Ph.D. dissertation

TITLE

Recognition of Parallel Multiple Context-Free Grammars and Finite State Translation Systems

supervisor
Professor Tadao Kasami

Yuichi Kaji

January 28, 1994

Osaka University

Recognition of Parallel Multiple Context-Free Grammars and Finite State Translation Systems

Yuichi Kaji

Abstract

A number of computational models whose generative powers are between context-free grammars and context-sensitive grammars have been proposed, and the classes of language generated (or accepted) by those models have been widely studied. Among them, the class of *parallel multiple context-free grammars* (*PMCFG*), which was introduced to describe the syntax of natural languages, is an interesting formalism.

In the first half of this dissertation, computational complexities of the universal recognition problems for PMCFG and its subclasses are investigated. The *universal recognition problem* for a class $\mathcal{G}$ of grammars is the one to decide, taking a grammar $G \in \mathcal{G}$ and a string w as an input, whether G can generate w or not. Characteristics of PMCFG, relations among subclasses of PMCFG, and their significance in natural language processing are discussed based on theoretical results.

The latter half of this dissertation is devoted to a study on the generative powers of subclasses of *finite-state translation systems* (*FSTS*) which was introduced as a model of transformational grammars. It is shown that deterministic FSTS has the same generative power as that of PMCFG. As a corollary, any yield language generated by deterministic FSTS is recognizable in $O(n^{e+1})$-time where n is the length of an input word and e is a constant called the degree of the deterministic FSTS. It is also shown that there is a nondeterministic finite-state translation system that generates an $\mathcal{NP}$-complete language even if very strong constraint, namely, monadicness and two state-boundness, is assumed on the systems.

Keywords

formal language, parallel multiple context-free grammar, universal recognition problem, finite-state translation system, computational complexity

Publication List

Journal Papers (Reviewed)

[1] Kaji Y., Nakanishi R., Seki H. and Kasami T.: "The Universal Recognition Problems for Multiple Context-Free Grammars and for Linear Context-Free Rewriting Systems", IEICE Transaction on Information and Systems, **E75-D**, 1, pp.78–88 (Jan. 1992).

[2] Kaji Y., Nakanishi R., Seki H. and Kasami T.: "The Universal Recognition Problems for Parallel Multiple Context-Free Grammars and for Their Subclasses", IEICE Transaction on Information and Systems, **E75-D**, 7, pp.499-508 (July 1992).

[3] Kaji Y., Nakanishi R., Seki H. and Kasami T.: "The Universal Recognition Problem for Parallel Multiple Context-Free Grammars", Computational Intelligence, Special Issue on Papers Presented at TAG+ Workshop, Springer-Verlag (to appear).

[4] Kaji Y., Seki H. and Kasami T.: "Finite State Translation Systems and Parallel Multiple Context-Free Grammars", IEICE Transaction on Information and Systems (to appear).

International Conferences

[5] Kaji Y., Nakanishi R., Seki H. and Kasami T.: "The Computational Complexity of the Universal Recognition Problem for Parallel Multiple Context-Free Grammars", Abstracts of TAG+ Workshop, pp.25.1–25.2, University of Pennsylvania, Philadelphia (June 1992, Invited).

[6] Sections 3 and 6 of: Seki H., Nakanishi R., Kaji Y., Ando S. and Kasami T.: "Parallel Multiple Context-Free Grammars, Finite-State Translation Systems, and Polynomial-Time Recognizable Subclasses of Lexical-Functional Grammmars", Proceedings of 31st Annual Meeting of Association for Computational Linguistics, pp.130–139, Ohio State University, Columbus (June 1993, Reviewed).

Workshops

[7] Kaji Y., Nakanishi R., Seki H. and Kasami T.: "The Universal Recognition Problems for Multiple Context-Free Grammars and for Its Subclasses", IEICE Technical Report, **COMP91**-25 (June 1991).

[8] Kaji Y., Nakanishi R., Seki H. and Kasami T.: "Parallel Multiple Context-Free Grammars and Finite State Translation Systems", IEICE Technical Report, **COMP92**-34 (Sept. 1992).

[9] Nakanishi R., Ando S., Kaji Y., Seki H. and Kasami T.: "On the Generative Capacities of Tree Translation Systems and Lexical-Functional Grammars", Technical Paper of FAI, Japanese Society for Artificial Intelligence, **SIG-FAI-9302** (Feb. 1993).

Acknowledgement

I have been fortunate to have received great support from many individuals during the course of this work. Among them, I would especially like to thank my supervisor, Professor Tadao Kasami for his guidance to this work, continuous support and encouragement. I am also greatly appreciate to Professor Nobuki Tokura, Professor Kenichi Taniguchi, and Professor Tohru Kikuno for their helpful suggestions and valuable comments. I am also very appreciate to Associate Professor Hiroyuki Seki for his insightful comments and valuable discussions on this work.

I have received valuable instructions from a number of professors of Osaka University. I especially thank to Professors of Department of Information and Computer Sciences. I also thank to Professor Aravind K. Joshi of University of Pennsylvania, and Professor David Weir of Sussex University for their helpful support and valuable comments.

I would also like to express my thanks to Dr. Noriya Kobayashi who inspired the significance of universal recognition problems, Dr. Koji Nakano for his kind advise and support, and Mr. Yasunori Ishihara for his helpful comments. I also thank to Dr. Giorgio Satta of University of Pennsylvania for his kind support and valuable information.

I am greatly appreciate to Associate Professors Toru Fujiwara, Toyoo Takata, Research Associates Masahiro Higuchi and Robert Morelos-Zaragoza for their kind supports. I also thank to Ms. Machiko Uehara for valuable assistance. I would also like to express my thanks to my collaborators Ms. Sachiko Ando and Mr. Ryuichi Nakanishi for their helpful comments and valuable discussions.

Finally, special thanks go to the entire staff of the Laboratory of Information Theory and Logics, Department of Information and Computer Sciences, Osaka University.

Contents

1 Introduction **6**

2 Definitions **16**
2.1 Parallel Multiple Context-Free Grammars 16
2.2 Finite State Translation Systems 22

3 Universal Recognition Problems **26**
3.1 General Case 26
3.1.1 Containment in EXP-POLY Time 26
3.1.2 Basic Ideas 33
3.1.3 EXP-POLY time-hardness 39
3.2 With Non-Erasing Condition 47
3.3 Bounded Dimension 51
3.3.1 m-MCFG with $m \geq 2$ 51
3.3.2 m-PMCFG with $m \geq 1$ 61
3.4 Bounded Degree 64
3.5 Results and Their Implication 66

4 Generative Powers of FSTS **68**
4.1 Deterministic FSTS 68
4.1.1 yL(DFSTS) $\subseteq$ PMCFL 68
4.1.2 PMCFL $\subseteq yL$(DFSTS) 74
4.1.3 Recognition of yL(DFSTS) 79
4.2 Monadic FSTS 80

5 Conclusions **83**

References **85**

1 Introduction

Many researchers have investigated the "gap" between the class of context-free languages (CFL) and the class of context-sensitive languages (CSL) [1]. Their studies are motivated by two different interests; an interest from the viewpoint of natural language processing, and an interest from the viewpoint of computational complexity theory.

In the field of natural language processing, it has been often claimed that the generative power of context-free grammars (CFG) is not strong enough to describe the syntax of natural languages; for example, discontinuous phrase structure such as "respectively" sentence cannot be described by any cfg in a simple manner. An example of a "respectively" sentence is: "A dog and cats runs and walk, respectively". In this sentence, "runs" corresponds to "a dog", and "walk" corresponds to "cats". This discontinuous and interleaving phrase structure can be modeled by a formal language $\{ww \mid w \in \Sigma^*\}$ which cannot be generated by any cfg. If one admits that the length of w in this language should be finite in natural languages, then the language can be generated by a cfg. But a derivation tree of the cfg will have a quite unnatural structure as compared with our intuitive understanding on the phrase structure of respectively sentence.

On the other hand, the generative power of context-sensitive grammars (CSG) is too strong for efficient handling. Taking these problems into considerations, a number of new grammatical formalisms of which generative powers are stronger than that of CFG have been proposed. These new grammars include head grammars (HG)[19], tree adjoining grammars (TAG)[25], generalized context-free grammars (GCFG)[19]. Among them, GCFG is a natural extension of CFG, and phrase structure is simply defined in GCFG. However, it was shown in Ref.[13] that GCFG has generative power equal to that of type-0 grammars and hence they cannot be handled efficiently.

To get rid of such intractability of GCFG, a subclass of GCFG, called *parallel multiple context-free grammars (PMCFG)*[13] was introduced. The class of languages generated by pmcfg's is called *parallel multiple context-free languages (PMCFL)*. *Multiple context-free grammars (MCFG)*[13] is

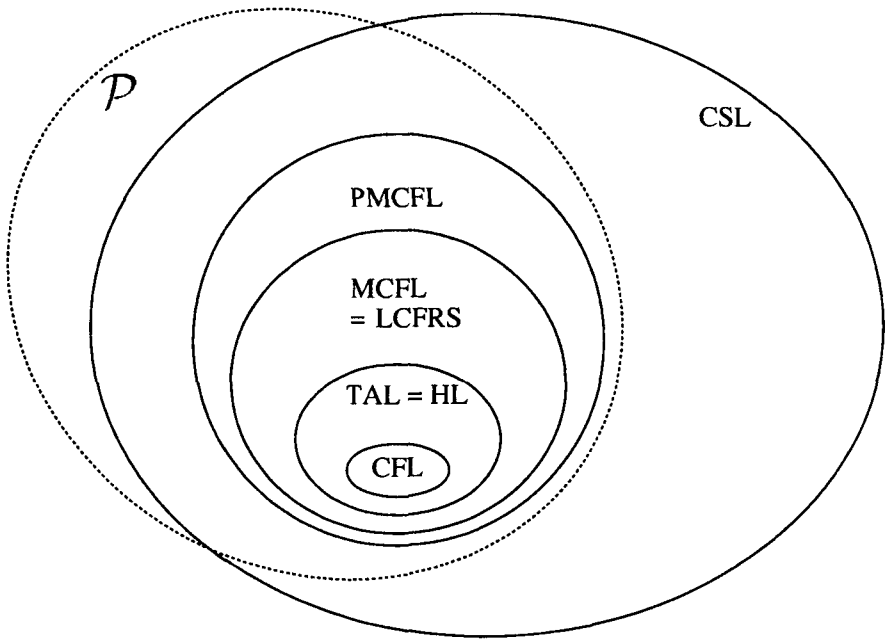
[1] Names of "classes" of languages, grammars, or systems are capitalized in this dissertion.

a subclass of PMCFG, and the class of languages generated by mcfg's is called *multiple context-free languages* (*MCFL*). PMCFG and MCFG can be considered as natural extensions of CFG. For each nonterminal symbol A of a pmcfg, a positive integer $d(A)$ is defined and A derives $d(A)$-tuples of strings. The maximum of $d(A)$ among nonterminals A of a pmcfg G is called the *dimension* of G. A cfg is a special case of mcfg such that its dimension is one. *Linear context-free rewriting systems* (*LCFRS*) introduced by Vijay-Shanker et al.[26] is essentially the same grammatical formalism as MCFG.

As for the classes of languages, it was shown in Ref.[13] that MCFL properly includes CFL, and is properly included in PMCFL, which in turn is properly included in CSL. It was also shown[26] that MCFL properly includes TAL (the class of tree adjoining languages)[11] and HL (the class of head languages)[19]. Moreover, it has been already shown[14] that PMCFL is included in the class $\mathcal{P}$ of computational complexity, i.e., the fixed-language recognition problem for any language generated by a pmcfg is solvable in deterministic polynomial time (Figure 1).

Some subclasses of PMCFG and MCFG can be defined in natural ways. Among those are classification via dimensions, and classification via degrees. A pmcfg (mcfg) with dimension m or less is called an m-pmcfg (m-mcfg). For each $m(\geq 1)$, the class of languages generated by $(m+1)$-pmcfg's ($(m+1)$-mcfg's) properly includes the class of languages generated by m-pmcfg's (m-mcfg's). The degree of a pmcfg G is defined as the maximum of the sizes of production rules of G (see Chapter 2 for formal definition). The class of modified head grammars[22], which were shown[22] to have the same generative power as HG, is a proper subclass of MCFG with dimension 2 and degree 6.

For those newly introduced grammars, including PMCFG and MCFG, their mathematical properties have been extensively studied. But most of those studies focused mainly on properties of the class of "languages" generated by those grammatical formalisms, and properties of "grammars" themselves have not been studied so widely. *Universal recognition problem*, which will be described later, is one of the typical problems for grammars. To clarify computational complexity of the universal recognition problem for a certain class of grammars has great importance if one



The inclusion relation between CSL and $\mathcal{P}$ is a conjecture.
All other inclusion relations are proper.

Figure 1: Inclusion relations among classes of languages (known results only).

aims to use the grammars to describe the syntax of natural languages. The half of this dissertation is devoted to a study on computational complexities of the universal recognition problems for PMCFG, MCFG and their subclasses.

Let $L(G)$ be the language generated by a grammar G. For a class $\mathcal{G}$ of grammars, the *universal recognition problem* for $\mathcal{G}$ is formally defined as follows; take a description of a grammar $G \in \mathcal{G}$ and a string w as an input, decide whether $w \in L(G)$. Note that the size of an input is $|G|+|w|$, where $|G|$ and $|w|$ denote the length of the description of G and that of w, respectively, which is different from the case of fixed-language recognition (parsing) problem. A *fixed-language recognition (parsing) problem* for a language L is defined as follows; take a string w as an input, decide whether $w \in L$. The size of an input is $|w|$ only; any favorable grammar G such that $L = L(G)$ is thought to have "built in" the algorithm. In this case, the size of a grammar G is considered to be a constant and thereby, the succinctness of the grammar does not have effect on the complexity of the fixed-language recognition problem. If one has interest in grammars as representations which explain languages, especially their syntactic structures, then the complexity of a fixed-language recognition problem is not an appropriate measure of syntactical complexity. Fixed-language recognition is a problem for a language, while universal recognition is a problem for a class of grammars. Table 1 summarizes known results on the universal recognition problems for some classes of grammars, where RLFG[18] denotes a subclass of lexical functional grammars and IG denotes the class of indexed grammars[1]. As one can see from the table, the universal recognition problems for many well-known classes of grammars are often intractable. From a viewpoint of computational linguistics, it is significant to find out a class of grammars such that

(i) it has enough generative power to describe the syntax of natural languages, and

(ii) the universal recognition problem for the class is tractable.

In addition, the following property is strongly desired for natural language processing;

(iii) fixed-language recognition problem for any language generated by a grammar in that class is tractable.

To the author's knowledge, no grammatical formalisms $\mathcal{G}$ have been introduced such that the class of languages generated by $\mathcal{G}$ properly includes PMCFL, and is properly included in both of CSL and $\mathcal{P}$, i.e. PMCFG has the strongest generative power among the known classes of grammars which define tractable classes of languages within CSL. Thereby it can be concluded that PMCFG satisfies properties (i) and (iii) above. Hence it is significant to clarify the computational complexities of the universal recognition problems for PMCFG and its subclasses.

In this dissertation, four completeness results on computational complexities of the universal recognition problems for PMCFG, MCFG and their subclasses are shown: The universal recognition problems for PMCFG (MCFG) without any constraint, for PMCFG (MCFG) with nonerasing condition, for m-PMCFG (m-MCFG), and for PMCFG (MCFG) with degree e are EXP-POLY time-complete, PSPACE-complete, $\mathcal{NP}$-complete, and $\mathcal{P}$-complete, respectively (see Figure 2). Characteristics, relations among those subclasses, and their implication in natural language processing, especially language acquisition, are discussed based on these theoretical results.

The other half of this dissertation investigates the gap between CFL and CSL from the viewpoint of complexity theory. It has been known that any language in CFL can be recognized in deterministic polynomial time, while there is an $\mathcal{NP}$-complete language in CSL[21], and hence one may conjecture that there is the border between $\mathcal{P}$ and $\mathcal{NP}$ in the gap between CFL and CSL. A number of computational models have been introduced to clarify computational theoretic hierarchy in this gap. For example, tree automata and their variants, extensions of push-down automata, and finite-state translation systems are widely studied models for this purpose.

Finite state translation system (*FSTS*)[20] was originally introduced as a model of transformational grammars. Later it was found to be an interesting general computational model, and properties of FSTS and its subclasses have been extensively investigated[6, 7, 28]. An fsts consists of a *tree transducer* M and a cfg G[20, 24]. A tree transducer M takes a tree as

Table 1: Known results

grammars	complexity of universal recognition
CFG	$\mathcal{P}$-complete[10]
GPSG[8]	EXP-POLY time-hard[2]
RLFG[18]	$\mathcal{NP}$-hard[2]
IG[1]	EXP-POLY time-complete[23]
CSG	PSPACE-complete[12]

Table 2: The complexities of the universal recognition problems

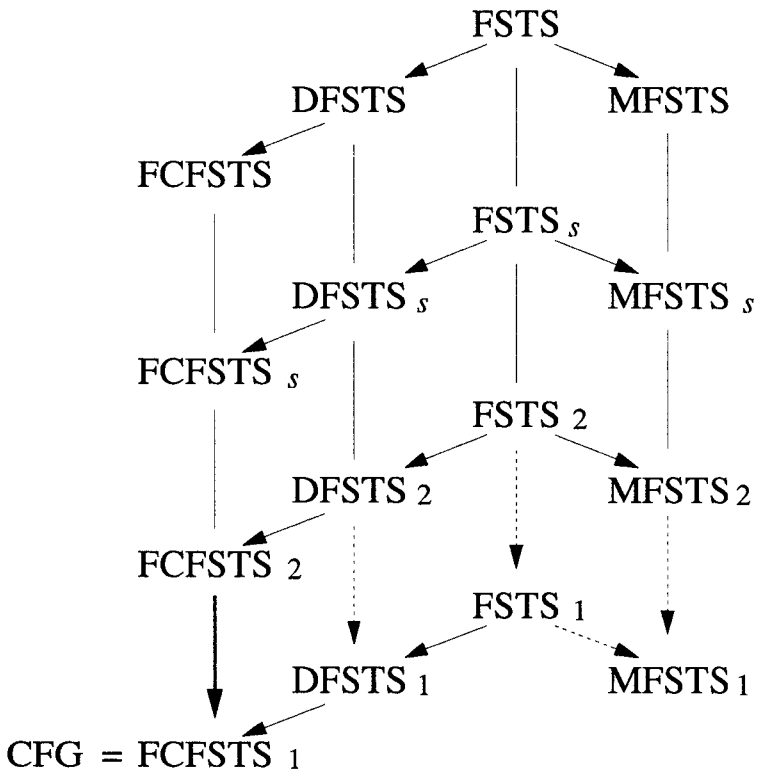
	PMCFG	MCFG
(with no constraint)	EXP-POLY time-complete	
with non-erasing condition	PSPACE-complete	
m- — (bounded dimension)	$\mathcal{NP}$-complete for $m \geq 1$	for $m \geq 2$
with degree e	$\mathcal{P}$-complete for $e \geq 3$ solvable in $O(\|G\|^2\|w\|^{e+1})$-time	solvable in $O(\|G\|^2\|w\|^e)$-time

an input, starts from the initial state with its head scanning the root node of an input. According to the current state and the label of the scanned node, M transforms an input tree into an output tree in a top-down way. An fsts (M, G) is a tree transducer M with its input domain being the set of derivation trees of the cfg G[20, 24]. The output set of trees is called the *tree language generated by* (M, G), and the *yield language generated by* (M, G) is defined to be the set of strings obtained by concatenating (the labels of) leaves of a tree in the tree language.

A number of studies have been devoted to the generative powers of FSTS and its subclasses. Engelfriet et al. summarized these results in Ref.[7]. It has been shown that the generative power of *deterministic FSTS* is properly stronger than that of *finite-copying* FSTS, and is properly weaker than that of (nondeterministic) FSTS. *Monadic* FSTS (ET0L system in Ref.[7]) is another subclass discussed in Ref.[7], and it has been shown to have properly weaker generative power than that of nondeterministic FSTS. In Ref.[7], concept of *state-bound* of fsts is introduced, and a hierarchy of generative power via state-bound is investigated. Figure 2 summarizes relations between the generative power of subclasses of FSTS, where FSTS_s, DFSTS_s, and MFSTS_s denote FSTS with state-bound s, deterministic FSTS with state-bound s, and monadic FSTS with state-bound s, respectively, and FCFSTS_s denotes finite-copying FSTS with *copying-bound* s.

In Ref.[28], it is shown that the class of yield languages generated by finite-copying FSTS equals to the class of LCFRS, hence MCFL. In Chapter 4 of this dissertation, it is shown that the class of yield languages generated by deterministic FSTS equals to PMCFL. It is also shown that there is an $\mathcal{NP}$-complete language in the class of yield languages generated by nondeterministic monadic FSTS with state-bound 2. See Figure 3 and compare it with Figure 1.

By our results, a number of known properties of PMCFL and MCFL will be used for the study of FSTS and their yield languages, and vice versa. In fact, as a corollary of our results, it can be concluded that the (fixed-language) recognition problem for the class of yield languages generated by deterministic FSTS is solvable in $O(n^{e+1})$-time, where n is the length of an input word and e is a constant called the degree of the



$A \longrightarrow B$: the generative power of A is properly stronger than B .

$A \dashrightarrow B$: the generative power of A is stronger than B .

Figure 2: Generative power hierarchy of subclasses of FSTS[7].

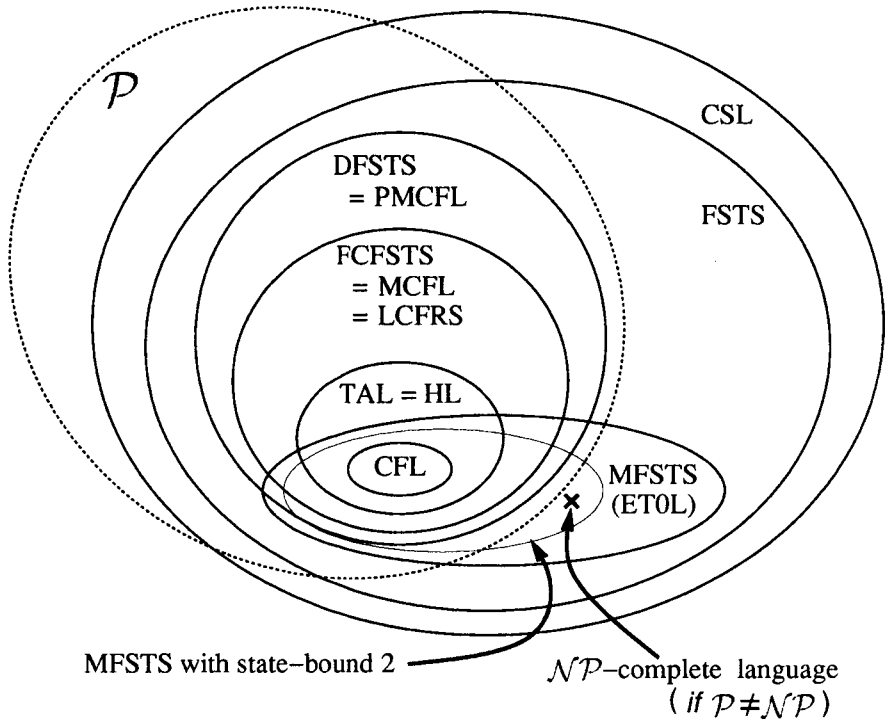


Figure 3: Inclusion relations among classes of languages (known results and results in this dissertation).

deterministic fsts.

2 Definitions

2.1 Parallel Multiple Context-Free Grammars

A *parallel multiple context-free grammar* (*pmcfg*) is defined to be a 5-tuple $G = (V_N, V_T, F, P, S)$ which satisfies the following conditions (G1) through (G5)[14, 22].

