Y} Yy'$ and $Z \Longrightarrow^*_{\gamma_Z} Zz'$, there are standard derivations σ_Y and σ_Z for Yy' and Zz', respectively. Therefore, $\sigma_1\sigma_Y\sigma_Z$ is a standard derivation for w through the legal string $\alpha_1 = XX_{r_1}X_{r_2}YZ$.

2. $W \neq X$ and $\alpha_k = XxWW_{r_1}W_{r_2}YyZz$, where $XxWW_{r_1}W_{r_2} \in (\Omega_1 \cup N)^*\Omega_1$ and $YyZz \in (\Omega_1 \cup N)^*$.
   Since the substring xW is inserted by r-pair rules in P_γ, the string α_k satisfies $\alpha_k = XX_{p_1}X_{p_2}\beta WW_{r_1}W_{r_2}YyZz$ for $(\lambda, X_{p_1}Z', \lambda)$, $(\lambda, X_{p_2}Y', \lambda) \in P_\gamma$ and $\beta \in (\Omega_1 \cup N)^*$. Let us consider a derivation

$$\begin{aligned} X \overset{\sigma_X}{\Longrightarrow}_{\gamma_X} &\; XX_{p_1}X_{p_2}Y'Z' \\ \Longrightarrow^*_{\gamma_X} &\; XX_{p_1}X_{p_2}\beta W \\ \overset{\sigma_1}{\Longrightarrow}_{\gamma_X} &\; XX_{p_1}X_{p_2}\beta WW_{r_1}W_{r_2}YZ \\ \Longrightarrow^*_{\gamma_X} &\; XX_{p_1}X_{p_2}\beta WW_{r_1}W_{r_2}YyZz = \alpha_k \\ \Longrightarrow^*_{\gamma_X} &\; w. \end{aligned}$$

   Let y' and z' be strings in $(\Omega_1 \cup \Omega_2)^*$ such that $XX_{p_1}X_{p_2}y'z' = w$ and $Y' \Longrightarrow^*_{\gamma_{Y'}} y'$, $Z' \Longrightarrow^*_{\gamma_{Z'}} z'$.
   From the induction hypothesis for y' and z', there are standard derivations $\sigma_{Y'}$, $\sigma_{Z'}$ such that $Y' \overset{\sigma_{Y'}}{\Longrightarrow}_{\gamma_{Y'}} y'$, $Z' \overset{\sigma_{Z'}}{\Longrightarrow}_{\gamma_{Z'}} z'$, respectively. Therefore, $\sigma_X\sigma_{Y'}\sigma_{Z'}$ is a standard derivation for w through the legal string $\alpha_1 = XX_{p_1}X_{p_2}Y'Z'$.

□

The following two lemmata are essential for the proof of Theorem 5.

Lemma 10. *For any X in N, if there is a derivation $X \Longrightarrow^*_G w$ with $w \in T^+$ then there is a string w' in $L(\gamma_X) \cap R_X$ such that $h(w') = w$.*

Proof. We will show that, for any X in N, if there is a derivation $X \overset{r_1 \cdots r_n}{\Longrightarrow}_G a_1 \cdots a_l$ with $a_i \in T$ $(l \geq 1, 1 \leq i \leq l)$ then there is a string w' in $L(\gamma_X) \cap R_X$ such that $h(w') = a_1 \cdots a_l$ by the induction on n.

Base step: Consider a derivation $X \overset{r}{\Longrightarrow}_G a$. From the definition of P_γ, an insertion rule $(\lambda, X_{r_3}a, \lambda)$ is in P_γ. By the construction of R_X, $XX_{r_3} \in A_X$ and $X_{r_3}a \in B_X$ hold. Then the string $XX_{r_3}a$ in R_X satisfies that $X \Longrightarrow_\gamma XX_{r_3}a$ and $h(XX_{r_3}a) = a$.

Induction step: We suppose that the claim holds for any $n \leq k$. Consider a string yz which satisfies that $X \Longrightarrow_G YZ \overset{r_1 \cdots r_j}{\Longrightarrow}_G yZ \overset{r_{j+1} \cdots r_k}{\Longrightarrow}_G yz$, where $r_i \in P$ for each $1 \leq i \leq k$ and $1 \leq j < k$. For the derivations $Y \overset{r_1 \cdots r_j}{\Longrightarrow}_G y$ and $Z \overset{r_{j+1} \cdots r_k}{\Longrightarrow}_G z$, from the induction hypothesis, there are strings y' and z' such that $y' \in L(\gamma_Y) \cap R_Y$, $z' \in L(\gamma_Z) \cap R_Z$, and $h(y') = y$, $h(z') = z$.

For the production rule $r : X \to YZ$, r-pair insertion rules $(\lambda, X_{r_1}Z, \lambda)$ and $(\lambda, X_{r_2}Y, \lambda)$ are in P_γ. Then, there is a derivation

$$X \Longrightarrow_{\gamma_X} XX_{r_1}Z \Longrightarrow_{\gamma_X} XX_{r_1}X_{r_2}YZ \Longrightarrow^*_{\gamma_X} XX_{r_1}X_{r_2}y'z'.$$

Further, for the production rule $r : X \to YZ$, we have $XX_{r_1} \in A_X$, $X_{r_1}X_{r_2}, X_{r_2}Y \in C$. Note that the following holds:

$$A_Y \cup B_Y \cup \{aZ \mid a \in T\} \cup A_Z \subset C.$$

Therefore, the string $XX_{r_1}X_{r_2}y'z'$ in $L(\gamma_X) \cap R_X$, satisfies $h(XX_{r_1}X_{r_2}y'z')$ $= h(y')h(z') = yz$. □

Lemma 11. *For any γ_X, if a nonempty string w' is in $L(\gamma_X) \cap (\Omega_1 \cup \Omega_2)^*$, then there is a derivation $X \Longrightarrow^*_G h(w')$.*

Proof. Consider X in N and a nonempty string $w' \in L(\gamma_X) \cap (\Omega_1 \cup \Omega_2)^*$. From Lemma 9, without loss of generality, we may consider a standard derivation $X \Longrightarrow^*_{\gamma_X} \alpha_1 \Longrightarrow^*_{\gamma_X} \alpha_2 \Longrightarrow^*_{\gamma_X} \cdots \Longrightarrow^*_{\gamma_X} \alpha_n = w'$, where $n \geq 1$ and α_i is a legal string for each $1 \leq i \leq n$. We will show that there is a derivation $X \Longrightarrow^*_G h(w')$ by the induction on n.

Base step: Consider a standard derivation $X \overset{\sigma}{\Longrightarrow}_{\gamma_X} XX_{r_3}a$, where an insertion rule $(\lambda, X_{r_3}a, \lambda)$ is used in σ. Then a production rule $r : X \to a$ is in P, which implies a derivation $X \Longrightarrow_G a$, where $h(XX_{r_3}a) = a$.

Induction step: Consider a standard derivation $X \overset{\sigma_1}{\Longrightarrow}_\gamma \alpha_1 \overset{\sigma}{\Longrightarrow}_\gamma \alpha_{n+1} = w'$.

Suppose that a form-(3) rule is applied in σ_1, then there is no derivation $\alpha_1 \overset{\sigma}{\Longrightarrow} w'$, where σ consists of form-(3) rules. Then r-pair insertion rules are used in σ_1 and let $\alpha_1 = XX_{r_1}X_{r_2}YZ$ and $\alpha_{n+1} = XX_{r_1}X_{r_2}Yy'Zz'$. For the r-pair insertion rules $(\lambda, X_{r_1}Z, \lambda)$ and $(\lambda, X_{r_2}Y, \lambda)$, there is a production rule $r : X \to YZ$ in P.

For the strings y' and z', we have $Yy' \in L(\gamma_Y) \cap (\Omega_1 \cup \Omega_2)^*$ and $Zz' \in L(\gamma_Z) \cap (\Omega_1 \cup \Omega_2)^*$. From the induction hypothesis, there are derivations $Y \Longrightarrow^*_G y$ and $Z \Longrightarrow^*_G z$ such that $h(Yy') = y$ and $h(Zz') = z$.