(G1) V_N is a finite set of *nonterminal symbols*. For each nonterminal $A \in V_N$, a positive integer $d(A)$ is associated, and called the *dimension of* A. The *dimension of the grammar* G is defined as $\max\{d(A) \mid A \in V_N\}$.

(G2) V_T is a finite set of *terminal symbols* which is disjoint with V_N. For a positive integer d, the set of all the d-tuples of strings over V_T is denoted by $(V_T^*)^d$.

(G3) F is a finite set of *functions* satisfying the following conditions. For each $f \in F$, positive integers $a(f)$, $d_i(f)$ $(1 \leq i \leq a(f))$ and $r(f)$ are given, and f is a total function from $(V_T^*)^{d_1(f)} \times (V_T^*)^{d_2(f)} \times \cdots \times (V_T^*)^{d_{a(f)}(f)}$ to $(V_T^*)^{r(f)}$ which satisfies the following condition (f1). Let

$$\bar{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id_i(f)})$$

denote the ith argument of f for $1 \leq i \leq a(f)$, and let

$$X = \{x_{ij} \mid 1 \leq i \leq a(f), 1 \leq j \leq d_i(f)\}. \tag{1}$$

(f1) For $1 \leq h \leq r(f)$, the hth component of f, denoted by $f^{[h]}$ is defined by a concatenation of some terminal strings in V_T^* and some variables in X. That is, a nonnegative integer $v_h(f)$ is defined and

$$f^{[h]}[\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_{a(f)}] = \alpha_{h0} z_{h1} \alpha_{h1} z_{h2} \cdots z_{hv_h(f)} \alpha_{hv_h(f)}, \tag{2}$$

where $\alpha_{hk} \in V_T^*$ $(0 \leq k \leq v_h(f))$ and $z_{hk} \in X$ $(1 \leq k \leq v_h(f))$.

(G4) P is a finite set of *productions* of the form $A \rightarrow f[A_1, A_2, \ldots, A_{a(f)}]$ where $A, A_1, A_2, \ldots, A_{a(f)} \in V_N$, $f \in F$, $r(f) = d(A)$ and $d_i(f) = d(A_i)$ $(1 \leq i \leq a(f))$. If $a(f) = 0$, then f has no argument and f

equals to a tuple of strings over V_T. A production with a function f such that $a(f) = 0$ is called a *terminating production*, otherwise it is called a *nonterminating production*. A terminating production $A \rightarrow f$ with $f = \bar{\alpha} \in (V_T^*)^{r(f)}$ may be denoted by $A \rightarrow \bar{\alpha}$.

(G5) $S \in V_N$ is the *initial symbol*, and $d(S) = 1$.

The class of pmcfg's is denoted by *PMCFG*.

The language generated by a pmcfg $G = (V_N, V_T, F, P, S)$ is defined as follows. For $A \in V_N$, define $L_G(A) \subseteq (V_T^*)^{d(A)}$ as the smallest set satisfying the following two conditions:

(L1) If there is a terminating production $A \rightarrow f$ and $f = \bar{\alpha} \in (V_T^*)^{d(A)}$, then $\bar{\alpha} \in L_G(A)$.

(L2) If $A \rightarrow f[A_1, A_2, \ldots, A_{a(f)}] \in P$ and $\bar{\alpha}_i \in L_G(A_i)$ $(1 \leq i \leq a(f))$, then $\bar{\alpha} = f[\bar{\alpha}_1, \bar{\alpha}_2, \ldots, \bar{\alpha}_{a(f)}] \in L_G(A)$. We say that $A \rightarrow f[A_1, A_2, \ldots, A_{a(f)}]$ is *the last production applied to obtain* $\bar{\alpha}$.

Let $L(G) \triangleq L_G(S)$. (Note that $d(S) = 1$ by (G5) of the definition, hence $L(G)$ is a set of strings.) $L(G)$ is called the *parallel multiple context-free language (pmcfl) generated by* G. The class of pmcfl's is denoted by *PMCFL*.

If all the functions of a pmcfg G satisfy the following condition (f2), then G is called a *multiple context-free grammar (mcfg)*, and the language generated by G is called the *multiple context-free language generated by* G. The class of mcfg's and the class of mcfl's are denoted by *MCFG* and *MCFL*, respectively.

(f2) For each variable x_{ij} in X, the total number of occurrences of x_{ij} in the right-hand sides of (2) from $h = 1$ through $r(f)$ is at most one.

If some variable occurs in the right-hand side of (2) more than once, or occurs in the right-hand sides of (2) for different h's, the string substituted for the variable will be copied more than once. It has been shown that such copy operations increase the generative power of grammars[13] i.e. MCFL $\subsetneq$ PMCFL. Condition (f2) inhibits these copy operations.

Example 2.1: Let $G_1 = (V_N, V_T, F, P, S)$ where $V_N = \{A, B, S\}(d(A) = d(B) = 2, d(S) = 1)$, $V_T = \{a, b, c, d\}$, $F = \{f, g_A, g_B, h\}$ and the rules in P are

$$\begin{array}{lll} r_0: & S \to f[A, B] & \text{where} \quad f[(x_{11}, x_{12}), (x_{21}, x_{22})] = x_{11}x_{21}x_{12}x_{22} \\ r_1: & A \to g_A[A] & \text{where} \quad g_A[(x_1, x_2)] = (ax_1, cx_2) \\ r_2: & B \to g_B[B] & \text{where} \quad g_B[(x_1, x_2)] = (bx_1, dx_2) \\ r_3: & A \to h & \text{where} \quad h = (\varepsilon, \varepsilon) \\ r_4: & B \to h. & \end{array}$$

Note that G_1 is an mcfg since no variable occurs more than once in the right-hand side of the functions. The language generated by G_1 is defined as follows: By rule r_3, $(\varepsilon, \varepsilon)$ belongs to $L_G(A)$. Substituting $(\varepsilon, \varepsilon)$ for A in the right-hand side of r_1, we obtain $(a, c) \in L_G(A)$. Repeating application of r_1, $(a^m, c^m) \in L_G(A)$ for $m \geq 0$. In a similar way, $(b^n, d^n) \in L_G(B)$ for $n \geq 0$. By rule r_0, $L_G(S) = \{a^m b^n c^m d^n | m, n \geq 0\}$ and this is the language generated by G_1. □

Example 2.2: Let $G_2 = (V_N, V_T, F, P, S), V_N = \{S\}, V_T = \{a\}, F = \{f, f_a\}$ and the rules in P are

$$\begin{array}{lll} r_0: & S \to f[S] & \text{where} \quad f[x] = xx \\ r_1: & S \to f_a & \text{where} \quad f_a = a. \end{array}$$

G_2 is a pmcfg but is not an mcfg since the function f does not satisfy the condition (f2). The language generated by G_2 is $\{a^{2^n} | n \geq 0\}$, which cannot be generated by any mcfg (see Lemma 6 of [14]). □

Hereafter, we will define subclasses of PMCFG and MCFG. If all the functions f of a pmcfg (resp. mcfg) G satisfy the following condition (f3), then G is a *pmcfg with non-erasing condition* (resp. *mcfg with non-erasing condition*). The class of pmcfg's (mcfg's) with non-erasing condition is denoted by *NEPMCFG* (*NEMCFG*).

(f3) Let X be defined as in (1). Each variable $x \in X$ appears at least once in the right-hand side of (2) for some h $(1 \leq h \leq r(f))$. □

In a mcfg with non-erasing condition, each variable appears exactly once in the right-hand side of (2) for some h. This is a same formalism to a

subclass of linear context-free rewriting systems[26], a subclass which deals with only tuples of strings. From a grammatical viewpoint, NEPMCFG (NEMCFG) is a proper subclass of PMCFG (MCFG). But it has been already shown, as in the following lemma, that condition (f3) does not weaken the generative power of PMCFG (MCFG)[22].

Lemma 2.1[22]: For a given pmcfg (resp. mcfg), we can construct a weakly equivalent pmcfg (resp. mcfg) with non-erasing condition.

Sketch of Proof: The idea behind the construction is similar to that of ε-rule elimination procedure of a context-free grammar. For example, assume that there is a production $A \rightarrow f[B_1, \ldots, B_n]$ and x_{ij} does not appear in the right-hand side of (2). Then a new nonterminal B_i' with $d(B_i') = d(B_i) - 1$ is introduced, and this production is replaced by $A \rightarrow f'[B_1, \ldots, B_i', \ldots, B_n]$ where f' is identical to f except that the dimension of the ith argument is smaller by one than f. Furthermore, for each production whose left-hand side is B_i, add a new production whose left-hand side is B_i' and whose function in the right-hand side is defined by deleting jth component of the original one. See Lemma 1 of Ref.[14] for the formal proof. □

Next, pmcfg's (mcfg's) with bounded dimension is introduced. For a positive integer m, if the dimension of a pmcfg G is not greater than m, then G is called an *m-pmcfg*. The class of m-pmcfg's is denoted by m-PMCFG. In other words, m-PMCFG is a subclass of PMCFG such that the dimension of each grammar in the class is equal to or smaller than the previously given constant m. Note that m is treated as a constant in m-PMCFG. For a pmcfg G in the class of (unconstrained) PMCFG, there is a number m which happens to be the dimension of G. But in general, m is in $O(|G|)$ and we cannot treat it as a constant in this case. For $m \geq 1$, m-PMCFG is a proper subclass of $m+1$-PMCFG, and the generative power of the former is properly weaker than the latter. In a similar way, m-MCFG is defined.

The last subclass we introduce is the class of pmcfg's with bounded degree. (Refer to the expression (2) of (f1)). The *degree* of a function f is defined as $\sum_{h=1}^{r(f)}(v_h(f)+1)$, which equals to the total number of variables

appearing in the right-hand side of f plus the dimension of f. If the maximum degree among the functions in F of G is e, then G is called a *pmcfg with degree e*. In the same way, an *mcfg with degree e* is defined. The class of pmcfg's (mcfg's) with degree e is denoted by PMCFG_e (MCFG_e). Note that in these classes of grammars, a degree e is treated as a constant. For $e \geq 1$, PMCFG_e is a proper subclass of PMCFG_{e+1}, but the relation between the generative power of them is in general not known.

We note that if a degree of a grammars is bounded, then a dimension of the grammars is also bounded, but not vice-versa. Indeed, PMCFG_e is a subclass of e-PMCFG and hence the dimension of each grammar in that class can be treated as a constant. But in m-PMCFG, there is a grammar with an arbitrary large degree and hence it cannot be treated as a constant in the class.

The mcfg G_1 introduced in Example 2.1 has dimension 2 and degree 5 (rule r_0 has the maximum degree) and the pmcfg G_2 in Example 2.2 has dimension 1 and degree 3.

The *size* of a pmcfg $G = (V_N, V_T, F, P, S)$ is defined as follows. The size of a function $f \in F$, denoted by $|f|$, is defined as the sum of the lengths of the right-hand sides of (2), that is,

$$|f| \triangleq \sum_{h=1}^{r(f)} (v_h(f) + \sum_{k=0}^{v_h(f)} |\alpha_{hk}|).$$

The size of a rule $r : A \rightarrow f[A_1, A_2, \ldots, A_{a(f)}]$, denoted by $|r|$, is defined to be $a(f) + 2$ (each of $A, A_1, A_2, \ldots, A_{a(f)}$ and f counts for one). Define the size of G, denoted by $|G|$, to be the sum of $|V_N|, |V_T|, \sum_{f \in F} |f|$ and $\sum_{r \in P} |r|$.

Lastly, a derivation is defined. Let $G = (V_N, V_T, F, P, S)$ be a pmcfg. For a nonterminal symbol $A \in V_N$ and k ($1 \leq k \leq d(A)$), the kth component of A is represented by a symbol $(A^{[k]}, v)$, where v is an index to distinguish distinct "instances" of the same nonterminal symbol A in a derivation. Define $\mathrm{C}(V_N) \triangleq \{(A^{[k]}, v) \mid A \in V_N, 1 \leq k \leq d(A) \text{ and } v \geq 0\}$. Each $(A^{[k]}, v) \in \mathrm{C}(V_N)$ is called a *component symbol* of A and v is called the *index* of $(A^{[k]}, v)$. Let (A, v) denote $((A^{[1]}, v), (A^{[2]}, v), \ldots, (A^{[d(A)]}, v)) \in \mathrm{C}(V_N)^{d(A)}$ for $A \in V_N$ and $v \geq 0$.

Assume that $\bar{\alpha}$ is a tuple of strings over $V_T \cup \mathrm{C}(V_N)$ and there exists $v \geq 0$ and $A \in V_N$ such that $(A^{[k]}, v)$ appears at least once in $\bar{\alpha}$ for some k $(1 \leq k \leq d(A))$. Let $r : A \rightarrow f[A_1, A_2, \ldots, A_{a(f)}]$ be a production in P. Let $\bar{\beta}$ denote the tuples of strings obtained from $\bar{\alpha}$ by replacing each $(A^{[h]}, v)$ $(1 \leq h \leq a(f))$ (if exists in $\bar{\alpha}$) with $f^{[h]}[(A_1, v_1), (A_2, v_2), \ldots, (A_{a(f)}, v_{a(f)})]$ where v_i's $(1 \leq i \leq a(f))$ are distinct nonnegative integers such that no $(B^{[j]}, v_i)$ does not appear in $\bar{\alpha}$ for any $B \in V_N$ and $j \geq 0$, that is, v_i's are newly introduced indices which are not used in $\bar{\alpha}$. Then, α *directly derives* β (by *applying* production r to$(A^{[h]}, v)$'s), denoted by $\bar{\alpha} \underset{G}{\Rightarrow} \bar{\beta}$.

Let $\overset{*}{\underset{G}{\Rightarrow}}$ denote the reflexive transitive closure of $\underset{G}{\Rightarrow}$. If $\bar{\alpha} \overset{*}{\underset{G}{\Rightarrow}} \bar{\beta}$, then $\bar{\alpha}$ is said to *derive* $\bar{\beta}$. If G is understood by context, $\underset{G}{\Rightarrow}$ and $\overset{*}{\underset{G}{\Rightarrow}}$ are written as $\Rightarrow$ and $\overset{*}{\Rightarrow}$, respectively.

It is easily shown that, for each $A \in V_N$, $\bar{\alpha} \in (V_T^*)^{d(A)}$ and $v \geq 0$,

$$\bar{\alpha} \in L_G(A) \textit{ iff } (A, v) \overset{*}{\underset{G}{\Rightarrow}} \bar{\alpha}.$$

Example 2.3: Consider mcfg G_1 defined in Example 2.1. Then,

$$\begin{aligned}
(S^{[1]}, v) &\Rightarrow (A^{[1]}, v_0)(B^{[1]}, v_1)(A^{[2]}, v_0)(B^{[2]}, v_1) \\
&\Rightarrow a(A^{[1]}, v_2)(B^{[1]}, v_1)c(A^{[2]}, v_2)(B^{[2]}, v_1) \\
&\quad \cdots \\
&\Rightarrow a^m(A^{[1]}, v_{m+1})(B^{[1]}, v_1)c^m(A^{[2]}, v_{m+1})(B^{[2]}, v_1) \\
&\Rightarrow a^m(B^{[1]}, v_1)c^m(B^{[2]}, v_1) \\
&\Rightarrow a^m b(B^{[1]}, v_{m+2})c^m d(B^{[2]}, v_{m+2}) \\
&\quad \cdots \\
&\Rightarrow a^m b^n(B^{[1]}, v_{m+n+1})c^m d^n(B^{[2]}, v_{m+n+1}) \\
&\Rightarrow a^m b^n c^m d^n
\end{aligned}$$

for any m, n (≥ 0) and $v, v_0, \ldots, v_{m+n+1}$ (≥ 0). Since the indices are redundant for this derivation, the derivations are written without indices for simplicity. For example, the above derivation is written as

$$\begin{aligned}
S^{[1]} &\Rightarrow A^{[1]}B^{[1]}A^{[2]}B^{[2]} \\
&\Rightarrow aA^{[1]}B^{[1]}cA^{[2]}B^{[2]} \\
&\quad \cdots
\end{aligned}$$

$$
\begin{aligned}
&\Rightarrow\ a^m A^{[1]} B^{[1]} c^m A^{[2]} B^{[2]} \\
&\Rightarrow\ a^m B^{[1]} c^m B^{[2]} \\
&\Rightarrow\ a^m b B^{[1]} c^m d B^{[2]} \\
&\quad \cdots \\
&\Rightarrow\ a^m b^n B^{[1]} c^m d^n B^{[2]} \\
&\Rightarrow\ a^m b^n c^m d^n .
\end{aligned}
$$

□

2.2 Finite State Translation Systems

A set Σ of symbols is a *ranked alphabet* if, for each $\sigma \in \Sigma$, a unique non-negative number $\rho(\sigma)$ which is called the *rank* of σ is associated. Define $\mathcal{T}_\Sigma$ as the smallest set such that;

- If $\rho(\sigma) = 0$ for $\sigma \in \Sigma$, then $\sigma \in \mathcal{T}_\Sigma$.
- If $\rho(\sigma) = n\ (\geq 1)$ for $\sigma \in \Sigma$ and $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$, then $t = \sigma(t_1, \ldots, t_n) \in \mathcal{T}_\Sigma$. σ is called the *root symbol*, or shortly, the *root* of t.

Hereafter, a term in $\mathcal{T}_\Sigma$ may be called a *tree*.

Let $G = (V_N, V_T, P, S)$ be a context-free grammar (cfg) where V_N, V_T, P and S are a set of *nonterminal symbols*, a set of *terminal symbols*, a set of *productions* and the *initial symbol*, respectively. A *derivation tree of the cfg* G is a term defined as follows.

(T1) For every $a \in V_T$, a is a derivation tree of G.

(T2) Assume that there are a production $r : A \rightarrow X_1 \cdots X_n$ $(A \in V_N, X_1, \ldots, X_n \in V_N \cup V_T)$ in P where r is the label of this production, and n derivation trees $t_1, \ldots t_n$ whose roots are labeled with $r_1, \ldots, r_n$, respectively, and

- if $X_i \in V_N$, then r_i $(1 \leq i \leq n)$ is the label of a production $r_i : X_i \rightarrow \cdots$, whose left-hand side is X_i, and
- if $X_i \in V_T$, then $r_i = t_i = X_i$.

Then $r(t_1, \ldots, t_n)$ is a derivation tree of G.

(T3) There are no other derivation trees.

Let $\mathcal{R}(G)$ be the set of derivation trees whose root is the label of a production of which left-hand side is the initial symbol S. Remark that if we take $\Sigma = \{\text{the labels of productions in } P\} \cup V_T$, and define $\rho(r) = n$ for $r : A \rightarrow X_1 \cdots X_n \in P$ and $\rho(a) = 0$ for $a \in V_T$, then Σ is a ranked alphabet and $\mathcal{R}(G) \subseteq \mathcal{T}_\Sigma$.

A *tree transducer* is defined in Ref.[20] as a generalization of a generalized sequential machine, and it defines a mapping from trees to trees. But in this paper, since we are mainly interested in a string language generated by it, a "tree-to-string" version of transducer defined in Ref.[7] is reviewed. For sets Q and X, let

$$Q[X] \triangleq \{q[x] \mid q \in Q, x \in X\}.$$

A *tree-to-string transducer* (*yT-transducer* or simply *transducer*) is defined to be a 5-tuple $M = (Q, \Sigma, \Delta, q_0, R)$ where

- Q is a finite set of *states*,
- Σ is an *input ranked alphabet*,
- Δ is an *output alphabet*,
- $q_0 \in Q$ is the *initial state*, and
- R is a set of *rules* of the form

$$q[\sigma(x_1, \ldots, x_n)] \rightarrow v$$

where $q \in Q, \sigma \in \Sigma, \rho(\sigma) = n$ and $v \in (\Delta \cup Q[\{x_1, \ldots, x_n\}])^*$.

If different rules in R have different left-hand sides, then M is called *deterministic*[7].

A *configuration* of a yT-transducer is an element in $(\Delta \cup Q[\mathcal{T}_\Sigma])^*$. *Derivation* of M is defined as follows. Let $c = \alpha_1 q[\sigma(t_1, \ldots, t_n)]\alpha_2$ be a configuration where $\alpha_1, \alpha_2 \in (\Delta \cup Q[\mathcal{T}_\Sigma])^*$, $q \in Q$, $\sigma \in \Sigma$, $\rho(\sigma) = n$ and $t_1, \ldots, t_n \in \mathcal{T}_\Sigma$. Assume that there is a rule $q[\sigma(x_1, \ldots, x_n)] \rightarrow v$ in R,

and v' can be obtained from v by substituting $t_1, \ldots, t_n$ for $x_1, \ldots, x_n$, respectively, then $c \Rightarrow \alpha_1 v' \alpha_2$. Let $\overset{*}{\Rightarrow}$ be reflexive and transitive closure of $\Rightarrow$. For configurations c and c', if $c \overset{*}{\Rightarrow} c'$, then c *derives* c'. If there is no $c' \in \Delta^*$ such that $c \overset{*}{\Rightarrow} c'$, then c *derives no output*. For example, if there is no rule whose left-hand side is $q[\sigma(x_1, \ldots, x_n)]$, then $c = \alpha_1 q[\sigma(t_1, \ldots, t_n)]\alpha_2$ derives no output.

Example 2.4[20]: Let $M = (Q, \Sigma, \Delta, q_d, R)$ be a yT-transducer where

$$\begin{aligned} &Q = \{q_d, q_i\} \\ &\Sigma = \{c, y, +, \cdot\} \qquad (\rho(c) = \rho(y) = 0, \rho(+) = \rho(\cdot) = 2) \\ &\Delta = \Sigma \cup \{0, 1\} \end{aligned}$$

and the rules in R are:

$$\begin{aligned} q_i[c] &\to c & q_i[y] &\to y \\ q_i[+(x_1, x_2)] &\to q_i[x_1] + q_i[x_2] \\ q_i[\cdot(x_1, x_2)] &\to q_i[x_1] \cdot q_i[x_2] \\ q_d[c] &\to 0 & q_d[y] &\to 1 \\ q_d[+(x_1, x_2)] &\to q_d[x_1] + q_d[x_2] \\ q_d[\cdot(x_1, x_2)] &\to q_d[x_1] \cdot q_i[x_2] + q_i[x_1] \cdot q_d[x_2]. \end{aligned}$$

Intuitively, an element in $\mathcal{T}_\Sigma$ represents an arithmetic expression, and state q_d and q_i represent "differential" and "identity", respectively. Let $t = q_d[\cdot(y, +(c, y))]$ and $t' = q_d[y] \cdot q_i[+(c, y)] + q_i[y] \cdot q_d[+(c, y)]$, then $t \Rightarrow t'$, which corresponds to $\frac{d}{dy}(y \cdot (c + y)) = \frac{d}{dy} y \cdot (c + y) + y \cdot \frac{d}{dy}(c + y)$. □

A *tree-to-string finite state translation system* (*yT-fsts*, or *fsts* for short) is defined by a yT-transducer M and a cfg G, written as (M, G) [2]. The class of fsts' is denoted by $FSTS$.

Define $yL(M, G)$, called the *yield language generated by a yT-fsts* (M, G), as

$$yL(M, G) \triangleq \{t \in \Delta^* \mid \exists t' \in \mathcal{R}(G), q_0[t'] \overset{*}{\Rightarrow} t\}$$

[2]In Ref.[20], a yT-fsts is defined by a yT-transducer and a *recognizable set of trees*. In Ref.[24], it is shown that the class of recognizable sets of trees is equal to the class of sets of derivation trees of CFG. Hence a yT-fsts is defined by a yT-transducer and a cfg in this paper.

where Δ is an output alphabet and q_0 is the initial state of M. Note that $\mathcal{R}(G)$ is a set of derivation trees of the cfg G and hence recognizable set of trees. An fsts is called *deterministic*[7] if the transducer M is deterministic. The class of deterministic fsts' is denoted by *DFSTS*. We use a terminology "*nondeterministic*" when we emphasize that we don't assume determinism of the transducer.