Therefore, there is a derivation $X \Longrightarrow_G YZ \Longrightarrow^*_G yz$, where $h(w') = h(XX_{r_1}X_{r_2}Yy'Zz') = h(y')h(z') = yz$. □

Proof of Theorem 5. Let us consider the case where λ is in $L(G)$. Since G is in Chomsky normal form, λ is in $L(G)$ if and only if there is a derivation $S \overset{r}{\Longrightarrow} \lambda$ for $r : S \to \lambda$ in P. By the construction of P_γ and R, the string λ is in $L(G)$ if and only if $(\lambda, S_r, \lambda) \in P_\lambda$ and $SS_r \in A \cap B$. Then there is a derivation $S \Longrightarrow_\gamma SS_r \in A \cap B$. From the definition of h, the string SS_r satisfies $h(SS_r) = \lambda$. Therefore, λ is in $L(G)$ if and only if λ is in $h(L(\gamma) \cap R)$. We slightly note that $R \subset (\Omega_1 \cup \Omega_2)^* \Omega_2 \cup \{SS_r \mid r : S \to \lambda \in P\}$ implies that no string in $(\Sigma - T)^*$ satisfies $L(\gamma) \cap R$ other than SS_r.

From Lemma 10, Lemma 11, and the fact $R_X \subset (\Omega_1 \cup \Omega_2)^* \Omega_2 \subset (\Omega_1 \cup \Omega_2)^*$, considering the case $X = S$, a nonempty string w is in $L(G)$ if and only if there is a string w' such that $w' \in L(\gamma) \cap R$ and $h(w') = w$. □

Since the class of context-free languages is closed under intersection with regular languages and morphism, the fact $INS_2^0 \subset CF$ implies $H(INS_2^0 \cap LOC(2)) \subseteq CF$. Therefore, from Theorem 5, we have the following theorem.

Theorem 6. $CF = (INS_2^0 \cap LOC(2))$

Furthermore, from the fact that, for arbitrary k and i with $k \geq 1$ and $i \geq 1$, the class of regular languages includes $LOC(k)$ in Theorem 1 and the class of context-free languages includes INS_i^0 in Theorem 2, we have the following corollary.

Corollary 4. $CF = H(INS_i^0 \cap LOC(k))$ ($i, k \geq 2$).

From Lemma 4, Theorem 6, and the fact that $INS_i^0 \subseteq INS_{i+1}^0$ with $i \geq 1$, we have the following corollary.

Corollary 5. $H(INS_i^0 \cap LOC(1)) \subset CF$ ($i \geq 2$).

5. Characterizations of RE Languages

In this section, we will show that any recursively enumerable language can be represented by using insertion systems and strictly locally testable languages in the similar way to context-free and regular languages.

Theorem 7. $RE = H(INS_3^3 \cap LOC(2))$.

Construction of an insertion system γ: Let $G = (N, T, P, S)$ be a type-0 grammar in Penttonen normal form [3]. In this normal form, the rules in P are of the following types:

Type 1 : $X \to \alpha \in P$, where $X \in N$, $\alpha \in (N \cup T)^*$, $|\alpha| \leq 2$.
Type 2 : $XY \to XZ \in P$, where $X, Y, Z \in N$.

By introducing new symbols # and c, we construct the insertion system $\gamma = (\Sigma, P_\gamma, \{Scc\})$, where $\Sigma = N \cup T \cup \{\#, c\}$ and P_γ contains the following insertion rules:

- Group 1: For each rule $r : X \to YZ \in P$ of Type 1, with $X \in N$ and $Y, Z \in N \cup T \cup \{\lambda\}$, we construct the following insertion rules

  $$\text{form-}(r_1) \quad (X, \#YZ, \alpha_1\alpha_2) \text{ in } P_\gamma, \text{ where } \alpha_1\alpha_2 \in (N \cup T \cup \{c\})^2.$$

- Group 2: For each rule $r : XY \to XZ \in P$ of Type 2, with $X, Y, Z \in N$, we construct the following insertion rules

  $$\text{form-}(r_2) \quad (XY, \#Z, \alpha_1\alpha_2) \text{ in } P_\gamma, \text{ where } \alpha_1\alpha_2 \in (N \cup T \cup \{c\})^2.$$