Next, a *state-bound* of fsts and *finite-copying FSTS*[7] are defined. Let (M, G) be an fsts with an output alphabet Δ and an initial state q_0. Let $t \in \mathcal{R}(G)$ and consider a derivation $\alpha : q_0[t] \stackrel{*}{\Rightarrow} w \in \Delta^*$. Let t' be a subtree of t. Now, delete from the original derivation α all the derivation steps which operates on t'. This leads to the following new derivation which keeps t' untouched:

$$\alpha' : q_0[t] \stackrel{*}{\Rightarrow} w_1 q_{i_1}[t'] w_2 \cdots w_n q_{i_n}[t'] w_{n+1}$$

where $w_i \in \Delta^*$ $(1 \leq i \leq n+1)$.

The *state sequence of* t' *in derivation* α is defined to be $\langle q_{i_1}, \ldots, q_{i_n} \rangle$. The derivation α has a *state-bound* s if, for each subtree of t, the number of different states in the state sequence is at most s. α has a *copying-bound* k if, for each subtree of t, the length of its state sequence is at most k. An fsts (M, G) has a *state-bound* s if for each $w \in yL(M, G)$, there is a derivation tree $t \in \mathcal{R}(G)$ such that the derivation $q_0[t] \stackrel{*}{\Rightarrow} w$ has a state-bound s. An fsts (M, G) is a *finite-copying fsts* if there is a constant k such that for each $w \in yL(M, G)$, there is a derivation tree $t \in \mathcal{R}(G)$ such that the derivation $q_0[t] \stackrel{*}{\Rightarrow} w$ has a copying-bound k. The class of finite-copying fsts' is denoted by *FCFSTS*.

An fsts (M, G) whose second component G is a regular grammar is called an ET0L system (see Ref.[7]). In this paper, we say a *monadic fsts* for an ET0L system. The class of monadic fsts' is denoted by *MFSTS*.

Figure 2 shows relationship among the generative power of subclasses of FSTS. In the figure, FSTS_s, DFSTS_s and MFSTS_s denote the classes of each fsts' with state-bound s, respectively. For FCFSTS, the subscript denotes its copying-bound. An arrow from a class A to another class B means that A has properly stronger power than B.

3 Universal Recognition Problems

3.1 General Case

In this section, the universal recognition problems for PMCFG and for MCFG are both shown to be EXP-POLY time-complete, where

$$\text{EXP-POLY time} \triangleq \{L \mid L \text{ is solvable in deterministic } O(c^{p(n)}) \text{ time for some } c > 1 \text{ and some polynomial } p, \text{ where } n \text{ is the size of an input}\}.$$

Since MCFG are a subclass of PMCFG, it suffices to show that the problem for PMCFG belongs to EXP-POLY time, and that the problem for MCFG is EXP-POLY time-hard.

3.1.1 Containment in EXP-POLY Time

In this section,, the universal recognition problem for PMCFG is shown to belong to EXP-POLY time. An algorithm which solves the problem for MCFG are presented first, and it is extended for PMCFG.

First, a table NULL$(A, (k_1, k_2, \ldots, k_r))$ $(A \in V_N, 1 \leq k_1 < k_2 < \cdots < k_r \leq d(A))$ is computed to satisfy

- NULL$(A, (k_1, \ldots, k_r)) = 1$ if there is some $(w_1, \ldots, w_{d(A)}) \in L_G(A)$ such that $w_{k_1} = w_{k_2} = \cdots = w_{k_r} = \varepsilon$, and
- NULL$(A, (k_1, \ldots, k_r)) = 0$ otherwise.

If NULL$(A, (k_1, k_2, \ldots, k_r)) = 1$ then $(k_1, k_2, \ldots, k_r)$ is called a *nullable combination for* A.

A simple way to check whether G generates w or not is to simulate derivation of w on a working tape nondeterministically. However, such a method may require a working tape of size exponential to $|G|$ since there may exist a string w such that, in every derivation $(S^{[1]}, 0) \stackrel{*}{\Rightarrow} w$, a length of a string on the tape once grows exponential to $|G|$ due to component symbols $(A^{[k_1]}, v), (A^{[k_2]}, v), \ldots$, and $(A^{[k_r]}, v)$ such that $(k_1, k_2, \ldots, k_r)$ is a nullable combination for A. If the table NULL is precomputed, it can be decided whether $w \in L(G)$ or not with a polynomial bounded

working tape by referring the table NULL and deleting directly arbitrary component symbols $(A^{[k_1]}, v), (A^{[k_2]}, v), \ldots$,and $(A^{[k_r]}, v)$ such that $(k_1, k_2, \ldots, k_r)$ is a nullable combination for A.

Lemma 3.1: The table NULL can be constructed in EXP-POLY time.

Proof. The table can be constructed as follows:

Step 1. If a terminating production $A \to (\alpha_1, \ldots, \alpha_{d(A)})$ is in P, then for every tuple $(k_1, \ldots, k_r)$ such that $1 \leq k_1 < \cdots < k_r \leq d(A)$ and $\alpha_{k_j} = \varepsilon$ $(1 \leq j \leq r)$, set the value of NULL$(A, (k_1, \ldots, k_r))$ to be 1. The others are set to 0.

Step 2. If a nonterminating production $A \to f[A_1, \ldots, A_{a(f)}]$ is in P and NULL $(A_i, (k_{i1}, \ldots, k_{ir_i})) = 1$ for $1 \leq i \leq a(f)$, then let $\bar{w} = (w_1, \ldots, w_{d(A)})$ be the tuple obtained by replacing x_{ij} $(1 \leq i \leq a(f), j \in \{k_{i1}, \ldots, k_{ir_i}\})$ with ε in the right-hand side of the definition of f (see (2) in section 2.1). For every tuple $(k'_1, \ldots, k'_r)$ such that $1 \leq k'_1 < \cdots < k'_r \leq d(A)$ and $w_{k'_1} = \cdots, w_{k'_r} = \varepsilon$, set the value of NULL$(A, (k'_1, \ldots, k'_r))$ to be 1. This step is applied to all productions and all entries of the current table simultaneously.

Step 3. Repeat Step 2 until the table is not changed.

We assume that the read/write operation for a single entry of the table NULL, and the evaluation of the value of $f \in F$ for given arguments can be perfomed as elementary operations. Let m be the dimension of G. Since the number of the terminating productions is at most $|G|$ and the number of the subsets of $\{1, 2, \ldots, d(A)\}$ is at most $2^m \in O(2^{|G|})$, Step 1 takes $O(|G|2^{|G|})$ time. Consider Step 2. The number of the nonterminating productions and that of the nonterminal symbols in the right-hand side of each production are both $O(|G|)$. For each A_i $(1 \leq i \leq a(f))$, there are at most $2^m \in O(2^{|G|})$ entries $(A_i, (\cdots))$ whose values are equal to 1, and for a single value $\bar{w}$ of f in Step 2, the number of entries to be set to 1 is at most $O(2^{|G|})$. As a whole, Step 2 takes $O(|G| \times (2^{|G|})^{|G|} \times 2^{|G|}) = O(|G|2^{|G|^2+|G|})$ time. Finally, this table has $|V_N|2^m$ entries, and hence Step 2 loops at most $|V_N|2^m$ times. That is, the table can be constructed by $O(|V_N|2^m \times |G|2^{|G|^2+|G|}) = O(c^{|G|^2})$

elementary operations for some $c > 1$. It follows that the table can be constructed in EXP-POLY time in $|G|$. □

Next, a nondeterministic algorithm is presented which decides whether $w \in L(G)$ for an mcfg G and string w with the table NULL and a polynomial space bounded working tape.

Algorithm 3.1:

input : an mcfg $G = (V_N, V_T, F, P, S)$ with the table NULL and a string w.

Try to generate w by simulating a derivation from the initial symbol nondeterministically on a working tape as follows.

Step 1. Write the component symbol $(S^{[1]}, 0)$ of the initial symbol S on the working tape.

Step 2. Execute one of the following (a) and (b).

(a) Choose a rewriting production $A \rightarrow f[A_1, A_2, \ldots, A_{a(f)}]$ nondeterministically, and apply it to component symbols $(A^{[h]}, v)$'s $(1 \leq h \leq d(A))$ on the tape. Remark that, since G does not necessary satisfy the non-erasing condition, some newly introduced component symbols $(A_i^{[k]}, v_i)$ might be lost to the tape. If the length of the string on the tape exceeds $m|w| + |f_{\max}|$, where $|f_{\max}|$ denotes the maximum size of the function in F, then halt.

(b) Choose component symbols $(B^{[k_1]}, v)$, $(B^{[k_2]}, v)$, ..., $(B^{[k_r]}, v)$ nondeterministically, with $(B^{[j]}, v)$'s being lost to the tape for each j $(1 \leq j \leq d(B)$ and $j \neq k_i$ for any i $(1 \leq i \leq r))$. If $(k_1, k_2, \ldots, k_r)$ is a nullable combination for B, then erase the component symbols and shift the other symbols to fill the blanks.

Step 3. Repeat Step 2 until there exists no component symbol on the tape.

Step 4. If the string on the tape equals to w, then accept w. □

Next, it is shown that

Algorithm 3.1 accepts w *iff* $w \in L(G)$.

The *only if* part is obvious and *if* part will be shown. It suffices to show that, for $w \in L(G)$, Algorithm 3.1 can generate w by using only $m|w| + |f_{\max}|$ symbols on the tape. First, following proposition is shown.

Proposition 3.2: Assume that $(S^{[1]}, 0) \stackrel{*}{\Rightarrow} \alpha \stackrel{*}{\Rightarrow} w$ ($\alpha \in (V_T \cup \mathrm{C}(V_N))^*, w \in V_T^*$) and $|\alpha| \leq m|w| + |f_{\max}|$. Also assume that, after executing Step 2 finite times, Algorithm 3.1 reaches a state such that α is on the working tape. Then, there exists a sequence of moves from this state to the accepting state such that the number of symbols on the working tape is always $m|w| + |f_{\max}|$ or less.

Proof. The lemma is shown by the induction on the length of the derivation $\alpha \stackrel{*}{\Rightarrow} w$. For the basis, the lemma holds clearly since $\alpha = w$ and $|w| \leq |w| + |f_{\max}|$.

Suppose that the lemma holds for all derivations $\beta \stackrel{*}{\Rightarrow} w$ of length τ ($\tau \geq 0$) or less, and consider a derivation $\alpha \stackrel{*}{\Rightarrow} w$ of length $\tau + 1$.

First, assume $|\alpha| \leq m|w|$. As $\alpha \stackrel{*}{\Rightarrow} w$ is of length 1 or more, there is some production $r : A \rightarrow f[A_1, A_2, \ldots, A_{a(f)}]$ and $\alpha' \in (V_T \cup \mathrm{C}(V_N))^*$ such that $\alpha \Rightarrow \alpha'$ by the application of r and $\alpha' \stackrel{*}{\Rightarrow} w$. Since the size of the right-hand side of this production is at most $|f_{\max}|$, $|\alpha'| \leq m|w| + |f_{\max}|$. By executing (a) of Step 2 and applying r, α' is generated on the tape. As $\alpha' \stackrel{*}{\Rightarrow} w$ is of length τ, by the inductive hypothesis, there exists a sequence of moves to the accepting state such that the number of symbols on the working tape is always $m|w| + |f_{\max}|$ or less.

Next, assume $m|w| < |\alpha| \leq m|w| + |f_{\max}|$ and let i_t be the number of terminal symbols and i_n be the number of distinct indices of component symbols (not the number of component symbols) appearing in α. Observe that $i_n > |w| - i_t$ since $m|w| < |\alpha| \leq i_t + mi_n$. Since i_t terminal symbols out of $|w|$ have been generated, the other $|w| - i_t$ terminal symbols must be generated from i_n nonterminal symbols. Hence, there exists at least $i_n - (|w| - i_t) > 0$ distinct indices v's such that every $(B_{i_v}^{[k]}, v)$ in α derives ε. Let u be any of such indices and $\alpha'' \in (V_T \cup \mathrm{C}(V_N))^*$ be the string obtained from α by replacing all $(B_{i_u}^{[k']}, u)$'s in α with ε. It can be

easily shown that there is a derivation $\alpha \stackrel{*}{\Rightarrow} \alpha'' \stackrel{*}{\Rightarrow} w$ of length $\tau + 1$ (apply productions to erase $(B_{i_u}^{[k']}, u)$'s first). By executing (b) of Step 2, α'' can be obtained from α on the tape and $|\alpha''| < |\alpha| \leq m|w| + |f_{\max}|$. As $\alpha'' \stackrel{*}{\Rightarrow} w$ is of length τ or less, by the inductive hypothesis, from this state there exists a sequence of moves to the accepting state such that the number of symbols on the working tape is always $m|w| + |f_{\max}|$ or less. Hence, the proposition holds. □

Assume that $w \in L(G)$. Then $(S^{[1]}, 0) \stackrel{*}{\Rightarrow} w$. At the first time Step 2 is executed, there is only one symbol $(S^{[1]}, 0)$ on the tape. Hence, letting $w = \alpha$, Proposition 3.2 implies that Algorithm 3.1 generates w and accepts it by using at most $m|w| + |f_{\max}|$ symbols at a time on the tape.

Next, it must be considered how the symbol should be represented upon the tape. Let $n = |G| + |w|$ be the size of an input. As the number of distinct terminal symbols is at most n, they can be represented in $O(\log n)$ size per one symbol. For each component symbol $(A^{[k]}, v)$ $(1 \leq k \leq d(A), v \geq 0)$, information on $(1)A$, $(2)k$, and $(3)v$ is kept on the tape. (1) and (2) can be represented in $O(\log n)$ space, and $(3)v$ can be represented in $O(\log(m|w| + |f_{\max}|))$ space since at most $m|w| + |f_{\max}|$ symbols are on the tape simultaneously by Proposition 3.2. Hence, one component symbol can be represented in $O(\log n + \log n + \log(m|w| + |f_{\max}|))$ space. As $m, |w|$ and $|f_{\max}|$ are all $O(n)$, $O(\log n + \log n + \log(m|w| + |f_{\max}|)) = O(\log n)$. The number of the component symbols appearing on the tape is $O(m|w| + |f_{\max}|) = O(n^2)$. Therefore the total size needed upon the tape is $O(n^2 \log n)$.

The following lemma can be obtained from Lemma 3.1 and the analysis of Algorithm 3.1 described above.

Lemma 3.3: The universal recognition problem for MCFG belongs to EXP-POLY time.

Proof. Given an mcfg G and a string w, the table NULL can be constructed in EXP-POLY time by Lemma 3.1. Algorithm 3.1 decides whether $w \in L(G)$ by using w, G and the table NULL (on the read-only input tape) and an $O(n^2 \log n)$ bounded working tape where $n = |G| + |w|$. Since the size of NULL is $O(n2^n)$, the size of the input tape is $O(n2^n)$. It can be easily shown in a similar way to the proof of Savitch's

theorem[9] that there exists a deterministic Turing machine M_1 which decides whether $w \in L(G)$ with the same read-only input tape as Algorithm 3.1 and an $O(n^4 \log^2 n)$ bounded working tape. From M_1, a nondeterministic Turing machine M_2 which accepts $L(G)$ within $O(d^{n^4 \log n^2})$ time can be constructed in a similar way to the proof of Theorem 12.10(b) in Ref.[9]. □

Next, Algorithm 3.1 is extended for PMCFG. For PMCFG there may exist a pmcfg G and a string w such that in every derivation of w, the number of occurrences of an identical component symbol $(A^{[h]}, v)$ on a tape grows exponentially to $|w|$ by copy operations, and lastly, $(A^{[h]}, v)$ derives ε. If above Algorithm 3.1 is extended for PMCFG in a straightforward way, the size of a working tape needed can not be bounded by any polynomial.

To extend the algorithm for PMCFG, a special treatment is needed for the component symbols which are copied, and derive ε at last. Let $(A^{[h]}, v)$ be such a component symbol. Note that the number of the occurrences of $(A^{[h]}, v)$ makes no influence on the string to be generated on the tape since $(A^{[h]}, v)$ will derive ε.

By using this property, a derivation can be simulated as follows. First, choose a component symbol $(A^{[h]}, v)$ nondeterministically. Intuitively, the chosen symbol $(A^{[h]}, v)$ is "guessed" to derive ε. Mark one $(A^{[h]}, v)$ and erase the other occurrences of $(A^{[h]}, v)$ (if there is on the tape). All the symbols derived from the marked symbol will be also marked. If a terminal string is derived from a marked symbol, which contradicts the "guess", then reject the input and halt. If all marked symbol derive ε, then it turns out that the nondeterministic choice was correct and the terminal string generated on the tape can be derived by G.

The following Algorithm 3.2 is obtained by modifying Algorithm 3.1 in such a way that the number of component symbols can not be greater than $m + |w|$ times as many as the number of the distinct indices.

Algorithm 3.2:

input : a pmcfg $G = (V_N, V_T, F, P, S)$ with the table NULL and a string w.

Step 1. Write the component symbol $(S^{[1]}, 0)$ of the initial symbol S on the working tape (and do not mark it).

Step 2. Execute one of the following (a),(b) and (c).

(a) Choose a rewriting production $A \rightarrow f[A_1, A_2, \ldots, A_{a(f)}]$ nondeterministically, and apply it to component symbols $(A^{[h]}, v)$'s $(1 \leq h \leq d(A))$ on the tape. If $(A^{[h]}, v)$ $(1 \leq h \leq d(A))$ is marked, then mark to all the symbols derived from $(A^{[h]}, v)$. If a terminal symbol is derived from marked component, or the length of the string on the tape exceeds $(m + |w|)|w| + |f_{\max}|$, where $|f_{\max}|$ denotes the maximum size of the function in F, then halt.

(b) Choose component symbols $(B^{[k_1]}, v)$, $(B^{[k_2]}, v)$, $\ldots$, $(B^{[k_r]}, v)$ nondeterministically, with $(B^{[j]}, v)$'s being lost to the tape for each j $(1 \leq j \leq d(B)$ and $j \neq k_i$ for any i $(1 \leq i \leq r))$. If $(k_1, k_2, \ldots, k_r)$ is a nullable combination for B, then erase the component symbols and shift the other symbols to fill the blanks.

(c) Choose a component symbol and mark it. If there are other occurrences of the symbol, then erase them.

Step 3. Repeat Step 2 until no component symbol remains on the tape.

Step 4. If the string on the tape equals to w, then accept w. □

The next proposition claims that, for $w \in L(G)$, Algorithm 3.2 can generate w by using only $(m + |w|)|w| + |f_{\max}|$ symbols on the tape.

Proposition 3.4: Assume that $(S^{[1]}, 0) \stackrel{*}{\Rightarrow} \alpha \stackrel{*}{\Rightarrow} w$ $(\alpha \in (V_T \cup \mathrm{C}(V_N))^*, w \in V_T^*)$ and $|\alpha| \leq m|w| + |f_{\max}|$. Also assume that, after executing Step 2 finite times, Algorithm 3.2 reaches a state such that α is on the working tape. Then, there exists a sequence of moves from this state to the accepting state such that the number of symbols on the working tape is always $(m + |w|)|w| + |f_{\max}|$ or less.

Proof. This is a PMCFG version of Proposition 3.2, and the proof is similar to that of Proposition 3.2. The difference between them are

- $m|w|$ in Proposition 3.2 is replaced by $(m + |w|)|w|$, and
- discussion for the case $(m + |w|)|w| < |\alpha| \leq (m + |w|)|w| + |f_{\max}|$ in the inductive step is replaced by follows.

Assume that $(m + |w|)|w| < |\alpha| \leq (m + |w|)|w| + |f_{\max}|$ and let i_t be the number of terminal symbols and i_n be the number of distinct indices of component symbols (not the number of component symbols) appearing in α. Let $(A_1, v_1), \ldots, (A_{i_n}, v_{i_n})$ be nonterminal symbols on the tape and let w_i $(1 \leq i \leq i_n)$ be the tuple of strings derived from (A_i, v_i) by G. Without loss of generality, the number of unmarked component symbols of (A_i, v_i) is not greater than $|w_i|$ (otherwise, execute (c) of Step 2), and the number of marked component symbols of (A_i, v_i) is not greater than m. Observe that $i_n > |w| - i_t$ holds since $(m + |w|)|w| < |\alpha| \leq i_t + \sum_{i=1}^{i_n}(m + |w_i|) = i_t + mi_n + |w|$ and $|w|^2 \geq |w|$. The proof proceeds as in the proof for MCFG. □

By the above lemma and Savitch's theorem, the following lemma can be shown in a same way to Lemma 3.3.

Lemma 3.5: The universal recognition problem for PMCFG belongs to EXP-POLY time. □

3.1.2 Basic Ideas

First, a basic algorithm is presented which translates a given polynomial space-bounded Turing machine M into an mcfg G such that every valid computation of M can be simulated by a derivation of G. In the following sections 3.1.3 and 3.2, this algorithm is modified to show that the universal recognition problems for MCFG and for NEMCFG is EXP-POLY time-hard and PSPACE-hard, respectively.

Let $M = (Q, \Sigma, \Gamma, B, \delta, q_S, Q_F)$ be a Turing machine where

Q is the finite set of states,
$\Sigma \subset \Gamma$ is the finite set of input symbols,
Γ is the finite set of tape symbols,
$B \in \Gamma - \Sigma$ is the blank symbol,
$\delta : (Q \times \Gamma) \to 2^{(Q \times \Gamma \times \{L,R\})}$ is the transition function,
$q_S \in Q$ is the initial state, and
$Q_F \subseteq Q$ is the finite set of final state.

An *ID* of M is a triple (q, k, α), where $q \in Q, \alpha \in \Gamma^*$ and k is a positive integer such that $1 \leq k \leq |\alpha|$ which denotes the position of the head on the tape. Let "$\vdash$" denote one step transition between IDs of M. Define ACC(M) as the smallest subset of IDs of M satisfying the following conditions (a) and (b).

(a) If $q \in Q_F$ then $(q, k, \alpha) \in \mathrm{ACC}(M)$ for every $\alpha \in \Gamma^*$ and k $(1 \leq k \leq |\alpha|)$.

(b) If $I \vdash I'$ and $I' \in \mathrm{ACC}(M)$, then $I \in \mathrm{ACC}(M)$.

It is obvious that, for $t \in \Sigma^*$

$$M \text{ accepts } t \textit{ iff } (q_S, 1, t) \in \mathrm{ACC}(M). \tag{3}$$

Fix a polynomial p and a $p(n)$ space-bounded Turing machine $M = (Q, \Sigma, \Gamma, B, \delta, q_S, Q_F)$. By using the following Algorithm 3.3, the problem whether M accepts a given string $t \in \Sigma^*$ can be reduced to the universal recognition problem for MCFG. First, the idea behind the reduction is explained. For convenience, number the symbols in Γ as $c_1, \ldots, c_{|\Gamma|}$, and define a pairing function as $\langle k, c_j \rangle = (k-1)|\Gamma| + j$ $(1 \leq k \leq p(n), 1 \leq j \leq |\Gamma|)$.