- Group 3 (Relocation task for X): For each $X, Y \in N$, we construct the following insertion rules

  $$\begin{array}{ll} \text{form-}(r_3) & (XY\#, \#X, \alpha), \text{ where } \alpha \in (N \cup T \cup \{c\}), \\ \text{form-}(r_4) & (X, \#, Y\#\#), \\ \text{form-}(r_5) & (\#Y\#, Y, \#X). \end{array}$$

We define a projection $h : \Sigma^* \to T^*$ by

$$\begin{array}{ll} h(a) = a & \text{for all } a \in T, \\ h(a) = \lambda & \text{otherwise.} \end{array}$$

Finally, let $R = A\Sigma^* \cap \Sigma^* B - \Sigma^+ C' \Sigma^+$ with $C' = \Sigma^2 - C$,

$$\begin{array}{rl} A = & \{X\# \mid X \in N\}, \\ B = & \{cc\}, \\ C = & \{X\# \mid X \in N\} \cup \{\#X \mid X \in N\} \cup \{aX \mid X \in N\} \cup \\ & \{ab \mid a, b \in T\} \cup \{ac \mid a \in T\} \cup \{\#a \mid a \in T\} \cup \{\#c\}. \end{array}$$

Then R is a strictly 2-testable language prescribed by $S_2 = (A, B, C)$. The language R can be represented by $R = N\{\#\}(T \cup N\{\#\})^*\{cc\}$.

Then we obtain $L(G) = h(L(\gamma) \cap R)$, which will be proven in the sequel. We start by introducing some useful notions.

We call the symbol # a *marker*. A symbol in N followed by # is said to be *#-marked* (briefly *marked*). A symbol in $N \cup T$ which is not marked is said to be *unmarked*. We call a string in $N\{\#\}$ a *wreck* and a string in

$(N\{\#\})^+$ a *wrecks*. Since the symbols c and # are special symbols, they are neither marked nor unmarked. A string xcc, where x is in $(N\{\#\} \cup N \cup T)^*$, is a *legal* string.

An intuitive explanation of marked symbols, unmarked symbols, and a wreck is the followings:

Note 1. A marked symbol means that the symbol has been used (i.e. consumed) for some derivation in γ.

Note 2. In γ at each step a wreck is considered to be a "garbage" and a string consisting of unmarked symbols of a legal string indicates a sentential form of G.

By the construction of R, making $L(\gamma) \cap R$ leads to only legal strings. Then if we erase the "wrecks" and the symbol c, we get the legal strings of unmarked symbols which are exactly sentential forms of G.

By using the rules of Group 1 and Group 2, we can simulate the rules of Type 1 and Type 2 respectively. By using the rules of Group 3, we move an unmarked symbol to the right across a block $M\#$, where $M \in N$. Thus the nonterminal pairs XY can be ready for simulating the rules $XY \to YZ$ of Type 2.

In order to prove the equality $L(G) = h(L(\gamma) \cap R)$, we first prove the inclusion $L(G) \subseteq h(L(\gamma) \cap R)$.

Fact 1. Applying a form-(r_1) rule : $(X, \#YZ, \alpha_1\alpha_2)$ to an occurrence of a string $X\alpha_1\alpha_2$ with $\alpha_1\alpha_2 \in (N \cup T \cup \{c\})^2$ makes a new occurrence of the string $X\#YZ\alpha_1\alpha_2$. Note that the unmarked symbol X becomes marked, while the symbols Y, Z are newly created unmarked symbols.

Fact 2. Applying a form-(r_2) rule : $(XY, \#Z, \alpha_1\alpha_2)$ to an occurrence of a string $XY\alpha_1\alpha_2$ with $\alpha_1\alpha_2 \in (N \cup T \cup \{c\})^2$ makes a new occurrence of the string $XY\#Z\alpha_1\alpha_2$. Note that the symbol X is preserved in just the unmarked state, the unmarked symbol Y becomes marked, while the symbol Z is newly created unmarked symbol.

Lemma 12. *The rules in Group 3 can replace a substring $XY\#\alpha$ ($\alpha \in N \cup T \cup \{c\}$) by a substring consisting of the strings in $N\{\#\}$ and ending with $X\alpha$. The symbol X is unmarked before and after the derivations.*

Proof. A form-(r_3) rule $(XY\#, \#X, \alpha)$ can be applied to a string $XY\#\alpha$, where $X, Y \in N$, $\alpha \in N \cup T \cup \{c\}$. After applying the form-(r_3) rule, we have $XY\#\#X\alpha$. Then the form-(r_4) rule $(X, \#, Y\#\#)$ can be applied for the substring $XY\#\#$, and we have $X\#Y\#\#X\alpha$. Now we apply the form-(r_5) rule $(\#Y\#, Y, \#X)$ for the substring $\#Y\#\#X$, and the substring is replaced by $\#Y\#Y\#X$.

Therefore, the substring $XY\#\alpha$ is replaced by $X\#Y\#Y\#X\alpha$, which has the unmarked symbol X on the rightmost position. □

Thus the insertion rules in γ simulate the rules in G, and generate legal strings from the legal string Scc.

We will give separate consideration to the case of using the rules in Group 3.

Lemma 13. *Once a form-(r_3) rule : $(XY\#, \#X, \alpha)$ is applied to obtain a substring of a legal string, then the form-(r_4) rule and form-(r_5) rule are used in this order.*

Proof. We may consider a substring $XY\#\alpha$, where $X, Y \in N$, $\alpha \in N \cup T \cup \{c\}$. After using rule in form-(r_3), we obtain $XY\#\#X\alpha$. Because of the symbols $\#\#$, rules in form-(r_1) or (r_2) or (r_3) cannot be applied for the substring $XY\#\#$. In view of the construction of form-(r_5) rule, we cannot apply a form-(r_5) rule for $XY\#\#$. Hence, the only applicable rule for $XY\#\#$ is form-(r_4) rule.

After using form-(r_4) rule $(X, \#, Y\#\#)$ for $XY\#\#X\alpha$, we obtain the substring $X\#Y\#\#X\alpha$. For the symbol X following $\#\#$, we have a chance to apply one of the rules in form-(r_1), (r_2), (r_3), (r_4). If we apply form-(r_1) or form-(r_2) rule, we may take it as the first step of simulation for Type 1 or Type 2 respectively. Note that, during these simulations, X remains at the immediately to the right of $\#\#$. If we apply form-(r_3) or form-(r_4) rule, we may take it independently a new relocation task. Note that, after application of form-(r_3) or form-(r_4) rule, X remains immediately to the right of $\#\#$. Therefore, in all cases the symbol $\#\#$ is followed by X. Further, since the symbol X was originally unmarked in $XY\#\alpha$, X provides the possibility of applying one of the rules in form-(r_1), (r_2), (r_3), (r_4). Hence this application causes no trouble with the current relocation task.

After using form-(r_4) rule for $XY\#\#$, we obtain $X\#Y\#\#$. From the above notation, since X always follows the symbols $\#\#$, after applying form-

(r_4) rule, we obtain $X\#Y\#\#X$. In the substring $X\#Y\#\#$, both of the symbols X and Y are already marked, and in view of the form of the rules, none of form-(r_1), (r_2), (r_3), (r_4) rule can be used for this substring. Hence, the only applicable rule for $X\#Y\#\#X$ is form-(r_5) rule. After applying this rule, $(\#Y\#, Y, \#X)$, we have $X\#Y\#X\#X$, which is the substring of a legal string.

Hence to obtain a substring of a legal string, whenever we use the form-(r_3) rule, we have to use form-(r_4) rule and form-(r_5) rule in this order. □

From Lemma 13, for any derivation in γ, $x \overset{\pi}{\Longrightarrow}_\gamma y$, there is a *standard derivation* which satisfies that form-(r_4) rule and form-(r_5) rule are applied in this order immediately after applying form-(r_3) rule.

Denote by $umk(x)$ a string consisting of unmarked symbols in a legal string x generated by γ. Note that since c is the special symbol, neither marked nor unmarked, $umk(x)$ does not contain a suffix cc. We thus have the next lemma.