For each $q \in Q$ and integer k such that $1 \leq k \leq p(n)$, a nonterminal symbol A_{qk} with $d(A_{qk}) = p(n)|\Gamma|$ is introduced. Let

$$\begin{aligned}
ID_1 &= (q, \quad k, \quad\; b_1 b_2 \cdots b_{k-1} c b_{k+1} \cdots b_{p(n)}) \\
ID_2 &= (q', \quad k+1, \quad b_1 b_2 \cdots b_{k-1} c' b_{k+1} \cdots b_{p(n)}) \\
ID_3 &= (q'', \quad k-1, \quad b_1 b_2 \cdots b_{k-1} c'' b_{k+1} \cdots b_{p(n)})
\end{aligned}$$

and let α_1, α_2 and α_3 be the sequences of component symbols defined as

$$\begin{aligned}
\alpha_1 &= A_{qk}^{[\langle 1,b_1\rangle]} A_{qk}^{[\langle 2,b_2\rangle]} \cdots A_{qk}^{[\langle k-1,b_{k-1}\rangle]} A_{qk}^{[\langle k,c\rangle]} A_{qk}^{[\langle k+1,b_{k+1}\rangle]} \cdots A_{qk}^{[\langle p(n),b_{p(n)}\rangle]} \\
\alpha_2 &= A_{q'k+1}^{[\langle 1,b_1\rangle]} A_{q'k+1}^{[\langle 2,b_2\rangle]} \cdots A_{q'k+1}^{[\langle k-1,b_{k-1}\rangle]} A_{q'k+1}^{[\langle k,c'\rangle]} A_{q'k+1}^{[\langle k+1,b_{k+1}\rangle]} \cdots A_{q'k+1}^{[\langle p(n),b_{p(n)}\rangle]} \\
\alpha_3 &= A_{q''k-1}^{[\langle 1,b_1\rangle]} A_{q''k-1}^{[\langle 2,b_2\rangle]} \cdots A_{q''k-1}^{[\langle k-1,b_{k-1}\rangle]} A_{q''k-1}^{[\langle k,c''\rangle]} A_{q''k-1}^{[\langle k+1,b_{k+1}\rangle]} \cdots A_{q''k-1}^{[\langle p(n),b_{p(n)}\rangle]}.
\end{aligned}$$

Note that the component symbols are written without indices using abbreviation mentioned in Example 2.3. Intuitively saying, ID_1, ID_2 and ID_3 correspond to α_1, α_2 and α_3, respectively. The lower suffix of a nonterminal symbol A indicates the state in Q and the position of the head on the tape. The sequence of upper suffixes of component symbols represents the string on the tape.

Assume that $\delta(q,c) = \{(q',c',R),(q'',c'',L)\}$. Observe that the following (M1) and (M2) hold by the definition of ACC(M).

(M1) If $q \in Q_F$, then $ID_1 \in \mathrm{ACC}(M)$.

(M2) If $q \notin Q_F$, then

$$ID_1 \in \mathrm{ACC}(M) \textit{ iff } \text{either } ID_2 \in \mathrm{ACC}(M) \text{ or } ID_3 \in \mathrm{ACC}(M).$$

Productions of mcfg G are constructed to satisfy the following conditions (P1) through (P3).

(P1) If $q \in Q_F$ then $\alpha_1 \Rightarrow \varepsilon$.

(P2a) If $(q',c',R) \in \delta(q,c)$, that is, if $ID_1 \vdash ID_2$, then $\alpha_1 \Rightarrow \alpha_2$. See Figure 4.

(P2b) If $(q'',c'',L) \in \delta(q,c)$, that is, if $ID_1 \vdash ID_3$, then $\alpha_1 \Rightarrow \alpha_3$.

(P3) If $\alpha \Rightarrow \alpha'$ other than those in (P1), (P2a) and (P2b), then the symbol 1 appears at least once in α'.

If productions are constructed to satisfy (P1) through (P3) for every $q \in Q$ and $c \in \Gamma$, it can be shown that

$$ID_1 \in \mathrm{ACC}(M) \textit{ iff } \alpha_1 \overset{*}{\Rightarrow} \varepsilon.$$

Algorithm 3.3:

input : a string $t = t_1 t_2 \cdots t_n$ over Σ.

output : mcfg $G = (V_N, V_T, F, P, S)$ and string w such that M accepts t *iff* $w \in L(G)$.

For convenience, let $t = t_1 t_2 \cdots t_n BB \cdots B \in \Gamma^{p(n)}$. The mcfg G and string w are constructed as follows.

Step 1. Let $V_T = \{1\}$ and $w = \varepsilon$.

Step 2. Let $V_N = \{S\} \cup \{A_{qk} \mid q \in Q, 1 \leq k \leq p(n)\}$ where $d(A_{qk}) = p(n)|\Gamma|$ $(q \in Q, 1 \leq k \leq p(n))$.

Step 3. Add

$$f[\bar{x}] = x_{\langle 1,t_1\rangle} x_{\langle 2,t_2\rangle} \cdots x_{\langle n,t_n\rangle} x_{\langle n+1,B\rangle} \cdots x_{\langle p(n),B\rangle} \tag{4}$$

to F where $\bar{x} = (x_1, x_2, \ldots, x_{p(n)|\Gamma|})$, and add $S \to f[A_{q_S 1}]$ to P. The right-hand side of this production corresponds to the start ID $(q_S, 1, t_1 t_2 \cdots t_n B \cdots B)$.

Step 4. For each $q \in Q_F$, add terminating production $A_{qk} \to (\varepsilon, \ldots, \varepsilon)$ to P for each k $(1 \leq k \leq p(n))$.

Note that these productions realize the condition (P1) before this algorithm.

Step 5. If $(q', c', R) \in \delta(q, c)$, then add $f_{cc'k}$ to F and $A_{qk} \to f_{cc'k}[A_{q'k+1}]$ to P for each k $(1 \leq k < p(n))$, where $f_{cc'k}$ is defined as

$$f_{cc'k}^{\langle r,b\rangle}[\bar{x}] = x_{\langle r,b\rangle} \qquad (r \neq k) \tag{5}$$

$$f_{cc'k}^{\langle k,c\rangle}[\bar{x}] = x_{\langle k,c'\rangle} \tag{6}$$

$$f_{cc'k}^{\langle k,b\rangle}[\bar{x}] = 1 \qquad (b \neq c). \tag{7}$$

See Figure 4 and note that the conditions (P2a) and (P3) are realized by these productions.

Step 6. If $(q'', c'', L) \in \delta(q, c)$, then add $g_{cc''k}$ to F and $A_{qk} \to g_{cc''k}[A_{q''k-1}]$ to P for each k $(1 < k \leq p(n))$, where $g_{cc''k}$ is defined as

$$g_{cc''k}^{\langle r,b\rangle}[\bar{x}] = x_{\langle r,b\rangle} \qquad (r \neq k)$$

$$g_{cc''k}^{\langle k,c\rangle}[\bar{x}] = x_{\langle k,c''\rangle}$$

$$g_{cc''k}^{\langle k,b\rangle}[\bar{x}] = 1 \qquad (b \neq c).$$

These productions realize the conditions (P2b) and (P3) before this algorithm. □

The time complexity of Algorithm 3.3 is analyzed as follows. Remind that the Turing machine M is fixed and $|Q|, |\Sigma|, |\Gamma|, |Q_F|$ and the number of values of δ are considered to be constant. Step 1 of Algorithm 3.3 takes $O(1)$ time. In Step 2, the number of nonterminal symbols to be defined is $O(p(n))$, which implies that $O(\log p(n))$ space is required to denote a single nonterminal symbol, and Step 2 takes $O(p(n) \log p(n))$ time. Steps 3 and 4 take $O(p(n) \log p(n))$ time and $O(p(n)^2)$ time, respectively. For Step 5, $O(p(n) \log p(n))$ time is required to construct a single production and $O(p(n))$ productions are constructed, and therefore this step takes $O(p(n)^2 \log p(n))$ time. Similarly, Step 6 takes $O(p(n)^2 \log p(n))$ time. As a whole, Algorithm 3.3 can be executed in deterministic $O(p(n)^2 \log p(n))$ time.

For the end of this section, following theorem is presented. Refer to the Theorem 3.7 and Lemma 3.12 for the proof of this theorem.

Theorem 3.6: In Algorithm 3.3, M accepts t *iff* $w \in L(G)$. □

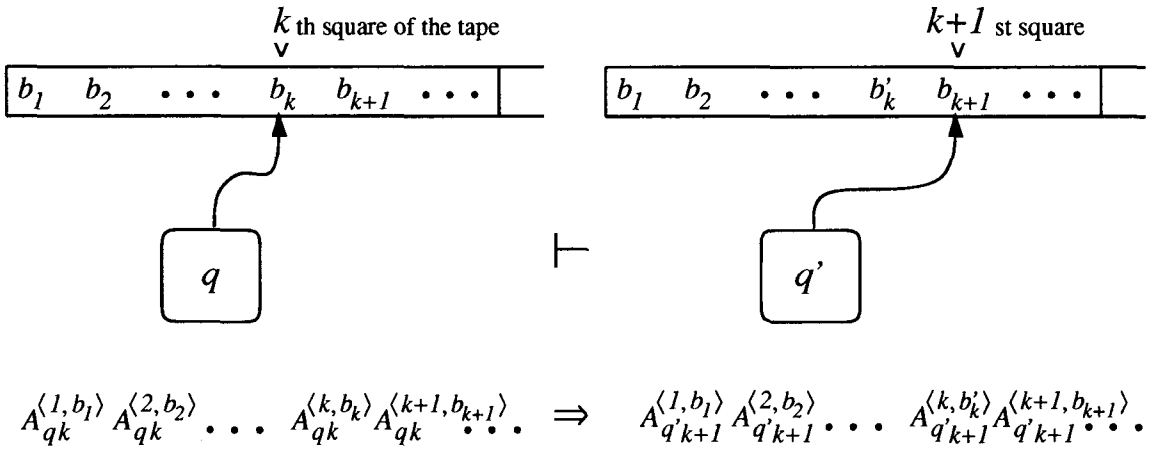


Figure 5: Simulation of a move of Turing machine M

3.1.3 EXP-POLY time-hardness

Next, it is shown that the universal recognition problem for MCFG is EXP-POLY time-hard. It is known that L belongs to EXP-POLY time if and only if L is accepted by a polynomial space-bounded alternating Turing machine (ATM)[3]. Let $M \triangleq (Q, \Sigma, \Gamma, B, \delta, q_S, Q_F, Q_U, Q_E)$ be an ATM where

Q is the finite set of states,
$\Sigma \subseteq \Gamma$ is the finite set of input symbols,
Γ is the finite set of tape symbols,
$B \in \Gamma - \Sigma$ is the blank symbol,
$\delta : (Q \times \Gamma) \times 2^{(Q \times \Gamma \times \{L,R\})}$ is the transition function,
$q_S \in Q$ is the initial state,
$Q_F \subseteq Q$ is the set of final state,
$Q_U \subseteq Q$ is the set of universal states, and
$Q_E \subseteq Q$ is the set of existential states,

where the followings are assumed:

- Q_F, Q_U and Q_E are disjoint, and
- $Q = Q_F \cup Q_U \cup Q_E$.

An *ID* (q, k, α) and a relation "$\vdash$" are defined in a same way as those of Turing machine. Define ACC(M) as the smallest subset of IDs of M satisfying the following conditions (a) through (c).

(a) If $q \in Q_F$ then $(q, k, \alpha) \in$ ACC(M) for every $\alpha \in \Gamma^*$ and k $(1 \leq k \leq |\alpha|)$.

(b) Let $I = (q, k, \alpha)$ be an ID with $q \in Q_U$. If every I' satisfying $I \vdash I'$ belongs to ACC(M), then I also belongs to ACC(M).

(c) Let $I = (q, k, \alpha)$ be an ID with $q \in Q_E$. If some I'' satisfying $I \vdash I''$ belongs to ACC(M), then I also belongs to ACC(M).

M *accepts* a string t if and only if $(q_S, 1, t) \in$ ACC(M).

Fix a polynomial p and $p(n)$ space-bounded ATM M. The problem whether M accepts a given string t can be reduced to the universal recognition problem for MCFG in deterministic polynomial time by the following Algorithm 3.4.

First, the idea behind the reduction is explained. Remind the basic algorithm 3.3 described in section 3.1.2. For convenience, number the symbols in Γ as $c_1, \ldots, c_{|\Gamma|}$, and define a pairing function as $\langle k, c_j \rangle = (k-1)|\Gamma| + j$ $(1 \leq k \leq p(n), 1 \leq j \leq |\Gamma|)$. For each $q \in Q$ and integer k such that $1 \leq k \leq p(n)$, a nonterminal symbol A_{qk} with $d(A_{qk}) = p(n)|\Gamma|$ is introduced. Let

$$\begin{aligned} ID_1 &= (q, \quad k, \quad b_1 b_2 \cdots b_{k-1} c b_{k+1} \cdots b_{p(n)}) \\ ID_2 &= (q', \quad k+1, \quad b_1 b_2 \cdots b_{k-1} c' b_{k+1} \cdots b_{p(n)}) \\ ID_3 &= (q'', \quad k-1, \quad b_1 b_2 \cdots b_{k-1} c'' b_{k+1} \cdots b_{p(n)}), \end{aligned}$$

and let

$$\begin{aligned} \alpha_1 &= A_{qk}^{[\langle 1,b_1 \rangle]} A_{qk}^{[\langle 2,b_2 \rangle]} \cdots A_{qk}^{[\langle k-1,b_{k-1} \rangle]} A_{qk}^{[\langle k,c \rangle]} A_{qk}^{[\langle k+1,b_{k+1} \rangle]} \cdots A_{qk}^{[\langle p(n),b_{p(n)} \rangle]} \\ \alpha_2 &= A_{q'k+1}^{[\langle 1,b_1 \rangle]} A_{q'k+1}^{[\langle 2,b_2 \rangle]} \cdots A_{q'k+1}^{[\langle k-1,b_{k-1} \rangle]} A_{q'k+1}^{[\langle k,c' \rangle]} A_{q'k+1}^{[\langle k+1,b_{k+1} \rangle]} \cdots A_{q'k+1}^{[\langle p(n),b_{p(n)} \rangle]} \\ \alpha_3 &= A_{q''k-1}^{[\langle 1,b_1 \rangle]} A_{q''k-1}^{[\langle 2,b_2 \rangle]} \cdots A_{q''k-1}^{[\langle k-1,b_{k-1} \rangle]} A_{q''k-1}^{[\langle k,c'' \rangle]} A_{q''k-1}^{[\langle k+1,b_{k+1} \rangle]} \cdots A_{q''k-1}^{[\langle p(n),b_{p(n)} \rangle]} \end{aligned}$$

as in section 3.1.2.

Assume that $\delta(q,c) = \{(q', c', R), (q'', c'', L)\}$. Observe that the following (M3) through (M5) hold by the definition of ACC(M).

(M3) If $q \in Q_F$, then $ID_1 \in \mathrm{ACC}(M)$.

(M4) If $q \in Q_E$, then

$$ID_1 \in \mathrm{ACC}(M) \textit{ iff } \text{either } ID_2 \in \mathrm{ACC}(M) \text{ or } ID_3 \in \mathrm{ACC}(M).$$

(M5) If $q \in Q_U$, then

$$ID_1 \in \mathrm{ACC}(M) \textit{ iff } \text{both } ID_2 \in \mathrm{ACC}(M) \text{ and } ID_3 \in \mathrm{ACC}(M).$$

Note that (M5) is newly introduced proposition for ATM. To realize (M5), productions of mcfg G are constructed to satisfy the following conditions (P4) through (P7).

(P4) If $q \in Q_F$, then $\alpha_1 \underset{G}{\Rightarrow} \varepsilon$.

(P5) If $q \in Q_E$, then both $\alpha_1 \underset{G}{\Rightarrow} \alpha_2$ and $\alpha_1 \underset{G}{\Rightarrow} \alpha_3$.

(P6) If $q \in Q_U$, then

$$\alpha_1 \underset{G}{\Rightarrow} A^{[\langle 1,b_1\rangle]}_{q'k+1} A^{[\langle 1,b_1\rangle]}_{q''k-1} \cdots A^{[\langle k-1,b_{k-1}\rangle]}_{q'k+1} A^{[\langle k-1,b_{k-1}\rangle]}_{q''k-1} A^{[\langle k,c'\rangle]}_{q'k+1} A^{[\langle k,c''\rangle]}_{q''k-1}$$
$$A^{[\langle k+1,b_{k+1}\rangle]}_{q'k+1} A^{[\langle k+1,b_{k+1}\rangle]}_{q''k-1} \cdots A^{[\langle p(n),b_{p(n)}\rangle]}_{q'k+1} A^{[\langle p(n),b_{p(n)}\rangle]}_{q''k-1}.$$

Notice that the right-hand side is the sequence obtained by concatenating α_2 and α_3 componentwise.

(P7) For each derivation $\alpha_1 \underset{G}{\Rightarrow} \alpha'$ other than those in (P4) through (P6), the symbol 1 appears at least once in α'.

Observe that the following (Q1) holds by (P5) and (P7). Similarly, (Q2) holds by (P6) and (P7).

(Q1) If $q \in Q_E$, then $\alpha_1 \overset{*}{\underset{G}{\Rightarrow}} \varepsilon$ *iff* $\alpha_2 \overset{*}{\underset{G}{\Rightarrow}} \varepsilon$ or $\alpha_3 \overset{*}{\underset{G}{\Rightarrow}} \varepsilon$.

(Q2) If $q \in Q_U$, then $\alpha_1 \overset{*}{\underset{G}{\Rightarrow}} \varepsilon$ *iff* $\alpha_2 \overset{*}{\underset{G}{\Rightarrow}} \varepsilon$ and $\alpha_3 \overset{*}{\underset{G}{\Rightarrow}} \varepsilon$.

If productions are constructed to satisfy (P4) through (P7) for every $q \in Q$ and $c \in \Gamma$, it can be shown that

$$ID_1 \in \mathrm{ACC}(M) \textit{ iff } \alpha_1 \overset{*}{\underset{G}{\Rightarrow}} \varepsilon.$$

by (M3) through (M5),(P4),(Q1) and (Q2).

Algorithm 3.4:

input : a string $t = t_1 t_2 \cdots t_n$ over Σ.
output : mcfg $G = (V_N, V_T, F, P, S)$ and string w such that M accepts t *iff* $w \in L(G)$.

For convenience, let $t = t_1 t_2 \cdots t_n BB \cdots B \in \Gamma^{p(n)}$. The mcfg G and string w are constructed as follows.

Step 1. Let $V_T = \{1\}$ and $w = \varepsilon$.

Step 2. Let $V_N = \{S\} \cup \{A_{qk} \mid q \in Q, 1 \leq k \leq p(n)\}$ where $d(A_{qk}) = p(n)|\Gamma|$ $(q \in Q, 1 \leq k \leq p(n))$.

Step 3. Add

$$f[\bar{x}] = x_{\langle 1,t_1\rangle} x_{\langle 2,t_2\rangle} \cdots x_{\langle n,t_n\rangle} x_{\langle n+1,B\rangle} \cdots x_{\langle p(n),B\rangle}$$

to F where $\bar{x} = (x_1, x_2, \ldots, x_{p(n)|\Gamma|})$, and add $S \to f[A_{q_S 1}]$ to P. The right-hand side of the production corresponds to the start ID $(q_S, 1, t_1 \cdots t_n B \cdots B)$.

Step 4. For each $q \in Q_F$, add terminating production $A_{qk} \to (\varepsilon, \ldots, \varepsilon)$ to P for each k $(1 \leq k \leq p(n))$. Remind the condition (P4) before this algorithm.

Step 5. If

$$\begin{aligned} \delta(q,c) &= \{(q_i', c_i', R) \mid 1 \leq i \leq n_R\} \\ &\quad \cup \{(q_j'', c_j'', L) \mid 1 \leq j \leq n_L\}, \end{aligned} \tag{8}$$

then for $(q_i', c_i', R) \in \delta(q,c)$, function $f_{cc_i'k}$ is defined for each k $(1 \leq k < p(n))$ as follows.

$$f_{cc_i'k}^{\langle r,b\rangle}[\bar{x}] = x_{\langle r,b\rangle} \qquad (r \neq k) \tag{9}$$

$$f_{cc_i'k}^{\langle k,c\rangle}[\bar{x}] = x_{\langle k,c_i'\rangle} \tag{10}$$

$$f_{cc_i'k}^{\langle k,b\rangle}[\bar{x}] = 1 \qquad (b \neq c). \tag{11}$$

Similarly, for $(q_j'', c_j'', L) \in \delta(q,c)$, function $g_{cc_j''k}$ is defined for each k $(1 < k \leq p(n))$ as follows.

$$g_{cc_j''k}^{\langle r,b\rangle}[\bar{x}] = x_{\langle r,b\rangle} \qquad (r \neq k) \tag{12}$$

$$g_{cc_j''k}^{\langle k,c\rangle}[\bar{x}] = x_{\langle k,c_j''\rangle} \tag{13}$$

$$g_{cc_j''k}^{\langle k,b\rangle}[\bar{x}] = 1 \qquad (b \neq c). \tag{14}$$

Suppose that $q \in Q_E$. Then by (c) in the definition of ACC(M), the ID

$$(q, k, b_1 b_2 \cdots b_{k-1} c b_{k+1} \cdots b_{p(n)}) \tag{15}$$

belongs to ACC(M) if one of the following IDs belongs to ACC(M):

$$(q_i', k+1, b_1 b_2 \cdots b_{k-1} c_i' b_{k+1} \cdots b_{p(n)}) \qquad (1 \leq i \leq n_R); \tag{16}$$

$$(q_j'', k-1, b_1 b_2 \cdots b_{k-1} c_j'' b_{k+1} \cdots b_{p(n)}) \qquad (1 \leq j \leq n_L). \tag{17}$$

In this case, add $f_{cc_i'k}$ to F and add

$$A_{qk} \to f_{cc_i'k}[A_{q_i'k+1}] \tag{18}$$

to P for $1 \leq i \leq n_R$ and $1 \leq k < p(n)$. Also add $g_{cc_j''k}$ to F and add

$$A_{qk} \to g_{cc_j''k}[A_{q_j''k-1}] \tag{19}$$

to P for $1 \leq j \leq n_L$ and $1 < k \leq p(n)$.

Note that the conditions (P5) and (P7) before this algorithm are realized by these productions.

Suppose that $q \in Q_U$. Then by (b) in the definition of ACC(M), the ID (15) belongs to ACC(M) if all of (16) and (17) belong to ACC(M). Define h_{ck} as

$$h_{ck}[\bar{y}_1, \ldots, \bar{y}_{n_R}, \bar{z}_1, \ldots, \bar{z}_{n_L}] = \mathrm{CONCAT}^{p(n)|\Gamma|}_{n_R+n_L}[f_{cc'_1 k}[\bar{y}_1], \ldots, f_{cc'_{n_R} k}[\bar{y}_{n_R}], g_{cc''_1 k}[\bar{z}_1], \ldots, g_{cc''_{n_L} k}[\bar{z}_{n_L}]]$$

for $1 \leq k \leq p(n)$ where $\bar{y}_i = (y_{i1}, \ldots, y_{ip(n)|\Gamma|})$ for $1 \leq i \leq n_R$, $\bar{z}_j = (z_{j1}, \ldots, z_{jp(n)|\Gamma|})$ for $1 \leq j \leq n_L$ and

$$\mathrm{CONCAT}^s_r[(x_{11}, x_{12}, \ldots, x_{1s}), \ldots, (x_{r1}, x_{r2}, \ldots, x_{rs})] = (x_{11}x_{21}\cdots x_{r1}, \ldots, x_{1s}x_{2s}\cdots x_{rs}).$$

Add h_{ck} to F and add

$$A_{qk} \to h_{ck}[A_{q'_1 k+1}, \ldots, A_{q'_{n_R} k+1}, A_{q''_1 k-1}, \ldots, A_{q''_{n_L} k-1}] \tag{20}$$

to P for $1 \leq k \leq p(n)$.

The conditions (P3) and (P4) are realized by these productions. □

Algorithm 3.4 is deterministic and its time complexity is estimated as follows. Steps 1,2 and 3 of Algorithm 3.4 take $O(1), O(\log n)$ and $O(p(n)\log n)$ time, respectively. For Step 4, $O(p(n)\log n)$ time are required to construct a single terminating production, $p(n)$ terminating productions are constructed, and therefore this step takes $O(p^2(n)\log n)$ time. For Step 5, $O(p(n)\log n)$ and $O(\log n)$ time are required to construct a single function and a single nonterminating production, respectively. The number of functions and that of nonterminating productions to be constructed are both $O(p(n))$, and hence this step takes $O(p^2(n)\log n)$ time. As a whole, Algorithm 3.4 can be executed in $O(p^2(n)\log n)$ time.