Lemma 14. *The nonterminal symbol S derives x in G if and only if there is a derivation $Scc \Longrightarrow_\gamma^* x'$ in γ such that $umk(x') = x$.*

Proof. We will show that if there is a derivation $S \Longrightarrow_G^n x$ with $x \in (N \cup T)^*$ then there is a derivation $Scc \Longrightarrow_\gamma^* x'$ such that $umk(x') = x$ and $x' \in \Sigma^*$ by induction on n.

Base step: If $n = 0$, then for the axiom Scc in γ, $umk(Scc) = S$ holds. Thus obviously the claim holds.

Induction step: We suppose that the claim holds for any $n \leq k$. Now consider a derivation $S \Longrightarrow_G^k x \Longrightarrow_G y$ with $x, y \in (N \cup T)^*$.

From the induction hypothesis, there is a derivation $Scc \Longrightarrow_\gamma^* x'$, where $umk(x') = x$ and $x' \in \Sigma^*$. If the rule applied for x is of Type 1 (Type 2, resp.) then we use the corresponding insertion rule in Group 1 (Group 2, resp.) for the string x'.

However, in the latter case (i.e. Group 2), if the insertion rule in Group 2 cannot be immediately applied for x', we need to apply some rules in Group 3. From Lemma 12, after application of the rules in Group 3, unmarked symbols of a legal string x' remain unchanged. We denote this process of derivations by $x' \Longrightarrow_\gamma^* x'' \Longrightarrow_\gamma y'$, where x'', a string ready for applying a rule in Group 2, is derived by using only rules in form-(r_3), (r_4), (r_5) in Group 3 and y' is derived by using only a rule in Group 2. Note that $umk(x') = umk(x'')$.

Then, in either case, from Fact 1 and Fact 2 we eventually have $umk(y') = y$. Therefore the claim holds for $k+1$.

Conversely, we will show that if there is a standard derivation $Scc \overset{\pi}{\Longrightarrow}_{\gamma} x'$ with $x' \in \Sigma^*$ then there is a derivation $S \Longrightarrow_G^* x$ such that $umk(x') = x$ and $x \in (N \cup T)^*$ by induction on the number n of legal strings in the derivation π.

Base step: For the axiom Scc, no rules in form-(r_3) or (r_4) or (r_5) can apply. Further, $umk(Scc) = S$ holds and Scc is legal. Thus, obviously the claim holds.

Induction step: We suppose that the claim holds for any $n \leq k$. Now consider a standard derivation $Scc \overset{\pi_1}{\Longrightarrow}_{\gamma} x' \overset{\pi_2}{\Longrightarrow}_{\gamma} y'$, where x' is the k-th legal string in π_1 and y' is the first legal string in π_2 with $x', y' \in \Sigma^*$. From the induction hypothesis, there is a derivation $S \Longrightarrow_G^* x$, where $umk(x') = x$.

Let r' denote the production which was applied first in π_2. Note that no rule in form-(r_4) or form-(r_5) can apply for legal strings. For the insertion rule r' of Group 1 (Group 2, resp.), there is the corresponding production rule in Type 1 (Type 2, resp.) for the string x. In either case, from Fact 1 and Fact 2 we eventually have $x \Longrightarrow_G^* y$, where $umk(y') = y$.

In case that the insertion rule r' is in Group 3 (i.e. r' is form-(r_3) rule), for standard derivation $x' \overset{\pi_2}{\Longrightarrow}_{\gamma} y'$ form-(r_4) rule and form-(r_5) rule are applied in this order. From Lemma 12, after application of the rules in Group 3, unmarked symbols of a legal string x' remains unchanged. Note that $umk(x') = umk(y')$. Then, from the induction hypothesis, there is a derivation $S \Longrightarrow_G^* x$ such that $umk(x') = umk(y') = x$.

Therefore the claim holds for $k+1$. □

In view of the manner of constructing the strictly 2-testable language R and the projection h, we have the following fact.

Fact 3. For any $y \in L(\gamma)$, if y is in R and $umk(y) \in T^*$, then $umk(y) = h(y)$.

From Lemma 14 and Fact 3, we obtain the inclusion $L(G) \subseteq h(L(\gamma) \cap R)$. Next we prove the inverse inclusion which completes the proof of Theorem 6.