Theorem 3.7: In Algorithm 3.4,

$$M \text{ accepts } t \textit{ iff } w \in L(G).$$

Proof. It suffices to show that $(q_S, 1, t) \in \mathrm{ACC}(M)$ if and only if there is $\bar{w} = (w_1, \ldots, w_{p(n)|\Gamma|}) \in L_G(A_{q_S 1})$ such that

$$\begin{aligned} w_{\langle r, t_r \rangle} &= \varepsilon \qquad (1 \leq r \leq n) \\ w_{\langle r, B \rangle} &= \varepsilon \qquad (n < r \leq p(n)), \end{aligned}$$

which is shown by following two lemmas. □

Lemma 3.8: Let $\bar{w} = (w_1, w_2, \ldots, w_{p(n)|\Gamma|}) \in L_G(A_{qk})$ $(q \in Q, 1 \leq k \leq p(n))$. If $w_{\langle r, b_r \rangle} = \varepsilon$ $(1 \leq r \leq p(n))$ for $b_1, b_2, \ldots, b_{p(n)} \in \Gamma$, then

$$(q, k, b_1 b_2 \cdots b_{p(n)}) \in \mathrm{ACC}(M).$$

Proof. The lemma is shown by induction on the number τ of the applications of (L1) and (L2) in section 2.1.

Let $\tau = 1$ and the production used in (L1) be $A_{qk} \to (\varepsilon, \ldots, \varepsilon)$ constructed in Step 4, then $q \in q_F$. By (a) in the definition of $\mathrm{ACC}(M)$,

$$(q, k, b_1 b_2 \cdots b_{p(n)}) \in \mathrm{ACC}(M)$$

and the lemma holds.

Assume that the lemma holds for $\tau \leq \nu$, and consider a case with $\tau = \nu + 1$. Suppose that $q \in Q_U$ and the last production used in (L2) is (20) defined in Step 5. Then, $b_k = c$ and there exists $\bar{u}_i = (u_{i1}, \ldots, u_{ip(n)|\Gamma|})$ with $1 \leq i \leq n_R$ and $\bar{v}_j = (v_{j1}, \ldots, v_{jp(n)|\Gamma|})$ with $1 \leq j \leq n_L$ such that

$$\bar{u}_i \in L_G(A_{q'_i k+1}), \bar{v}_j \in L_G(A_{q''_j k-1}),$$

and

$$\bar{w} = h_{ck}[\bar{u}_1, \ldots, \bar{u}_{n_R}, \bar{v}_1, \ldots, \bar{v}_{n_L}].$$

By (9) through (11), (12) through (14) and (20), followings hold.

$$\begin{aligned} w_{\langle r, b_r \rangle} &= u_{1\langle r, b_r \rangle} \cdots u_{n_R \langle r, b_r \rangle} v_{1\langle r, b_r \rangle} \cdots v_{n_L \langle r, b_r \rangle} && (r \neq k) \\ w_{\langle k, b_k \rangle} &= u_{1\langle k, c'_1 \rangle} \cdots u_{n_R \langle k, c'_{n_R} \rangle} v_{1\langle k, c''_1 \rangle} \cdots v_{n_L \langle k, c''_{n_L} \rangle} \\ w_{\langle k, b \rangle} &= 1 \cdots 11 \cdots 1 && (b \neq c). \end{aligned}$$

Since $w_{\langle r,b_r\rangle} = \varepsilon$ $(1 \leq r \leq p(n))$,

$$\begin{array}{rcl} u_{1\langle r,b_r\rangle} = \cdots = u_{n_R\langle r,b_r\rangle} & = & v_{1\langle r,b_r\rangle} = \cdots = v_{n_L\langle r,b_r\rangle} \quad = \varepsilon \quad (r \neq k) \\ u_{1\langle k,c'_1\rangle} = \cdots = u_{n_R\langle k,c'_{n_R}\rangle} & = & v_{1\langle k,c''_1\rangle} = \cdots = v_{n_L\langle k,c''_{n_L}\rangle} \quad = \varepsilon. \end{array}$$

That is, $\bar{u}_i = (u_{i1}, \ldots, u_{ip(n)|\Gamma|}) \in L_G(A_{q'_i k+1})$ satisfies $u_{i\langle r,b_r\rangle} = \varepsilon$ $(1 \leq r \leq p(n), r \neq k)$ and $u_{i\langle k,c'_i\rangle} = \varepsilon$ for $1 \leq i \leq n_R$. By the inductive hypothesis,

$$(q'_i, k+1, b_1 b_2 \cdots b_{k-1} c'_i b_{k+1} \cdots b_{p(n)|\Gamma|}) \in \mathrm{ACC}(M)$$

for all $1 \leq i \leq n_R$. Similarly,

$$(q''_j, k-1, b_1 b_2 \cdots b_{k-1} c''_j b_{k+1} \cdots b_{p(n)|\Gamma|}) \in \mathrm{ACC}(M)$$

for all $1 \leq j \leq n_L$. Since $q \in Q_U$, it follows from (b) in the definition of $\mathrm{ACC}(M)$ and (8) that

$$(q, k, b_1 \cdots b_{p(n)}) = (q, k, b_1 \cdots b_{k-1} c b_{k+1} \cdots b_{p(n)}) \in \mathrm{ACC}(M).$$

The proof can be done in a similar way for the cases that $q \in Q_E$ and the last productions to be applied are (18) and (19) defined in Step 5. □

Lemma 3.9: If $(q, k, b_1 b_2 \cdots b_{p(n)}) \in \mathrm{ACC}(M)$, then there is $\bar{w} = (w_1, w_2, \ldots, w_{p(n)|\Gamma|}) \in L_G(A_{qk})$ such that $w_{\langle r,b_r\rangle} = \varepsilon$ for $1 \leq r \leq p(n)$.

Proof. It is shown by induction on the number τ of the applications of (a),(b) and (c) in the definition of $\mathrm{ACC}(M)$.

If $\tau = 1$, then $q \in Q_F$ and there is a production $A_{qk} \to (\varepsilon, \ldots, \varepsilon)$ constructed in Step 4. Hence, $(\varepsilon, \ldots, \varepsilon) \in L_G(A_{qk})$ and the lemma holds clearly.

Assume that the lemma holds for every $\tau \leq \nu$, and consider a case with $\tau = \nu + 1$. Suppose that $q \in Q_U$ and

$$\begin{array}{rcl} \delta(q, b_k) & = & \{(q'_i, c'_i, R) \mid 1 \leq i \leq n_R\} \\ & & \cup \{(q''_j, c''_j, L) \mid 1 \leq j \leq n_L\}. \end{array}$$

Since $(q, k, b_1 \cdots b_{p(n)}) \in \mathrm{ACC}(M)$, followings hold by (b) in the definition of $\mathrm{ACC}(M)$.

$$\begin{array}{llll} (q'_i, & k+1, & b_1 \cdots b_{k-1} c'_i b_{k+1} \cdots b_{p(n)}) \in \mathrm{ACC}(M) & (1 \leq i \leq n_R) \\ (q''_j, & k-1, & b_1 \cdots b_{k-1} c''_j b_{k+1} \cdots b_{p(n)}) \in \mathrm{ACC}(M) & (1 \leq j \leq n_L). \end{array}$$

By the inductive hypothesis, there exists

$$\bar{u}_i = (u_{i1}, \ldots, u_{ip(n)|\Gamma|}) \in L_G(A_{q'_i k+1}) \qquad (1 \leq i \leq n_R)$$

such that $u_{i\langle r,b_r\rangle} = \varepsilon$ $(1 \leq r \leq p(n), r \neq k)$ and $u_{i\langle k,c'_i\rangle} = \varepsilon$ for $1 \leq i \leq n_R$. Similarly, there exists

$$\bar{v}_j = (v_{j1}, \ldots, v_{jp(n)|\Gamma|}) \in L_G(A_{q''_j k-1}) \qquad (1 \leq j \leq n_L)$$

such that $v_{j\langle r,b_r\rangle} = \varepsilon$ $(1 \leq r \leq p(n), r \neq k)$ and $v_{j\langle k,c''_j\rangle} = \varepsilon$ for $1 \leq j \leq n_L$. Define

$$\bar{w} = h_{ck}[\bar{u}_1, \ldots, \bar{u}_{n_R}, \bar{v}_1, \ldots, \bar{v}_{n_L}]$$

then $\bar{w} \in L_G(A_{qk})$ by (20) in Step 5, and followings hold by the definition of h_{ck}.

$$\begin{aligned} w_{\langle r,b_r\rangle} &= u_{1\langle r,b_r\rangle} \cdots u_{n_R\langle r,b_r\rangle} v_{1\langle r,b_r\rangle} \cdots v_{n_L\langle r,b_r\rangle} = \varepsilon \quad (r \neq k) \\ w_{\langle k,b_k\rangle} &= u_{1\langle k,c'_1\rangle} \cdots u_{n_R\langle k,c'_{n_R}\rangle} v_{1\langle k,c''_1\rangle} \cdots v_{n_L\langle k,c''_{n_L}\rangle} = \varepsilon. \end{aligned}$$

Thus the lemma holds. The case $q \in Q_E$ can be handled in a similar way. □

Now, Lemma 3.5, the estimation of time complexity of Algorithm 3.4 and Theorem 3.7 imply following theorem.

Theorem 3.10: The universal recognition problems for PMCFG and for MCFG are both EXP-POLY time-complete. □

3.2 With Non-Erasing Condition

In this section, the universal recognition problems for NEPMCFG and for NEMCFG are both shown to be PSPACE-complete. As in Section 3.1, it suffices to show that the problem for NEPMCFG belongs to PSPACE, and that the problem for NEMCFG is PSPACE-hard.

First, a nondeterministic algorithm to solve the problem for NEPMCFG is presented. The algorithm solves the problem in polynomial space by using Algorithm 3.2 and a slightly modified version of the algorithm in the proof of Lemma 3.1.

Let $G = (V_N, V_T, F, P, S)$ be a pmcfg with non-erasing condition and w be an input string. By the non-erasing condition, if $(S^{[1]}, 0) \stackrel{*}{\Rightarrow} \alpha \stackrel{*}{\Rightarrow} w$ and some $(A^{[k]}, v)$ appears in α, then every $(A^{[j]}, v)$ $(j \neq k, 1 \leq j \leq d(A))$ also appears in α. Therefore, only the entries $\mathrm{NULL}(A, (1, 2, \ldots, d(A)))$ for $A \in V_N$ are needed in Algorithm 3.2 for a pmcfg with non-erasing condition. Since the number of such entries is $O(|V_N|) = O(n)$, Algorithm 3.2 can be executed in $O(n^2 \log n)$ space by the analysis of Algorithm 3.2 in section 3.1, where $n = |G| + |w|$. These entries of the table NULL can be constructed in $O(|G|) = O(n)$ space. Hence, the next lemma holds.

Lemma 3.11: The universal recognition problem for NEPMCFG is in PSPACE. □

Next, it is shown that the universal recognition problem for NEMCFG is PSPACE-hard. If a problem belongs to PSPACE, then there is some polynomial p and some $p(n)$ space-bounded Turing machine $M = (Q, \Sigma, \Gamma, B, \delta, q_S, Q_F)$ which solves that problem. The problem whether M accepts t for a given $t \in \Gamma^*$ can be reduced to the universal recognition problem for NEMCFG by using the following deterministic algorithm in time polynomial in $|t|$.

Remind the basic idea and Algorithm 3.3 described in 3.1.2. The construction of mcfg is similar to that of Algorithm 3.3. The differences are:

(a) Remember that mcfg G constructed in Algorithm 3.3 does not satisfy non-erasing condition. For the mcfg constructed here to satisfy non-erasing condition, the extra component $f_{cc'k}^{p(n)|\Gamma|+1}$ is introduced and

the value of $f_{cc'k}^{p(n)|\Gamma|+1}$ is defined to be the concatenation of all the components of the arguments which do not appear in the right-hand side of the definition of any $f_{cc'k}^{h}$ for $1 \leq h \leq p(n)|\Gamma|$. Similarly, the extra component $g_{cc''k}^{p(n)|\Gamma|+1}$ is introduced for each $g_{cc''k}$.

(b) The extra component introduced in (a) derives, as it stands, some string whose length can increase in proportional to the length of the derivation. To restrict the length of the terminal string which the extra component derives to be less than some constant, the roles of ε's and 1's are interchanged. And the productions are constructed in such a way that a component symbol can derive ε whenever it can derive 1 so that the value of $f_{cc'k}^{p(n)|\Gamma|+1}$ can always be made ε. Remark that the construction here does not work in the case of an ATM since, by the productions constructed in (20) universal states, a string derived from the initial symbol which represents an ID in ACC(M) becomes a string of 1's and its length may be exponential to n.

Let assume that $ID_1, ID_2, ID_3, \alpha_1, \alpha_2$ and α_3 are the same as defined in 3.1.2. Productions are constructed so that the following (P8) through (P10) hold.

(P8) If $q \in Q_F$, then $\alpha_1 \underset{G}{\Rightarrow} 1^{p(n)}$.

(P9a) If $(q', c', R) \in \delta(q, c)$, that is, if $ID_1 \vdash ID_2$, then $\alpha_1 \underset{G}{\Rightarrow} \alpha_2$.

(P9b) If $(q'', c'', L) \in \delta(q, c)$, that is, if $ID_1 \vdash ID_3$, then $\alpha_1 \underset{G}{\Rightarrow} \alpha_3$.

(P10) If $\alpha \underset{G}{\Rightarrow} \alpha'$ other than those in (P8), (P9a) and (P9b), then α derives at most $p(n) - 1$ 1's.

If productions are constructed to satisfy (P8) through (P10) for every $q \in Q$ and $c \in \Gamma$, it can be shown that

$$ID_1 \in \text{ACC}(M) \textit{ iff } \alpha_1 \overset{*}{\underset{G}{\Rightarrow}} 1^{p(n)}.$$

Algorithm 3.5:

input : a string $t = t_1 t_2 \cdots t_n$ over Σ.

output : mcfg $G = (V_N, V_T, F, P, S)$ with non-erasing condition and string w such that M accepts t *iff* $w \in L(G)$.

The mcfg G and the string w are constructed as follows.

Step 1. Let $V_T = \{1, \#\}$ and let $w = 1^{p(n)}\#$.

Step 2. Let $V_N = \{S, D\} \cup \{A_{qk} \mid q \in Q, 1 \leq k \leq p(n)\}$ where $d(A_{qk}) = p(n)|\Gamma| + 1$ $(q \in Q, 1 \leq k \leq p(n))$ and $d(D) = 1$.

Step 3. Define f as follows.

$$f[\bar{x}] = x_{\langle 1,t_1\rangle} x_{\langle 2,t_2\rangle} \cdots x_{\langle n,t_n\rangle} x_{\langle n+1,B\rangle} \cdots x_{\langle p(n),B\rangle} \#$$
$$(\prod_{r=1}^{n} \prod_{b\in\Gamma, b\neq t_r} x_{\langle r,b\rangle})(\prod_{r=n+1}^{p(n)} \prod_{b\in\Sigma} x_{\langle r,b\rangle}) x_{p(n)|\Gamma|+1}$$

where $\bar{x} = (x_1, x_2, \ldots, x_{p(n)|\Gamma|+1})$ and $\prod_{i=k}^{l} \alpha_i$ denotes the concatenation $\alpha_k \alpha_{k+1} \cdots \alpha_l$. Add f to F and add $S \to f[A_{q_S 1}]$ to P [3].

Step 4. Add terminating productions $D \to 1$ and $D \to \varepsilon$.

Step 5. Add

$$i[(x_1), \ldots, (x_{p(n)|\Gamma|})] = (x_1, \ldots, x_{p(n)|\Gamma|}, \varepsilon)$$

to F. For each $q \in Q_F$ and k $(1 \leq k \leq p(n))$, add

$$A_{qk} \to i[D, D, \ldots, D]$$

to P.

Step 6. If $(q', c', R) \in \delta(q, c)$, then add $f_{cc'k}$ to F and $A_{qk} \to f_{cc'k}[A_{q'k+1}]$ for each k $(1 \leq k < p(n))$, where $f_{cc'k}$ is defined as

$$\begin{aligned}
f_{cc'k}^{\langle r,b\rangle}[\bar{x}] &= x_{\langle r,b\rangle} && (r \neq k) \\
f_{cc'k}^{\langle k,c\rangle}[\bar{x}] &= x_{\langle k,c'\rangle} && \\
f_{cc'k}^{\langle k,b\rangle}[\bar{x}] &= \varepsilon && (b \neq c) \\
f_{cc'k}^{p(n)|\Gamma|+1}[\bar{x}] &= (\prod_{b\in\Gamma, b\neq c'} x_{\langle k,b\rangle}) x_{p(n)|\Gamma|+1}. &&
\end{aligned}$$

[3] The parenthesis are used only to get rid of ambiguity, and it is not included in the definition of f.

Step 7. If $(q'', c'', L) \in \delta(q, c)$, then add $g_{cc''k}$ to F and $A_{qk} \rightarrow g_{cc''k}[A_{q''k-1}]$, for each k $(1 < k \leq p(n))$, where $g_{cc''k}$ is defined as

$$\begin{aligned}
g^{\langle r,b\rangle}_{cc''k}[\bar{x}] &= x_{\langle r,b\rangle} && (r \neq k) \\
g^{\langle k,c\rangle}_{cc''k}[\bar{x}] &= x_{\langle k,c''\rangle} && \\
g^{\langle k,b\rangle}_{cc''k}[\bar{x}] &= \varepsilon && (b \neq c) \\
g^{p(n)|\Gamma|+1}_{cc''k}[\bar{x}] &= (\prod_{b\in\Gamma, b\neq c''} x_{\langle k,b\rangle}) x_{p(n)|\Gamma|+1}. &&
\end{aligned}$$

□

It can be easily shown that the above algorithm can be executed in $O(p^2(n)\log n)$ time in the same way as is the case of Algorithm 3.3.

Following lemma can be shown in a same way as Theorem 3.7.

Lemma 3.12:

$$M \text{ accepts } t \textit{ iff } w \in L(G).$$

□

Now, Lemmas 3.11 and 3.12 imply the following theorem.

Theorem 3.13: The universal recognition problems for NEPMCFG and for NEMCFG are both PSPACE-complete. □

3.3 Bounded Dimension

In this section, the computational complexities of the universal recognition problems for m-PMCFG and for m-MCFG are investigated. The result here is that, for any fixed m ($m \geq 1$), the problem for m-PMCFG is $\mathcal{NP}$-complete, and that the problem for m-MCFG is also $\mathcal{NP}$-complete for any fixed m ($m \geq 2$).

3.3.1 m-MCFG with $m \geq 2$

First, the universal recognition problem for m-MCFG ($m \geq 2$) is shown to be $\mathcal{NP}$-complete. In this paper, only the case $m \geq 2$ is investigated since, for $m = 1$, any 1-mcfg is also a cfg and the time complexity of the universal recognition problem for cfg's is known to be $O(|G|^2|w|^3)$[4].

It is shown that the problem for m-MCFG is in $\mathcal{NP}$ as follows. Let G be a given m-mcfg and w be an input string. First, G is transformed into an m-mcfg G' such that $L(G) = L(G')$ as follows. If $(\varepsilon, \ldots, \varepsilon) \in L_G(A)$, then the nonterminal symbol A is called a *nullable symbol.*

Step 1. By using Lemmas 1 and 3 in Ref.[14], construct m-mcfg $G'' = (V_N'', V_T, F'', P'', S)$ which satisfies $L(G) = L(G'')$ and the following conditions (f3) and (f4).

(f3) G'' satisfies the non-erasing condition.

(f4) No terminal symbols appear in the right-hand side of the definition of any $f \in F$ such that $a(f) \geq 1$.

Step 2. Construct m-mcfg $G' = (V_N', V_T, F', P', S)$ from G'' by adding $A \rightarrow (\varepsilon, \ldots, \varepsilon)$ to P'' for every nullable symbol A in V_N'' (such a production is called an *ε-production*).

By the construction of G'' in Ref.[14], $|G''| \leq 2^m|G|$ and Step 1 can be executed in $O(2^m|G|^2) = O(|G|^2)$ time deterministically since m is a constant. Consider Step 2. Since G'' satisfies the above conditions (f3) and (f4), nullable symbols can be found as follows.

(a) If $A \rightarrow (\varepsilon, \ldots, \varepsilon) \in P''$, then A is a nullable symbol.

(b) If $A \to f(A_1, A_2 \ldots, A_{a(f)}) \in P''$ and $A_1, \ldots, A_{a(f)}$ are nullable symbols, then A is also a nullable symbol.

(c) Repeat (b) until no more nullable symbols can be found.

The above procedure halts in $O(|G|^2)$ time deterministically. Furthermore, $|G'| \leq 2|G''| \in O(|G|)$.

Next, a derivation tree t in G' is generated nondeterministically and is tested whether t is a derivation tree of w. The following lemma claims that if $\bar{w} \in L(G')$ then $\bar{w}$ has a derivation tree whose size is not greater than some polynomial in $|G'| + |w|$, where the *size* of a derivation tree is defined to be the number of nodes in t'.

Lemma 3.14: Let $G' = (V'_N, V_T, F', P', S)$ be the mcfg constructed from a given G in the above discussion. If $w \in L(G')$, then w has a derivation tree in G' whose size is $O(|w||G'|^2)$.

Proof. Let $M = \dfrac{(2m-1)!}{(m-1)!}$. Since m is a constant, M is also a constant. For a tuple $\bar{w} = (w_1, w_2, \ldots, w_n)$ of strings, let $|\bar{w}| \triangleq \sum_{k=1}^{n} |w_k|$, and $|\bar{w}|$ is called the *length* of $\bar{w}$. If t is a derivation tree of $\bar{w}$ whose size is not greater than that of any derivation tree of $\bar{w}$, then t is called a *minimal derivation tree* of $\bar{w}$. It will be shown that if $\bar{w} \in L_{G'}(A)$, then there is a derivation tree of $\bar{w}$ whose size is not greater than

$$\begin{cases} 2 & \text{if } A \to \bar{w} \in P', \\ 2M(2|\bar{w}| - 1)|G'|^2 & \text{otherwise.} \end{cases}$$

This is shown by induction on the length of $\bar{w}$.

The lemma holds clearly when $|\bar{w}| = 0$, since $\bar{w} = (\varepsilon, \ldots, \varepsilon)$ and $A \to \bar{w} \in P'$.

Suppose that the claim holds for every tuples of length s or less. Let $|\bar{w}| = s + 1$, and let t be a minimal derivation tree of $\bar{w}$ as shown in Figure 5, where $\bar{\varepsilon}$ denotes a tuple of ε's. Let $p : v_0, v_1, \ldots, v_k$ be the longest path from the root which satisfies the following conditions:

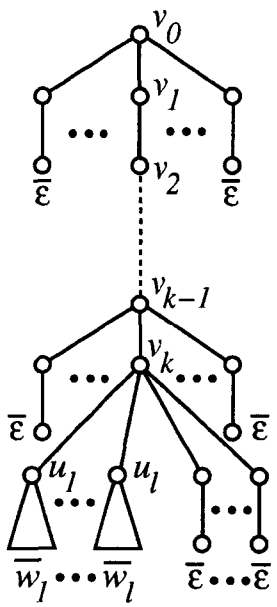


Figure 5: A derivation tree in G'

[1] $\bar{\varepsilon}$ is generated from every child of $v_j (0 \leq j < k)$ other than v_{j+1},

[2] $\bar{w}_1, \bar{w}_2, \ldots, \bar{w}_l$ $(l \geq 1)$ such that $\bar{w}_i \neq \bar{\varepsilon}$ $(1 \leq i \leq l)$ are generated from children $u_1, u_2, \ldots, u_l$ of v_k respectively, and

[3] $\bar{\varepsilon}$ is generated from every child of v_k other than $u_1, u_2 \ldots, u_l$.

Let $label(v)$ denote a nonterminal symbol or a terminal string attached to a node v as a label and let $g(v) \triangleq (g^1(v), g^2(v), \ldots, g^{d(lable(v))})$ denote the tuple of strings derived from a node v in t. For $j (0 \leq j \leq k)$, $\bar{w} = g(v_0)$ can be represented by $g(v_j)$ as follows:

$$g^h(v_0) = g^{\xi_j(h,1)}(v_j) g^{\xi_j(h,2)}(v_j) \cdots g^{\xi_j(h,n_{h_j})}(v_j) \quad (1 \leq h \leq d(label(v_0)))$$

where for each h' $(1 \leq h' \leq d(label(v_j)))$ there exist unique h and l such that $h' = \xi_j(h, l)$ by the non-erasing condition, that is, each component $g^{h'}(v_j)$ of $g(v_j)$ $(1 \leq h' \leq d(label(v_j)))$ appears exactly once in $g(v_0)$. Let

$$\begin{aligned} \mathrm{REP}(v_j) \triangleq & (\xi_j(1,1)\xi_j(1,2) \cdots \xi_j(1, n_{1j}), \\ & \ldots, \xi_j(d(label(v_i)), 1) \cdots \xi_j(d(label(v_i)), n_{d(label(v_i))j})). \end{aligned}$$

For example, if

$$g(v_i) = (g^5(v_j) g^3(v_j), \varepsilon, g^2(v_j) g^4(v_j), g^1(v_j))$$

then $\mathrm{REP}(v_j) = (53, \varepsilon, 24, 1)$. The number of distinct $\mathrm{REP}(v_j)$'s among v_j's labeled with an identical nonterminal symbol is not greater than $\dfrac{(2m-1)!}{(m-1)!}$.

Assume that $k \geq M|V'_N|$. Then there exist distinct nodes v_p and $v_q (0 \leq p < q \leq k)$ such that $label(v_p) = label(v_q)$ and $\mathrm{REP}(p) = \mathrm{REP}(q)$. Let t' denote the tree obtained from t by replacing the subtree whose root is v_p with the subtree whose root is v_q. It is obvious that t' is also a derivation tree of $\bar{w}$ and the size of t' is less than that of t, a contradiction. That is, $k < M|V'_N|$. Let ν denote the maximal number of arguments of a function in F'. By Step 2 of the construction of G', the definition of path p and the fact that t is minimal, an ε-production is applied at every child of v_j other than v_{j+1} $(0 \leq j < k)$. It follows that the size of "upper part" of t (that is, "the size of t" minus "the size of subtree whose root is v_k") is not greater than $2(\nu - 1)k + k = 2k\nu - k$.

If $l = 1$, then the size of the subtree whose root is u_1 is 2 by the definition of path p. In this case, the size of t is not greater than

$$2k\nu - k + 1 + 2\nu \leq 2k\nu + 1 + 2\nu = 2(k+1)\nu + 1.$$

Since $k < M|V'_N|$ and both ν and $|V'_N|$ are less than $|G'|$,

$$2(k+1)\nu + 1 \leq 2M|V'_N|\nu + 1 < 2M|G'|^2.$$

Clearly, $2M|G'|^2 \leq 2M(2|\bar{w}| - 1)|G'|^2$ and the lemma holds.

If $l \geq 2$, then by the inductive hypothesis, the sizes of the subtrees whose roots are $u_1, u_2, \ldots,$ and u_l are not greater than $2M(2|\bar{w}_1| - 1)|G'|^2, 2M(2|\bar{w}_2| - 1)|G'|^2, \ldots,$ and $2M(2|\bar{w}_l| - 1)|G'|^2$, respectively. The size of t is not greater than

$$2k\nu - k + 1 + 2(\nu - l) + \sum_{i=1}^{l} 2M(2|\bar{w}_i| - 1)|G'|^2. \tag{21}$$

By using $|\bar{w}_1| + |\bar{w}_2| + \cdots + |\bar{w}_l| = |\bar{w}|, l \geq 2$ and the following inequality

$$2k\nu - k + 1 + 2(\nu - l) < 2(k+1)\nu + 1 < 2M|G'|^2,$$

the expression (21) is upper bounded by

$$2M|G'|^2 + 2M(2|\bar{w}| - l)|G'|^2 \leq 2M(2|\bar{w}| - 1)|G'|^2.$$

Thus the induction completes. It follows that if $w \in L(G')$, then w has a derivation tree in G' whose size is $O(|w||G'|^2)$. □

For a derivation tree t, it can be easily shown that linear time to the size of t is sufficient to test whether t is a derivation tree of w. Since $|G'| \in O(|G|)$, Lemma 3.14 implies the following lemma.

Lemma 3.15: For any fixed m ($m \geq 2$), the universal recognition problem for m-MCFG belongs to $\mathcal{NP}$. □

Next, the problem is shown to be $\mathcal{NP}$-hard. It is sufficient to consider the case $m = 2$, since any 2-mcfg is also an m-mcfg for every $m \geq 2$.

By the following deterministic algorithm 3.6, 3SAT (the satisfiability problem of 3-conjunctive normal form Boolean expressions), which is known to be $\mathcal{NP}$-complete[9], is reducible to the universal recognition problem for 2-MCFG in polynomial time. The formal description of the algorithm is given first, followed by a simple example.

Algorithm 3.6:

input : 3-CNF Boolean expression E.
output : 2-mcfg $G = (V_N, V_T, F, P, S)$ and string w such that E is satisfiable *iff* $w \in L(G)$.

Let $E = E_1 \wedge E_2 \wedge \cdots \wedge E_q$, and $E_i = (l_{i1} \vee l_{i2} \vee l_{i3})$ where l_{ij} is a literal $(1 \leq i \leq q, 1 \leq j \leq 3)$.

Step 1. Count the distinct variables appearing in E. Let r be the number of them and let those distinct variables be $p_1, p_2, \ldots,$ and p_r.

Step 2. Let m_i and m'_i $(1 \leq i \leq r)$ be the numbers of the occurrences of positive literal p_i and negative literal $\neg p_i$ in E, respectively. Without loss of generality, assume that $m_i \geq m'_i$. Furthermore, let $t = \sum_{i=1}^{r}(m_i - m'_i)$. In Step 4, y_i's $(1 \leq i \leq t)$ will be used as fillers if the number of the occurrences of p_i is strictly greater than that of $\neg p_i$.

Step 3. Let $V_T = \{\$, \#, 1\}$ and $V_N = \{S, X, Y, A\}$, where $d(X) = d(A) = 2$, $d(Y) = d(S) = 1$.

Step 4. Let $F = \{f\}$ where f is defined as follows.

$$\begin{aligned} & f[\bar{a}_1, \ldots, \bar{a}_r, \bar{x}_{11}, \bar{x}_{12}, \bar{x}_{13}, \bar{x}_{21}, \ldots, \bar{x}_{q3}, y_1, \ldots, y_t] \\ &= x_{111}x_{121}x_{131}\#x_{211}x_{221}x_{231}\#\cdots\#x_{q11}x_{q21}x_{q31} \\ &\qquad \$a_{11}z_{11}z_{12}\cdots z_{1m_1}\#z'_{11}z'_{12}\cdots z'_{1m_1}a_{12} \\ &\qquad \cdots \\ &\qquad \$a_{r1}z_{r1}z_{r2}\cdots z_{rm_r}\#z'_{r1}z'_{r2}\cdots z'_{rm_r}a_{r2} \end{aligned} \tag{22}$$

where $\bar{x}_{ij} = (x_{ij1}, x_{ij2})$ $(1 \leq i \leq q, j = 1, 2, 3)$ and $\bar{a}_i = (a_{i1}, a_{i2})$ $(1 \leq i \leq r)$.

For each z_{uv} and z'_{uv} $(1 \leq u \leq r, 1 \leq v \leq m_u)$,

$$z_{uv}, z'_{uv} \in \{x_{ij2} \mid 1 \leq i \leq q, j = 1, 2, 3\} \cup \{y_k \mid 1 \leq k \leq t\}$$

and they must satisfy the followings.

(a) If the literal l_{ij} $(1 \leq i \leq q, j = 1,2,3)$ is the vth occurrence of p_u from the left, then $z_{uv} = x_{ij2}$.

(b) If the literal l_{ij} $(1 \leq i \leq q, j = 1,2,3)$ is the vth occurrence of $\neg p_u$ from the left, then $z'_{uv} = x_{ij2}$.

(c) The number of z'_{uv}'s which are not defined by the above (a) or (b) equals to t. Those z'_{uv}'s are equal to $y_1, \ldots, y_t$, respectively. (Since $m_i \geq m'_i$, it is sufficient to consider only z'_{uv}'s.)

Step 5. The rewriting productions are defined as:

$$\begin{aligned} P \;=\; & \{S \to f[\underbrace{A, A, \ldots, A}_{r}, \underbrace{X, X, \ldots, X}_{3q}, \underbrace{Y, Y, \ldots, Y}_{t}] \\ & X \to (1,1) \mid (\varepsilon,1) \mid (\varepsilon,\varepsilon) \\ & A \to (\#,\varepsilon) \mid (\varepsilon,\#) \\ & Y \to 1 \mid \varepsilon\}. \end{aligned} \tag{23}$$

Step 6. Define the input string w as

$$w = \underbrace{1\#1\#\cdots\#1}_{q \text{ 1's}}\$\#1^{m_1}\#\$\#1^{m_2}\#\$\cdots\$\#1^{m_r}\#. \tag{24}$$

□

Example 3.1: Let $E = (p_1 \vee \neg p_2 \vee p_3) \wedge (p_1 \vee p_2 \vee \neg p_3) \wedge (\neg p_1 \vee p_3 \vee p_2)$ be an input. The word w and the function f are:

$$w = 1\#1\#1\underbrace{\$\#11\#}_{(*)}\$\#11\#\$\#11\# \tag{25}$$

$$\begin{aligned} & f[\bar{a}_1, \bar{a}_2, \bar{a}_3, \bar{x}_{11}, \bar{x}_{12}, \bar{x}_{13}, \bar{x}_{21}, \ldots, \bar{x}_{33}, y_1, y_2, y_3] \\ & = \; x_{111}x_{121}x_{131}\#x_{211}x_{221}x_{231}\#x_{311}x_{321}x_{331} \quad \text{(I)} \\ & \qquad \$a_{11}x_{112}x_{212}\#x_{312}y_1a_{12} \quad \text{(II)} \\ & \qquad \$a_{21}x_{222}x_{332}\#x_{122}y_2a_{22} \quad \text{(III)} \\ & \qquad \$a_{31}x_{132}x_{322}\#x_{232}y_3a_{32}. \quad \text{(IV)} \end{aligned} \tag{26}$$

In Figure 6, the components of the same variable are linked by a line. Intuitively, the part (I) of f corresponds to the expression E, and parts (II),(III) and (IV) correspond to p_1, p_2 and p_3, respectively.

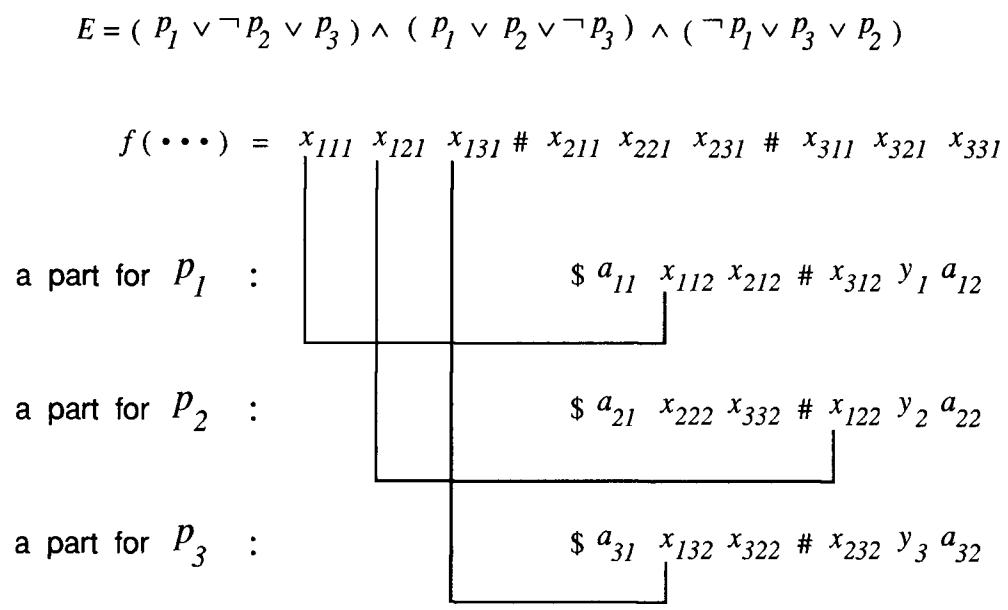


Figure 7: An example of the function f

The value of variable p_i $(1 \leq i \leq 3)$ is represented by $\bar{a}_i$ as

$$\begin{cases} \bar{a}_i = (\#, \varepsilon) & \textit{iff} \quad p_i \text{ is "TRUE"}, \\ \bar{a}_i = (\varepsilon, \#) & \textit{iff} \quad p_i \text{ is "FALSE"} \end{cases}$$

and the value of literal l_{ij} $(1 \leq i, j \leq 3)$ is represented by $\bar{x}_{ij} = (x_{ij1}, x_{ij2})$ as

$$\begin{cases} \bar{x}_{ij} = (1,1) \text{ or } (\varepsilon, 1) & \textit{iff} \quad l_{ij} \text{ is "TRUE"}, \\ \bar{x}_{ij} = (\varepsilon, \varepsilon) & \textit{iff} \quad l_{ij} \text{ is "FALSE"}. \end{cases} \tag{27}$$

The first component x_{ij1} of $\bar{x}_{ij}$ appears in the part (I) of (26) and is used for checking that the value of ith clause is "TRUE" or not, and the second component x_{ij2} appears in either (II),(III) or (IV) and is used for checking that, for each variable, the same value is assigned to every occurrence of the variable and the values assigned to positive and negative literals are distinct.

If the variable p_1 is to be "TRUE", then a_{11} and a_{12} are replaced by $\#$ and ε, respectively. To satisfy $w \in L(G)$, the part (II) in (26) must yield the part $(*)$ in (25), which implies that $x_{112}, x_{212}, x_{312}$ and y_1 must be replaced by $1, 1, \varepsilon$ and ε, respectively. By (23) in the algorithm, x_{111} and x_{211} which correspond to p_1 are replaced by 1 or ε, and x_{311} which corresponds to $\neg p_1$ is replaced by ε. In the case where p_1 is to be "FALSE", and in the cases of p_2 and p_3, a similar discussion follows. Hence, the first component x_{ij1} which corresponds to the literal l_{ij} to be "TRUE" is replaced by 1 or ε, and the one which corresponds to the literal to be "FALSE" is replaced by ε.

Now, it is clear that the part (I) in (26) can generate 1#1#1 if and only if at least one literal in each clause are to be "TRUE", that is, E is satisfiable. In this example, E is satisfiable and $w \in L(G)$. □

Apparently, Algorithm 3.6 can be executed in $O(p(|E|))$ time for some polynomial p where $|E|$ is the length of a description of E.

Lemma 3.16: E is satisfiable *iff* $w \in L(G)$.

(*If part*) Assume $w \in L(G)$. Then there exist $\bar{\alpha}_u$ $(1 \leq u \leq r)$, $\bar{\beta}_{ij}$ $(1 \leq i \leq q, j = 1, 2, 3)$ and γ_i $(1 \leq i \leq t)$ such that

$$f[\bar{\alpha}_1, \ldots, \bar{\alpha}_r, \bar{\beta}_{11}, \ldots, \bar{\beta}_{q3}, \gamma_1, \ldots, \gamma_t] = w \in L_G(S) = L(G). \tag{28}$$

By the definition of f, P and w (see Steps 4 through 6 in Algorithm 3.6), either $\bar{\alpha}_u = (\#, \varepsilon)$ or $\bar{\alpha}_u = (\varepsilon, \#)$ for each $1 \leq u \leq r$. Define $(v_1, \ldots, v_r)$ as

$$v_u = \begin{cases} \text{"TRUE"} & \text{if } \bar{\alpha}_u = (\#, \varepsilon) \\ \text{"FALSE"} & \text{if } \bar{\alpha}_u = (\varepsilon, \#) \end{cases} \tag{29}$$

for $1 \leq u \leq r$ and assign v_u $(1 \leq u \leq r)$ to the variable p_u.

Consider a clause $l_{i1} \vee l_{i2} \vee l_{i3}$ $(1 \leq i \leq q)$. Since $w = \underbrace{1\#1 \cdots \#1}_{q \text{ 1's}}\$\#$ $\cdots \# \in L(G)$ and $L_G(X) = \{(1,1), (\varepsilon, 1), (\varepsilon, \varepsilon)\}$, exactly one of $\bar{\beta}_{i1}, \bar{\beta}_{i2}$ and $\bar{\beta}_{i3}$ is $(1,1)$ by the definition of f. Without loss of generality, let $\bar{\beta}_{i1} = (1,1)$. If $l_{i1} = p_u$, then x_{i12} is on the left of the $(q-1+u)$th $\#$ by the definition of f. Hence, by (28), w can be written as

$$w = \cdots \$\alpha_{u1} \cdots 1 \cdots \# \cdots \alpha_{u2}\$ \cdots,$$

where $\bar{\alpha}_u = (\alpha_{u1}, \alpha_{u2})$. By the definition of w in (24) of Algorithm 3.6, $\bar{\alpha}_u = (\#, \varepsilon)$ and the value v_u assigned to l_{i1} is "TRUE" by (29). If l_{i1} is $\neg p_u$, then l_{i1} is also shown to be "TRUE" in the same manner. It follows that E is satisfiable.

(*Only if part*) Assume E is satisfiable, then there exists an assignment $(v_1, v_2, \ldots, v_r)$ which makes the value of E "TRUE", where v_i is the value assigned to p_i. Under this assignment, at least one of l_{i1}, l_{i2} and l_{i3} is "TRUE" for each i $(1 \leq i \leq q)$. Choose one of such literals for each i, say $l_{1j_1}, l_{2j_2}, \ldots, l_{qj_q}$. Under the above assignment, let

$$\bar{\alpha}_u = \begin{cases} (\#, \varepsilon) & \text{if } v_u = \text{"TRUE"} \\ (\varepsilon, \#) & \text{if } v_u = \text{"FALSE"} \end{cases}$$

for $1 \leq u \leq r$ and let

$$\bar{\beta}_{ij} = \begin{cases} (1,1) & \text{if the value of } l_{ij} \text{ is "TRUE" and } j = j_i \\ (\varepsilon, 1) & \text{if the value of } l_{ij} \text{ is "TRUE" and } j \neq j_i \\ (\varepsilon, \varepsilon) & \text{if the value of } l_{ij} \text{ is "FALSE"} \end{cases}$$

for $1 \leq i \leq q$ and $j = 1, 2, 3$. If $z'_{uv} = y_i$ at Step 4 (intuitively, y_i is a filler for the variable p_u), then let

$$\gamma_i = \begin{cases} 1 & \text{if } v_u = \text{"FALSE"} \\ \varepsilon & \text{if } v_u = \text{"TRUE"} \end{cases}$$

for $1 \leq i \leq t$.

By the definition of G, $\bar{\alpha}_u \in L_G(A), \bar{\beta}_{ij} \in L_G(X)$ and $\gamma_i \in L_G(Y)$. By (22) and (24) in Algorithm 3.6,

$$f[\bar{\alpha}_1, \ldots, \bar{\alpha}_r, \bar{\beta}_{11}, \ldots, \bar{\beta}_{q3}, \gamma_1, \ldots, \gamma_t] = w \in L_G(S) = L(G).$$

□

Lemma 3.16 implies that by using Algorithm 3.6, 3SAT is reducible to the universal recognition problem for 2-MCFG.

The following theorem summarizes the above results.

Theorem 3.17: For any fixed m ($m \geq 2$), the universal recognition problem for m-MCFG is $\mathcal{NP}$-complete. □

3.3.2 m-PMCFG with $m \geq 1$

In this section, it will be shown that the problem for m-PMCFG ($m \geq 1$) is also $\mathcal{NP}$-complete.

Lemma 3.18: For any fixed m ($m \geq 1$), the universal recognition problem for m-PMCFG belongs to $\mathcal{NP}$.

Sketch of Proof: Let $p : v_0, v_1, \ldots, v_k$ be the path as described in Lemma 3.14. Since G does not always satisfy the condition (f2) in this case, a copy operation is permitted, and the length of the string derived from the node v_i ($0 \leq i < k$) can be greater than that of v_{i+1} (see Example 2 in **2**). Let $len(v) \triangleq |g(v)|$, that is, the length of the tuple derived from a node v. By the assumption, $len(v_0) = |\bar{w}|$, and by the condition 2 of the definition of the path p, $len(v_k) \geq 1$. Let $i_0, i_1, \ldots, i_c$ ($0 = i_0 \leq i_1 < \cdots < i_c = k$) be the integers such that

$$|w| = len(v_{i_0}) = len(v_{i_1}) > len(v_{i_1+1}) = \cdots = len(v_{i_2}) > \cdots len(v_{i_c}) \geq 1.$$

Clearly, $c < |w|$, and it can be easily shown that $i_{b+1} - i_b \leq M|V_N|$ ($0 \leq b < c$). Therefore, $k < M|V_N||w|$.

In a similar way to the proof of Lemma 3.14, it can be shown that if $w \in L(G)$ then w has a derivation tree whose size is $O(n^4)$. □

It has already shown that the problem for m-MCFG is $\mathcal{NP}$-hard for $m \geq 2$. For m-PMCFG, the lemma holds for $m \geq 1$. Basic idea is the same as that for MCFG, and the differences are:

- For MCFG, the second component x_{ij2} of $\bar{x}_{ij} = (x_{ij1}, x_{ij2})$ is used to ensure that an identical value ("TRUE" or "FALSE") is assigned to each occurrence of a variable. This technique cannot be used for 1-PMCFG since only strings (1-tuples of a string) are available. Instead, the same mechanism is realized by copy operations.

- For MCFG, the first component x_{ij1} of $\bar{x}_{ij} = (x_{ij1}, x_{ij2})$ can be any of 1 and ε when the value of the corresponding literal l_{ij} is "TRUE" (see (27) in Example 3.1). Hence, the number of 1's in $x_{i11}, x_{i21}, x_{i31}$ for the clause $l_{i1} \vee l_{i2} \vee l_{i3}$ whose value is "TRUE" can always be made one. For 1-PMCFG, only 1 is allowed to represent the corresponding literal to take "TRUE" as the value and the number of 1's for each clause varies from 1 to 3. To make the number of 1's be a uniform value, say 4, for each clause whose value is "TRUE", three fillers ($y_{3i-2}, y_{3i-1}, y_{3i}$ in Step 3 of Algorithm 3.7) are added for each clause.

By the following deterministic algorithm, 3SAT is reducible to the universal recognition problem for 1-PMCFG in polynomial time.

Algorithm 3.7:

input : 3-CNF boolean expression E.
output : 1-pmcfg $G = (V_N, V_T, F, P, S)$ and string w such that E is satisfiable *iff* $w \in L(G)$.

Let $E = E_1 \wedge E_2 \wedge \cdots \wedge E_q$ and let $E_i = (l_{i1} \vee l_{i2} \vee l_{i3})$ where l_{ij} is a literal.

Step 1. Count the distinct variables appearing in E. Let r be the number of them and let those distinct variables be $p_1, p_2 \ldots$, and p_r.

Step 2. Let $V_T = \{\$, \#, 1\}$ and $V_N = \{S, X, Y\}$, where $d(X) = d(Y) = d(S) = 1$.