Fact 4. As far as unmarked symbols are concerned, the rules in Group 1 and Group 2 can only simulate the rules of Type 1 and Type 2 respectively in G.

Proof of Theorem 6. From Fact 4, Lemma 13 and Lemma 14, every string of a form $umk(xcc)$ is generated by the grammar G, where xcc is a legal string generated by γ.

Therefore, if for any $y \in L(\gamma)$, y is in R, then there is a string $h(y)$ such that $S \Longrightarrow_G^* h(y)$. This means that the inclusion $h(L(\gamma) \cap R) \subseteq L(G)$ holds. Together with the fact that $L(G) \subseteq h(L(\gamma) \cap R)$, we complete the proof of Theorem 6. □

Corollary 6. $RE = H(INS_3^3 \cap LOC(k))$ ($k \geq 2$).

6. Conclusion

In this paper, we have contributed to the study of insertion systems with new characterizations of recursively enumerable, context-free, and regular languages. Specifically, we have shown that

$REG = H(INS_1^0 \cap LOC(k))$ with $k \geq 2$.
$H(INS_1^0 \cap LOC(1)) \subset REG \subset H(INS_i^0 \cap LOC(k))$ with $i, k \geq 2$.
$CF = H(INS_i^0 \cap LOC(k))$ with $i, k \geq 2$.
$RE = H(INS_3^3 \cap LOC(k))$ with $k \geq 2$.

The followings are open problems:

- Can CF be represented as $CF = H(INS_i^j \cap LOC(k))$ for some $i, j, k \geq 1$?
- Can RE be represented as $RE = H(INS_i^j \cap LOC(2))$ for some $i < 3$ or $j < 3$?
- Whether CS (the class of context-sensitive languages) can be represented as $CS = H(INS_i^j \cap LOC(k))$ for some $i, j \geq 0$, $k \geq 1$?

Acknowledgements

The author is deeply indebted to T.Yokomori for his helpful discussions.

$$\begin{tikzcd}
 & \begin{array}{c} RE = H(INS_3^3 \cap LOC(k)) \\ (\forall k \geq 2) \end{array} & \\
 & \cdots \arrow[u] & \\
 & \begin{array}{c} CF = H(INS_2^0 \cap LOC(k)) \\ (\forall k \geq 2) \end{array} \arrow[u] & \\
\begin{array}{c} REG = H(INS_1^0 \cap LOC(k)) \\ (\forall k \geq 2) \end{array} \arrow[ur] \arrow[rr, dash, dashed, "\text{incomparable}"'] & & H(INS_2^0 \cap LOC(1)) \arrow[ul] \\
 & H(INS_1^0 \cap LOC(1)) \arrow[ul] \arrow[ur] &
\end{tikzcd}$$

Figure 1: Relationships between the classes of languages generated by insertion systems and strictly k-testable languages

References

[1] K. Onodera, Lecture Notes in Computer Science 5457 (2009) 648–659.

[2] M. Margenstern, G. Păun, Y. Rogozhin, S. Verlan, Theor. Comput. Sci. 330 (2005) 339–348.

[3] G. Păun, G. Rozenberg, A. Salomaa, DNA Computing. New Computing Paradigms., Springer, 1998.

[4] T. Yokomori, S. Kobayashi, IEEE Trans. Pattern Anal. Mach. Intell. 20 (1998) 1067–1079.

[5] T. Head, Discrete Applied Math 87 (1998) 87–139.

[6] C. Martin-Vide, G. Păun, A. Salomaa, Theor. Comput. Sci. 205 (1998) 195–205.

[7] K. Onodera, IPSJ Journal 44 (2003) 1424–1427.

[8] G. Păun, M. J. Pérez-Jiménez, T. Yokomori, Int. J. Found. Comput. Sci. 19 (2008) 859–871.

[9] G. Rozenberg, A. Salomaa (Eds.), Handbook of formal languages, Springer-Verlag New York, Inc., New York, NY, USA, 1997.

[10] R. McNaughton, S. A. Papert, Counter-Free Automata (M.I.T. research monograph no. 65), The MIT Press, 1971.