Step 3. Let $F = \{f\}$ where f is defined as follows.

$$\begin{aligned} f[x_{11}, x_{12}, x_{21}, x_{22}, \ldots, x_{r1}, x_{r2}, y_1, y_2, \ldots, y_{3q}] \\ = \; z_{11}z_{12}z_{13}y_1y_2y_3\# \cdots \# z_{q1}z_{q2}z_{q3}y_{3q-2}y_{3q-1}y_{3q} \\ \$x_{11}x_{12}\$x_{21}x_{22}\$ \cdots \$x_{p1}x_{p2}. \end{aligned}$$

For each z_{ij} $(1 \leq i \leq q, j = 1,2,3)$, $z_{ij} = x_{u1}$ if $l_{ij} = p_u$ and $z_{ij} = x_{u2}$ if $l_{ij} = \neg p_u$.

Step 4. The rewriting productions are defined as

$$\begin{aligned} P \;=\; \{S \rightarrow f[\underbrace{X, X, \ldots, X}_{2r}, \underbrace{Y, Y, \ldots, Y}_{3q}] \\ X \rightarrow 1 \mid \varepsilon \\ Y \rightarrow 1 \mid \varepsilon\}. \end{aligned}$$

Step 5. Define the string w as

$$w = \underbrace{1111\#1111\# \cdots \#1111}_{q \text{ 1111's}} \$ \underbrace{1\$1\$ \cdots \$1}_{r \text{ 1's}}.$$

□

The readers can easily verify that

$$E \text{ is satisfiable } \textit{iff } w \in L(G)$$

and that this algorithm can be executed in polynomial time, that is, the universal recognition problem for 1-PMCFG is $\mathcal{NP}$-hard, and the following theorem holds.

Theorem 3.19: For any fixed m $(m \geq 1)$, the universal recognition problem for m-PMCFG is $\mathcal{NP}$-complete. □

3.4 Bounded Degree

This section introduces subclasses of pmcfg's and mcfg's for which the universal recognition problems can be solved in deterministic polynomial time. Refer to the definition of the degree in Section 2.1. Notice that, by the definition, for each e ($e \geq 1$), there exists m ($m \geq 1$) such that every pmcfg with degree e or less is an m-pmcfg. On the other hand, for each m ($m \geq 1$) and e ($e \geq 1$), there always exists an m-pmcfg G with degree greater than e. That is, every pmcfg with bounded degree has also a bounded dimension, but not vice versa.

Lemma 3.20: For any fixed e ($e \geq 1$), the time complexity of the universal recognition problem for MCFG_e is $O(|G|^2|w|^e)$.

Sketch of proof: In Ref.[15], a fixed-language recognition algorithm for a language generated by a mcfg with bounded degree is presented. Although the algorithm assumes that a given mcfg satisfies (f3) in Lemma 2.1, ε-production freeness and some other conditions, it can be easily extended so as to be directly applicable to an arbitrary mcfg (without any equivalence transformation of the grammar). The analysis of the complexity is analogous to that of $O(|G|^2|w|^3)$ time algorithm for CFG.□

A *modified head grammar* (*MHG*) is introduced in Ref.[25] to define the syntax of natural languages. It has been shown in Ref.[16] that the generative capacity of MHG is equal to that of head grammars[19] and that of tree adjoining grammars[11], which are also introduced to define the syntax of natural languages. Since every mhg is a 2-mcfg with degree 6 or less by definition[16], the following holds as a corollary of Lemma 3.20.

Corollary 3.21: The universal recognition problem for MHG is solvable in deterministic $O(|G|^2|w|^6)$ time. □

Lemma 3.22: The universal recognition problem for 1-MCFG with degree 3 or less is $\mathcal{P}$-hard.

Sketch of proof: By Corollary 12 in Ref.[10], the universal recognition problem for Chomsky normal form CFG is $\mathcal{P}$-complete. Since any Chomsky normal form cfg is also a 1-mcfg with degree 3 or less[14], the lemma is proved. □

The next theorem is obtained from lemmas 3.20 and 3.22.

Theorem 3.23: For any fixed e ($e \geq 3$), the universal recognition problem for MCFG_e is $\mathcal{P}$-complete and solvable in deterministic $O(|G|^2|w|^e)$ time. □

In a similar way, the following theorem can be proved.

Theorem 3.24: For any fixed e ($e \geq 3$), the universal recognition problem for PMCFG_e is $\mathcal{P}$-complete and solvable in deterministic $O(|G|^2|w|^{e+1})$ time. □

3.5 Results and Their Implication

In this section, significance and implication of the results are discussed informally. Computational complexities of universal recognition problems are sometimes considered to have strong relation with "difficulty" of acquisition of a language: Suppose that one learns a language from a description of a grammar G. By compiling G (in mental representation), one will acquire an efficient parser of $L(G)$, by which one can decide whether a given text w belongs to the language generated by G or not. This process can be modeled as in Figure 7, using a parser generator for a class $\mathcal{G}$ of grammars to which G belongs. If the "acquisition of a language" means a construction of an efficient parser, the difficulty of the acquisition can be measured by the complexity of the parser generator. Let $U_{\mathcal{G}}$, $PG_{\mathcal{G}}$ and P_G be the computational complexity of the universal recognition problem for a class $\mathcal{G}$ of grammars, the computational complexity of the parser generator to produce a parser for a given G in $\mathcal{G}$, and the computational complexity of the parser to decide $w \in L(G)$ for a given w, respectively. We obtain $U_{\mathcal{G}} \leq PG_{\mathcal{G}} + P_G$. If the parser is efficient enough compared with the parser generator, i.e., $P_{\mathcal{G}} \ll PG_{\mathcal{G}}$, the difficulty of acquisition of a language is estimated by the complexity of the universal recognition problem. As for acquisition of a language, our intuition tells that an acquisition of a language from a grammar will be more difficult if the grammar allows "omissions". In the case of PMCFG, an omission of a word corresponds to an erasing of a variable on the right-hand side of the expression (2). Thereby, it is quite suitable to our intuition that the universal recognition problem for PMCFG with no constraint is more difficult than that for NEPMCFG. As mentioned above, non-erasing condition does not weaken the generative power of grammars. Now, one may think to transform a pmcfg G that violates the non-erasing condition into another weakly equivalent pmcfg G' that satisfies the non-erasing condition, and apply a polynomial space universal recognition algorithm to G'. But elimination of productions that violate the non-erasing condition may make the grammar size exponentially larger than the original one[14, 22], and hence the complexity of the universal recognition problem cannot be reduced by such a transformation.

Generally speaking, there is not much difference between the complex-

ities of the problems for PMCFG and for MCFG with the same restriction (compare results for PMCFG and MCFG in the same row in Table 2). It seems that copy operations do not have an effect on the succinctness of a grammar, which is a quite interesting contrast with the fact that erasing operations (which are inhibited by (f3)) have a great effect on the complexity of the problem.

The problem can be solved efficiently if one fixes the degree of grammars. In the language acquisition example, this result means that the acquisition of a language is tractable (i.e. performed in polynomial time) if the languages to be acquired are modeled by PMCFG(or MCFG) with fixed degree e. The remaining question is; "does the class of pmcfg's with fixed degree have enough generative power to describe the syntax of natural languages?" Our answer is positive to this question in the following sense.

Head grammars (HG)[19] and tree adjoining grammars (TAG)[11] have been widely accepted as grammatical formalisms to describe the syntax of natural languages. It has already been shown that the class of languages generated by HG and TAG coincide, and it is a proper subclass of languages generated by mcfg's with dimension 2 and degree 6[22]. Of course one can choose a larger degree and pay more to the universal recognition. This result agrees with our intuitive understanding that, for any grammars, there is a trade-off relation between the generative capacity and the difficulty of acquisition.

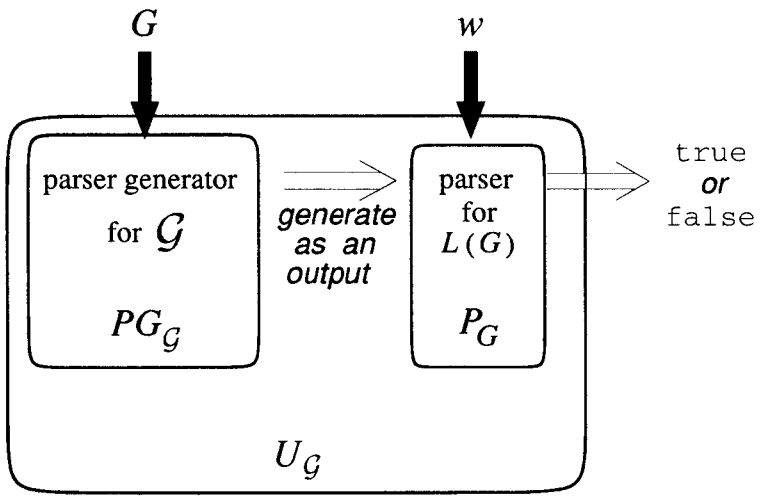


Figure 7: A model of acquisition of a language from a description of a grammar

4 Generative Powers of FSTS

This section is devided into two parts. In the first section, it is shown that yL(DFSTS), the class of yield languages generated by deterministic FSTS, equals to PMCFL. Based on this result, the concept of degree of PMCFG is extended to deterministic FSTS, and the fixed-language recognition problem for yL(DFSTS) is shown to be recognized in $O(n^{e+1})$-time where n is the length of an input word and e is the degree of the deterministic fsts. In Section 4.2, in contrast to these results, it is shown that there is a nondeterministic mondaic fsts with state-bound 2 which generates an $\mathcal{NP}$-complete language.

4.1 Deterministic FSTS

4.1.1 yL(DFSTS) $\subseteq$ PMCFL

Let (M, G) be a deterministic yT-fsts where $M = (Q, \Sigma, \Delta, q_1, R)$ and $G = (V_N, V_T, P, S)$. We assume that $Q = \{q_1, \ldots, q_\ell\}$, $V_T = \{a_1, \ldots, a_n\}$ and the productions in P are labeled with $r_1, \ldots, r_m$. Since the input domain of M is the set of derivation trees of G, we assume that $\Sigma = \{r_1, \ldots, r_m, a_1, \ldots, a_n\}$ without loss of generality.

A pmcfg $G' = (V'_N, V'_T, F', P', S')$ such that $yL(M, G) = L(G') \cap \Delta^*$ is constructed as follows. Let $V'_T = \Delta \cup \{b\}$ where b is a newly introduced symbol and let

$$V'_N = \{S', R_1, \ldots, R_m, A_1, \ldots, A_n\}$$

where $d(R_i) = d(A_j) = \ell$ for $1 \leq i \leq m$ and $1 \leq j \leq n$. Note that each R_i $(1 \leq i \leq m)$ and A_j $(1 \leq j \leq n)$ correspond to production r_i and terminal a_j of cfg G, respectively. Productions and functions of G' will be constructed to have the following property.

Property 4.1: There is $(\alpha_1, \ldots, \alpha_\ell) \in L_{G'}(R_h)$ (resp. $L_{G'}(A_h)$) such that

$$\begin{cases} \text{each of } \alpha_{s_1}, \ldots, \alpha_{s_u} \text{ does not contain } b, \text{ and} \\ \text{each of the remaining } \alpha_{t_1}, \ldots, \alpha_{t_v} \text{ contains } b \end{cases}$$

if and only if there is a derivation tree t of G such that the root of t is r_h

(resp. a_h) and

$$\begin{cases} q_{s_p}[t] \overset{*}{\Rightarrow} \alpha_{s_p} & (1 \leq p \leq u) \\ q_{t_p}[t] \text{ derives no output} & (1 \leq p \leq v). \end{cases}$$

□

The basic idea is to simulate the move of tree transducer M which is scanning a symbol r_h (resp. a_h) with state q_i by the ith component of the nonterminal R_h (resp. A_h) of pmcfg G'. During the move of M, it may happen that no rule is defined for a current configuration and hence no output will be derived. The symbol b is introduced to represent such an undefined move explicitly.

To construct productions and functions, Define $\mathrm{RS}(X)$ $(X \in V_N \cup V_T)$ as follows.

$$\mathrm{RS}(X) = \begin{cases} \{R_h \mid \text{the left-hand side of } r_h \text{ is } X\} & \text{if } X \in V_N \\ \{A_h\} & \text{if } X = a_h \in V_T. \end{cases}$$

Productions and functions are defined as follows.

Step 1: For each production $r_h : Y_0 \to Y_1 \cdots Y_k$ $(Y_0 \in V_N, Y_u \in V_N \cup V_T$ for $1 \leq u \leq k)$ of cfg G, construct nonterminating productions

$$R_h \to f_{r_h}[Z_1, \ldots, Z_k]$$

for arbitrary combinations of $Z_u \in \mathrm{RS}(Y_u)$ $(1 \leq u \leq k)$ where f_{r_h} is defined as follows: For $1 \leq i \leq \ell$,

- if there is no rule whose left-hand side is $q_i[r_h(x_1, \ldots, x_k)]$, then

$$f_{r_h}^{[i]}[\bar{x}_1, \ldots, \bar{x}_k] \triangleq b, \tag{30}$$

- if the transducer M has a rule $q_i[r_h(x_1, \ldots, x_k)] \to w_{i,1} q_{\eta(i,1)}[x_{\mu(i,1)}] w_{i,2} \cdots w_{i,n_i} q_{\eta(i,n_i)}[x_{\mu(i,n_i)}] w_{i,n_i+1}$, where $1 \leq \eta(i, j) \leq \ell$ and $1 \leq \mu(i,j) \leq k$ $(1 \leq j \leq n_i)$, then

$$f_{r_h}^{[i]}[\bar{x}_1, \ldots, \bar{x}_k] \triangleq w_{i,1} x_{\mu(i,1)\eta(i,1)} w_{i,2} \cdots w_{i,n_i} x_{\mu(i,n_i)\eta(i,n_i)} w_{i,n_i+1} \tag{31}$$

where $\bar{x}_u = (x_{u1}, \ldots, x_{u\ell})$ $(1 \leq u \leq k)$.

(Since M is deterministic, there exists at most one rule whose left-hand side is $q_i[r_h(\cdots)]$ and hence the above construction is consistent.)

Step 2: For each $a_h \in V_T$, construct a terminating production $A_h \rightarrow f_{a_h}$ where f_{a_h} is defined as follows: For $1 \leq i \leq \ell$,

- if there is no rule whose left-hand side is $q_i[a_h]$, then $f_{a_h}^{[i]} \triangleq b$.
- if $q_i[a_h] \rightarrow w_i$, then $f_{a_h}^{[i]} \triangleq w_i$.

Step 3: For each $R_h \in \mathrm{RS}(S)$, construct $S' \rightarrow f_{\mathrm{first}}[R_h]$ where $f_{\mathrm{first}}[(x_1, \ldots, x_\ell)] \triangleq x_1$. Intuitively, the right-hand side of this production corresponds to the configuration that M is in an initial state q_1 and scanning the root symbol r_h of a derivation tree, where r_h is the label of a production of G whose left-hand side is the initial symbol S.

In the following, it is shown that the pmcfg G' defined as above has Property 4.1.

(*Only if part*) It is shown by induction on the number of applications of (L1) and (L2) in section 2 to obtain a tuple of strings $(\alpha_1, \ldots, \alpha_\ell)$. For the basis, assume that $\bar{\alpha} = (\alpha_1, \ldots, \alpha_\ell) \in L_{G'}(X)$ is obtained by only one application of (L1). It is clear that the applied (terminating) production is constructed in Step 2, and hence there is some h such that $X = A_h$, $A_h \rightarrow f_{a_h}$ and $f_{a_h} = \bar{\alpha}$. Let $t = a_h$ and consider how derivations proceed from $q_i[t]$ for $1 \leq i \leq \ell$. If $\alpha_i = b$ then $f_{a_h}^{[i]} = b$ and hence there should be no rule whose left-hand side is $q_i[a_h]$. If α_i does not contain b, then transducer M has a rule $q_i[a_h] \rightarrow \alpha_i$, and the property holds.

Assume that the property holds for every tuple of strings which can be obtained by d' applications or less, and suppose the case that $(\alpha_1, \ldots, \alpha_\ell) \in L_{G'}(X)$ is obtained by $d' + 1$ applications. The last (nonterminating) production applied in (L2) must be constructed in Step 1, hence there is some h such that $X = R_h$, and the applied production is

$$R_h \rightarrow f_{r_h}[Z_1, \ldots, Z_k]. \tag{32}$$

Furthermore, there exist $\bar{\beta}_u = (\beta_{u1}, \ldots, \beta_{u\ell}) \in L_{G'}(Z_u)$ for $1 \leq u \leq k$ such that $(\alpha_1, \ldots, \alpha_\ell) = f_{r_h}[\bar{\beta}_1, \ldots, \bar{\beta}_k]$. For each u $(1 \leq u \leq k)$, if $Z_u = R_{h_u}$

for some h_u (resp. $Z_u = A_{h_u}$ for some h_u), then $\bar{\beta}_u \in L_{G'}(R_{h_u})$ (resp. $\bar{\beta}_u \in L_{G'}(A_{h_u})$), and by the inductive hypothesis there is a derivation tree t_u which satisfies Property 4.1 with $\bar{\beta}_u$. That is, the root of t_u is r_{h_u} (resp. a_{h_u}), and for v $(1 \leq v \leq \ell)$,

$$\begin{cases} q_v[t_u] \stackrel{*}{\Rightarrow} \beta_{uv} & \text{if } \beta_{uv} \text{ does not contain } b, \\ q_v[t_u] \text{ derives no output} & \text{if } \beta_{uv} \text{ contains } b. \end{cases} \tag{33}$$

We note that since (32) is constructed in Step 1 as a production of pmcfg G', cfg G has a production $r_h : Y_0 \rightarrow Y_1 \cdots Y_k$ and $Z_u \in \mathrm{RS}(Y_u)$ holds for $1 \leq u \leq k$. Now, $Z_u = R_{h_u} \in \mathrm{RS}(Y_u)$ (resp. $Z_u = A_{h_u} \in \mathrm{RS}(Y_u)$) holds and it follows that the left-hand side of production r_h is Y_u by the definition of RS (resp. Y_u is the terminal symbol a_{h_u}). Hence, if we take $t = r_h(t_1, \ldots, t_k)$ then t is a derivation tree of cfg G. Now, consider a derivation of M from $q_i[t]$ for $1 \leq i \leq \ell$.

- If α_i contains b, then there are two cases.
  - $f_{r_h}^{[i]}$ is defined to be b by (30). In this case, there exists no rule whose left-hand side is $q_i[r_h(x_1, \ldots, x_k)]$. Hence $q_i[t]$ derives no output and the property holds.
  - $f_{r_h}^{[i]}$ is defined by (31). In this case, α_i can be written as $\alpha_i = w_{i,1}\beta_{\mu(i,1)\eta(i,1)}w_{i,2}\cdots w_{i,n_i}\beta_{\mu(i,n_i)\eta(i,n_i)}w_{i,n_i+1}$ and $\beta_{\mu(i,j)\eta(i,j)}$ contains b for some j $(1 \leq j \leq n_i)$. By the construction of function f_{r_h} in Step 1, there is a derivation from $q_i[t]$;

    $$q_i[t] = q_i[r_h(t_1, \ldots, t_k)] \Rightarrow w_{i,1}q_{\eta(i,1)}[t_{\mu(i,1)}]w_{i,2}\cdots w_{i,n_i}q_{\eta(i,n_i)}[t_{\mu(i,n_i)}]w_{i,n_i+1}$$

    and there are no other derivation since M is deterministic. If $\beta_{\mu(i,j)\eta(i,j)}$ contains b, then by (33), $q_{\eta(i,j)}[t_{\mu(i,j)}]$ derives no output and hence $q_i[t]$ also cannot derive output.

- If α_i does not contain b, then $f_{r_h}^{[i]}$ is defined by (31), α_i can be written as $\alpha_i = w_{i,1}\beta_{\mu(i,1)\eta(i,1)}w_{i,2}\cdots w_{i,n_i}\beta_{\mu(i,n_i)\eta(i,n_i)}w_{i,n_i+1}$ and each $\beta_{\mu(i,j)\eta(i,j)}$ $(1 \leq j \leq n_i)$ does not contain b. By (33), $q_{\eta(i,j)}[t_{\mu(i,j)}] \stackrel{*}{\Rightarrow} \beta_{\mu(i,j)\eta(i,j)}$ holds for $1 \leq j \leq n_i$, hence

  $$q_i[t] = q_i[r_h(t_1, \ldots, t_k)]$$

$$
\begin{aligned}
&\Rightarrow\ w_{i,1}q_{\eta(i,1)}[t_{\mu(i,1)}]w_{i,2}\cdots w_{i,n_i}q_{\eta(i,n_i)}[t_{\mu(i,n_i)}]w_{i,n_i+1}\\
&\overset{*}{\Rightarrow}\ w_{i,1}\beta_{\mu(i,1)\eta(i,1)}w_{i,2}\cdots w_{i,n_i}\beta_{\mu(i,n_i)\eta(i,n_i)}w_{i,n_i+1}\\
&=\ \alpha_i
\end{aligned}
$$

and the property holds.

(*If part*) If part is shown by induction on the size of a derivation tree t of G. (The *size of a tree* t is the number of occurrences of symbols of the ranked alphabet appearing in t.) For the basis, assume that the size of t is one, that is, $t = a_h$ for some $a_h \in V_T$. By Step 2, there is a production $A_h \rightarrow f_{a_h}$ and the property holds.

For the inductive step, assume that the property holds for every derivation tree whose size is not greater than d', and consider a derivation tree $t = r_h(t_1, \ldots, t_k)$ with size $d' + 1$. Since t is a derivation tree of cfg G, r_h is a production of the form $Y_0 \rightarrow Y_1 \cdots Y_k$ and the root of t_u $(1 \leq u \leq k)$ is;

$$
\begin{cases}
r_{h_u} \text{ (label of a production whose left-hand side is } Y_u) & \text{if } Y_u \in V_N\\
a_{h_u} & \text{if } Y_u = a_{h_u} \in V_T.
\end{cases}
$$

By the definition of RS, $R_{h_u} \in \mathrm{RS}(Y_u)$ (or $A_{h_u} \in \mathrm{RS}(Y_u)$) holds for $1 \leq u \leq k$, and hence pmcfg G' has a production $R_h \rightarrow f_{r_h}[Z_1, \ldots, Z_k]$ where $Z_u = R_{h_u}$ (or $Z_u = A_{h_u}$). (See the construction of productions in Step 1.)

Here, the size of each subtree t_u $(1 \leq u \leq k)$ equals to or less than d', by the inductive hypothesis, there exist $\bar{\beta}_u = (\beta_{u1}, \ldots, \beta_{u\ell}) \in L_{G'}(R_{h_u})$ (or $L_{G'}(A_{h_u})$) such that $\bar{\beta}_u$ and t_u satisfy Property 4.1. That is, for v $(1 \leq v \leq \ell)$,

$$
\begin{cases}
\beta_{uv} \text{ does not contain } b & \text{if } q_v[t_u] \overset{*}{\Rightarrow} \beta_{uv},\\
\beta_{uv} \text{ contains } b & \text{if } q_v[t_u] \text{ derives no output.}
\end{cases}
\tag{34}
$$

Now, let

$$
\bar{\alpha} = (\alpha_1, \ldots, \alpha_\ell) = f_{r_h}[\bar{\beta}_1, \ldots, \bar{\beta}_k] \in L_{G'}(R_h)
$$

and consider how α_i is defined for $1 \leq i \leq \ell$.

- If there is no rule whose left-hand side is $q_i[r_h(x_1, \ldots, x_k)]$, then $q_i[t]$ derives no output. In this case, $f_{r_h}^{[i]}$ is defined to be b and hence $\alpha_i = b$, the property holds.

- If the transducer M has a rule $q_i[r_h(x_1,\ldots,x_k)] \rightarrow w_{i,1}q_{\eta(i,1)}[x_{\mu(i,1)}]$ $w_{i,2}\cdots w_{i,n_i}q_{\eta(i,n_i)}[x_{\mu(i,n_i)}]w_{i,n_i+1}$, then we can write α_i as

$$\alpha_i = w_{i,1}\beta_{\mu(i,1)\eta(i,1)}w_{i,2}\cdots w_{i,n_i}\beta_{\mu(i,n_i)\eta(i,n_i)}w_{i,n_i+1} \qquad (35)$$

  by the construction of functions in Step 1. There are two cases:

  - For some j $(1 \leq i \leq n_i)$, $q_{\eta(i,j)}[t_{\mu(i,j)}]$ derives no output and hence $q_i[t]$ also. In this case, $\beta_{\mu(i,j)\eta(i,j)}$ contains b by (34) and it follows from (35) that α_i also contains b, the property holds.
  - For every j $(1 \leq j \leq n_i)$, $q_{\eta(i,j)}[t_{\mu(i,j)}]$ derives some output. Since M is deterministic, and by (34), the derived string should be $\beta_{\mu(i,j)\eta(i,j)}$ which does not contain b.

$$\begin{aligned} q_i[t] &= q_i[r_h(t_1,\ldots,t_k)] \\ &\Rightarrow w_{i,1}q_{\eta(i,1)}[t_{\mu(i,1)}]w_{i,2}\cdots w_{i,n_i}q_{\eta(i,n_i)}[t_{\mu(i,n_i)}]w_{i,n_i+1} \\ &\overset{*}{\Rightarrow} w_{i,1}\beta_{\mu(i,1)\eta(i,1)}w_{i,2}\cdots w_{i,n_i}\beta_{\mu(i,n_i)\eta(i,n_i)}w_{i,n_i+1} \\ &= \alpha_i \end{aligned}$$

  and the property holds.

The proof of if part is completed and Property 4.1 has been proved. □

Lemma 4.1: $yL(\text{DFSTS}) \subseteq \text{PMCFL}$.

Proof. Let $L_1 \triangleq L(G') \cap \Delta^*$. Since pmcfl's are closed under intersection with a regular set[14], L_1 is also a pmcfl. We show that $yL(M,G) = L_1$. By Property 4.1 and the productions constructed in Step 3, $w \in L_1$ if and only if there is a derivation tree t of G such that

- the root of t is r_h,
- the left-hand side of r_h is the initial symbol S, and
- $q_1[t] \overset{*}{\Rightarrow} w$

and the lemma holds. □

4.1.2 PMCFL $\subseteq yL$(DFSTS)

Let $G = (V_N, V_T, F, P, S)$ be a pmcfg with dimension ℓ. Without loss of generality, G is assumed to satisfy the non-erasing condition of Lemma 2.1. Also suppose that the nonterminating productions of G are labeled with $r_1, \ldots, r_m$, and the terminating productions are labeled with $r'_1, \ldots, r'_n$. Furthermore, for each nonterminal production r_h $(1 \leq h \leq m)$, we suppose that the function of the right-hand side of rule r_h is f_h (the suffix of the function is identical to that of the production), hence each nonterminal production can be written as $r_h : Y_0 \rightarrow f_h[Y_1, \ldots, Y_k]$ $(a(f_h) = k, Y_0, \ldots, Y_k \in V_N)$. We also suppose that each terminating production can be written as $r'_h : Y_0 \rightarrow f'_h$. A yT-fsts (M, G') such that $yL(M, G') = L(G)$ is constructed as follows.

First, define a cfg $G' = (V'_N, V'_T, P', S')$ with $V'_N \triangleq \{S', R_1, \ldots, R_m\}$ and $V'_T \triangleq \{a_1, \ldots, a_n\}$. Note that each nonterminal R_i $(1 \leq i \leq m)$ and terminal a_j $(1 \leq j \leq n)$ of cfg G' correspond to nonterminating production and terminating production of pmcfg G, respectively. To construct productions, $\mathrm{RS}'(X) \subseteq V'_N \cup V'_T$ for $X \in V_N$ is defined as follows.

$$\begin{aligned} \mathrm{RS}'(X) &= \{R_h \mid \text{the left-hand side of } r_h \text{ is } X\} \\ &\cup \{a_h \mid \text{the left-hand side of } r'_h \text{ is } X\}. \end{aligned}$$

By using RS', productions P' of cfg G' are defined as follows.

Step A: For each nonterminating production $r_h : Y_0 \rightarrow f_h[Y_1, \ldots, Y_k]$ of pmcfg G, construct productions

$$\hat{r}_{hZ_1 \cdots Z_k} : R_h \rightarrow Z_1 \cdots Z_k \tag{36}$$

for $Z_u \in \mathrm{RS}'(Y_u)$ $(1 \leq u \leq k)$.

Step B: For each $Z \in \mathrm{RS}'(S)$, construct

$$\hat{r}_{\mathrm{start}} : S' \rightarrow Z. \tag{37}$$

Note that each element in $\mathrm{RS}'(S)$ corresponds to the production of pmcfg G whose left-hand side is the initial symbol S of G.

Define $\Sigma \triangleq \{$the labels of productions in $P'\} \cup V'_T$, $\rho(\hat{r}_{hZ_1 \cdots Z_k}) = k$ for $\hat{r}_{hZ_1 \cdots Z_k} : R_h \rightarrow Z_1 \cdots Z_k$, $\rho(a_h) = 1$ for $a_h \in V'_T$ and $\rho(\hat{r}_{\mathrm{start}}) = 1$, then Σ is a ranked alphabet.

Next, we define yT-transducer $M = (Q, \Sigma, \Delta, q_1, R)$ with Σ defined above and $\Delta \triangleq V_T$. Q is defined to be $\{q_1, \ldots, q_\ell\}$ (note that ℓ is the dimension of G).

The rules in R will be defined to have the following property.

Property 4.2: There is $\bar{\alpha} = (\alpha_1, \ldots, \alpha_s) \in L_G(X)$ and the last production applied to obtain $\bar{\alpha}$ is $r_h : X \to f_h[Y_1, \ldots, Y_k]$ (resp. $r'_h : X \to f'_h$) if and only if there is a derivation tree t of G' such that the root is $\hat{r}_{hZ_1 \cdots Z_k} : R_h \to Z_1 \cdots Z_k$ ($Z_u \in \mathrm{RS}'(Y_u), 1 \leq u \leq k$) (resp. terminal symbol a_h) and $q_i[t] \overset{*}{\Rightarrow} \alpha_i$ for $1 \leq i \leq s$. ($q_i[t]$ derives no output for $i > s$.) □

Intuitively saying, a derivation tree of cfg G' represents how to apply productions to obtain tuple of string. The rules of transducer M are constructed to "expand" the tree into string. The rules in R are defined as follows.

Step I: For each nonterminating production $r_h : Y_0 \to f_h[Y_1, \ldots, Y_k]$ with f_h defined as

$$f_h^{[i]}[\bar{x}_1, \ldots, \bar{x}_k] = w_{i,1} x_{\mu(i,1)\eta(i,1)} w_{i,2} \cdots w_{i,n_i} x_{\mu(i,n_i)\eta(i,n_i)} w_{i,n_i+1}$$

where $\bar{x}_u = (x_{u1}, \ldots, x_{ud(Y_u)})$ ($1 \leq u \leq k$), $1 \leq \mu(i,j) \leq k$, and $1 \leq \eta(i,j) \leq d(Y_{\mu i,j})$ ($1 \leq j \leq n_i$), define rules

$$q_i[\hat{r}_{hZ_1 \cdots Z_k}(x_1, \ldots, x_k)] \to w_{i,1} q_{\eta(i,1)}[x_{\mu(i,1)}] w_{i,2} \cdots w_{i,n_i} q_{\eta(i,n_i)}[x_{\mu(i,n_i)}] w_{i,n_i+1} \tag{38}$$

where $Z_u \in \mathrm{RS}'(Y_u)$ ($1 \leq u \leq k$) and $1 \leq i \leq d(Y_0)$.

Step II: For each terminating production $r'_h : Y_0 \to f'_h$ with f'_h defined as $f'^{[i]}_h = w_i$, define rules $q_i[a_h] \to w_i$ for $1 \leq i \leq d(Y_0)$.

Step III: Define

$$q_1[\hat{r}_{\mathrm{start}}(x)] \to q_1[x]. \tag{39}$$

It is clear that the constructed transducer M is deterministic. A transducer M and a cfg G' defined as above have Property 4.2. Following is its proof.

(*Only if part*) It is shown by induction on the number of applications of (L1) and (L2) to obtain a tuple of strings $(\alpha_1, \ldots, \alpha_s)$. For the basis, assume that $\bar{\alpha} = (\alpha_1, \ldots, \alpha_s) \in L_G(X)$ and it is obtained by one application of (L1). Then the applied terminating production is $r'_h : X \to f'_h$ where $f'_h = \bar{\alpha}$. If we take $t = a_h$, then t is a derivation tree of cfg G' and the property holds by the construction of rules in Step II.

Next, assume that the property holds for every tuple of strings which can be obtained by d' or less applications of (L1) and (L2), and consider the case that $\bar{\alpha} = (\alpha_1, \ldots, \alpha_s) \in L_G(X)$ is obtained by $d'+1$ applications. Let

$$r_h : X \to f_h[Y_1, \ldots, Y_k] \tag{40}$$

be the last production applied to obtain $\bar{\alpha}$ where f_h is defined as

$$f_h^{[i]}[\bar{x}_1, \ldots, \bar{x}_k] = w_{i,1} x_{\mu(i,1)\eta(i,1)} w_{i,2} \cdots w_{i,n_i} x_{\mu(i,n_i)\eta(i,n_i)} w_{i,n_i+1} \tag{41}$$

for $1 \leq i \leq d(X)$. Then, there are $\bar{\beta}_u = (\beta_{u1}, \ldots, \beta_{ud(Y_u)}) \in L_G(Y_u)$ $(1 \leq u \leq k)$ such that

$$\bar{\alpha} = f_h[\bar{\beta}_1, \ldots, \bar{\beta}_k]. \tag{42}$$

Each $\bar{\beta}_u$ can be obtained by d' applications or less, and there is a nonterminating production $r_{h_u} : Y_u \to f_{h_u}[Y_{u1}, \ldots, Y_{uk_u}]$ (or terminal production $r'_{h_u} : Y_u \to f'_{h_u}$) which is the last production applied to obtain $\bar{\beta}_u$. By the inductive hypothesis, there are derivation trees t_u $(1 \leq u \leq k)$ such that

$$q_v[t_u] \overset{*}{\Rightarrow} \beta_{uv} \tag{43}$$

for $1 \leq v \leq d(Y_u)$, and the root of t_u is $\hat{r}_{h_u Z_{u1} \cdots Z_{uk_u}} : R_{h_u} \to Z_{u1} \cdots Z_{uk_u}$ (or a_{h_u}). Note that $R_{h_u} \in \mathrm{RS'}(Y_u)$ (or $a_{h_u} \in \mathrm{RS'}(Y_u)$) holds for $1 \leq u \leq k$. Since pmcfg G has a nonterminating production r_h (see (40)), cfg G' has a production $\hat{r}_{hZ_1 \cdots Z_k} : R_h \to Z_1 \cdots Z_k$ such that

$$\begin{cases} Z_u = R_{h_u} & \text{if the root of } t_u \text{ is } \hat{r}_{h_u Z_{u1} \cdots Z_{uk_u}}, \\ Z_u = a_{h_u} & \text{if the root of } t_u \text{ is } a_{h_u}. \end{cases} \tag{44}$$

Hence if we take $t = \hat{r}_{hZ_1 \cdots Z_k}(t_1, \ldots, t_k)$ then t is a derivation tree of G' and

$$q_i[t] = q_i[\hat{r}_{hZ_1 \cdots Z_k}(t_1, \ldots, t_k)]$$

$$
\begin{aligned}
&\Rightarrow\ w_{i,1}q_{\eta(i,1)}[t_{\mu(i,1)}]w_{i,2}\cdots w_{i,n_i}q_{\eta(i,n_i)}[t_{\mu(i,n_i)}]w_{i,n_i+1} && \text{by (38)}\\
&\overset{*}{\Rightarrow}\ w_{i,1}\beta_{\mu(i,1)\eta(i,1)}w_{i,2}\cdots w_{i,n_i}\beta_{\mu(i,n_i)\eta(i,n_i)}w_{i,n_i+1} && \text{by (43)}\\
&=\ f_h^{[i]}[\bar{\beta}_1,\ldots,\bar{\beta}_k] && \text{by (41)}\\
&=\ \alpha_i && \text{by (42)}
\end{aligned}
$$

for $1 \leq i \leq d(X)$. That is, $q_i[t] \overset{*}{\Rightarrow} \alpha_i$ $(1 \leq i \leq d(X))$ and the property holds.

(*If part*) The only if part is shown by induction on the size of a derivation tree t of cfg G'. For the basis, consider a derivation tree of size one, that is, $t = a_h$ for some $a_h \in V_T$, and assume that $q_i[t] \overset{*}{\Rightarrow} \alpha_i$ for $1 \leq i \leq s$ and it derives no output for $i > s$. Then, there are rules $q_i[a_h] \to \alpha_i$ $(1 \leq i \leq s)$ and by the construction of rules of M in Step II, pmcfg G has a terminating production $r'_h : Y_0 \to f'_h$ with $d(Y_0) = s$ and $f_h'^{[i]} = \alpha_i$ for $1 \leq i \leq s$. Hence, $(\alpha_1,\ldots,\alpha_s) \in L_G(Y_0)$ and the property holds.

Assume that the property holds for every derivation tree whose size is d' or less, and consider a derivation tree $t = \hat{r}_{hZ_1\cdots Z_k}(t_1,\ldots,t_k)$ of size $d'+1$ such that

$$
\begin{aligned}
q_i[t] &\Rightarrow w_{i,1}q_{\eta(i,1)}[t_{\mu(i,1)}]w_{i,2}\cdots w_{i,n_i}q_{\eta(i,n_i)}[t_{\mu(i,n_i)}]w_{i,n_i+1} && (45)\\
&\overset{*}{\Rightarrow} w_{i,1}\beta_{\mu(i,1)\eta(i,1)}w_{i,2}\cdots w_{i,n_i}\beta_{\mu(i,n_i)\eta(i,n_i)}w_{i,n_i+1}\\
&= \alpha_i && (46)
\end{aligned}
$$

for $1 \leq i \leq s$. Let $\hat{r}_{h_u Z_{u1}\cdots Z_{uk_u}}$ (or a_{h_u} possibly) be the root of subtree t_u $(1 \leq u \leq k)$. To apply the inductive hypothesis to each subtree t_u $(1 \leq u \leq k)$, we first investigate the nonterminal on the left-hand side of r_{h_u} which is a corresponding production of pmcfg G. Since t is a derivation tree of cfg G' and the left-hand side of $\hat{r}_{h_u Z_{u1}\cdots Z_{uk_u}}$ is R_{h_u} (see 36), there is a production $\hat{r}_{hZ_1\cdots Z_k} : R_h \to Z_1\cdots Z_k$ such that (44) holds. By the construction of productions of G' in Step A, pmcfg G has a production

$$r_h : Y_0 \to f_h[Y_1,\ldots,Y_k] \qquad (47)$$

such that $Z_u \in \mathrm{RS}'(Y_u)$ holds for $1 \leq u \leq k$. By the definition of RS$'$ and (44), it follows that the left-hand side of r_{h_u} (or r'_{h_u}) is Y_u.

Next, consider the rules of transducer M which are used in (45). Apparently, the rules used are defined in Step I, and it follows that the

function f_h in (47) is defined as

$$f_h^{[i]}[\bar{x}_1, \ldots, \bar{x}_k] = w_{i,1} x_{\mu(i,1)\eta(i,1)} w_{i,2} \cdots w_{i,n_i} x_{\mu(i,n_i)\eta(i,n_i)} w_{i,n_i+1} \tag{48}$$

for $1 \leq i \leq d(Y_0) = s$ where $\bar{x}_u = (x_{u1}, \ldots, x_{ud(Y_u)})$ $(1 \leq u \leq k)$. Since pmcfg G satisfies the non-erasing condition, for every u $(1 \leq u \leq k)$ and v $(1 \leq v \leq d(Y_u))$, the variable x_{uv} appears at least once on the right-hand side of (48) for some i $(1 \leq i \leq s)$. Hence, $q_v[t_u]$ appears at least once on the right-hand side of (45) for some i $(1 \leq i \leq s)$, and it follows that $q_v[t_u] \stackrel{*}{\Rightarrow} \beta_{uv}$ holds for every u $(1 \leq u \leq k)$ and v $(1 \leq v \leq d(Y_u))$. Since the size of t_u $(1 \leq u \leq k)$ equals to d' or less, by the inductive hypothesis,

$$\bar{\beta}_u = (\beta_{u1}, \ldots, \beta_{ud(Y_u)}) \in L_G(Y_u) \tag{49}$$

for each u $(1 \leq u \leq k)$. (Remind that the root of t_u is $\hat{r}_{h_u Z_{u1} \cdots Z_{uk_u}}$ (or a_{h_u}) and the left-hand side of r_{h_u} (or r'_{h_u}) is Y_u.) Now, replacing $\bar{x}$'s with $\bar{\beta}$'s in (48), and by (46),

$$\begin{aligned} & f_h^{[i]}[\bar{\beta}_1, \ldots, \bar{\beta}_k] \\ &= w_{i,1} \beta_{\mu(i,1)\eta(i,1)} w_{i,2} \cdots w_{i,n_i} \beta_{\mu(i,n_i)\eta(i,n_i)} w_{i,n_i+1} \\ &= \alpha_i \end{aligned} \tag{50}$$

for $1 \leq i \leq d(Y_0)$. By (47),(49) and (50), $(\alpha_1, \ldots, \alpha_s) \in L_G(Y_0)$ and the property holds. Hence, Property 4.2 has proved. □

Theorem 4.2: $yL(\text{DFSTS}) = \text{PMCFL}$.

Proof. In Lemma 4.1, it has been shown that $yL(\text{DFSTS}) \subseteq \text{PMCFL}$, and hence it suffices to show that $\text{PMCFL} \subseteq yL(\text{DFSTS})$. We show that $L(G) = yL(M, G')$ for M and G' constructed as above.

If $w \in L_G(S)$, then there is a production of pmcfg

$$r_h : S \to f_h[Y_1, \ldots Y_k] \tag{51}$$

which is the last production applied to obtain w. By Property 4.2, there is a derivation tree t of G' such that the root is $\hat{r}_{hZ_1 \cdots Z_k} : R_h \to Z_1 \cdots Z_k$ $(Z_u \in \text{RS}'(Y_u), 1 \leq u \leq k)$ and $q_1[t] \stackrel{*}{\Rightarrow} w$ holds. Let $t' = \hat{r}_{\text{start}}(t)$ then, since $R_h \in \text{RS'}(S)$, t' is also a derivation tree of cfg G'. Hence, $w \in yL(M, G')$ holds by (39).

In a reverse way, we can prove that if $w \in yL(M, G')$ then $w \in L_G(S)$, and the theorem holds. □

4.1.3 Recognition of yL(DFSTS)

In the previous sections, we show that yL(DFSTS) equals to PMCFL. Since deterministic polynomial time recognition algorithm for PMCFL has been proposed[15], it can be concluded that yL(DFSTS) is in $\mathcal{P}$ of computational complexity. This result has been noted in an earlier paper Ref.[5] as a corollary of its main result, but the running-time required for recognition was not analyzed.

By combining the recognition algorithm for PMCFL[15] and the construction procedure described in Section 4.1.1, we obtain an effective procedure to recognize yL(DFSTS). In the rest of this section, we investigate the complexity of the recognition of yL(DFSTS). First, we review results on the recognition of PMCFL. Please refer the definition of degree of PMCFG in Section 2.1.

Lemma 4.3[15]**:** A pmcfl which is generated by pmcfg with degree e can be recognized in $O(|w|^{e+1})$-time where $|w|$ denotes the length of an input. □

Next we define the *degree* of a deterministic yT-transducer $M = (Q, \Sigma, \Delta, q_0, R)$. For $\sigma \in \Sigma$ and $1 \in Q$, let $n_{q,\sigma}$ denote the number of occurrences of variables in the right-hand side of a rule whose left-hand side is $q[\sigma(x_1, \ldots, x_n)]$. If no rule is defined for $q[\sigma(x_1, \ldots, x_n)]$, then $n_{q,\sigma} = 0$. Since the yT-transducer is deterministic, $n_{q,\sigma}$ can be defined uniquely. For instance, in Example 2.4, $n_{q_i,+} = 2$ and $n_{q_d,\cdot} = 4$. Define the *degree* of a symbol $\sigma \in \Sigma$ as $|Q| + \sum_{q \in Q} n_{q,\sigma}$. If the maximum degree among the symbol in Σ is e, then M is called a *yT-transducer with degree e*. An *fsts with degree e* is an fsts of which yT-transducer is with degree e.

The readers can easily verify that a deterministic fsts with degree e is translated into a pmcfg with degree e by using the construction described in Section 4.1.1. Hence the following theorem holds.

Theorem 4.4: The yield language generated by an fsts with degree e can be recognized in $O(|w|^{e+1})$-time where $|w|$ denotes the length of an input. □

4.2 Monadic FSTS

In previous sections, the class of yield languages generated by deterministic FSTS is shown to be in $\mathcal{P}$. In Ref.[21], it has been shown that there is an $\mathcal{NP}$-complete language in the class of yield languages generated by nondeterministic FSTS. In this section, we give an $\mathcal{NP}$-complete language in a more "restricted" class of languages, $yL(\text{NMFSTS}_2)$, the class of yield languages generated by nondeterministic monadic FSTS with state-bound 2 (this class is denoted as m-fsts$_2$ in Figure 2). First, a language called *Unary-3SAT*[17], which is $\mathcal{NP}$-complete, is reviewed, and then it is shown to belong to $yL(\text{NMFSTS}_2)$.

A *Unary-3CNF* is a (nonempty) 3CNF in which the subscripts of variables are represented in unary. A positive literal x_i is represented by $1^i\$$ in a Unary-3CNF. Similarly, a negative literal $\neg x_i$ is represented by $1^i\#$. For example, a 3CNF

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (x_3 \vee \neg x_1 \vee \neg x_2)$$

is represented by

$$1\$11\$111\# \wedge 111\$1\#11\#$$

in a Unary-3CNF. *Unary-3SAT* is the set of all satisfiable Unary-3CNF's. Clearly, Unary-3SAT is $\mathcal{NP}$-complete.

A nondeterministic monadic yT-fsts (M, G) with state-bound 2 which generates Unary-3SAT is defined as follows. First, define a cfg $G = (V_N, V_T, P, S)$ where $V_N = \{S, T, F\}$, $V_T = \{e\}$ and the productions in P are as follows:

$$\begin{array}{llll} r_{SS} : S \to S & & r_{Te} : T \to e \\ r_{ST} : S \to T & & r_{FT} : F \to T \\ r_{SF} : S \to F & & r_{FF} : F \to F \\ r_{TT} : T \to T & & r_{Fe} : F \to e \\ r_{TF} : T \to F. & & \end{array}$$

Note that G is a regular grammar, and hence this fsts is monadic. Let u be a derivation tree of G. Then u has a following form;

$$u \triangleq \underbrace{r_{SS}(\cdots(r_{SS}}_{m-1}(r_{Sp_1}(u'))) \cdots)$$

where

$$u' = r_{p_1p_2}(r_{p_2p_3}(\cdots(r_{p_ne}(e))\cdots))$$

and $p_i \in \{T, F\}$ for $1 \leq i \leq n$. The outer m ($m \geq 1$) symbols of u are the rules whose left-hand side is S, and the next n symbols are the rules whose left-hand side is T or F.

Next, a yT-transducer $M = (Q, \Sigma, \Delta, q_0, R)$ is constructed to transduce u into a Unary-3CNF E such that

- E has m clauses,
- there are at most n distinct variables $x_1, \ldots, x_n$ in E, and
- the value of E becomes true if values are assigned to the variables as

$$x_i = \begin{cases} \text{TRUE} & \text{if } p_i \text{ is } T \\ \text{FALSE} & \text{if } p_i \text{ is } F \end{cases} \qquad (1 \leq i \leq n). \tag{52}$$

Let $Q = \{q_0, q_c, q_t, q_a\}$, $\Sigma = \{r_{SS}, \ldots, r_{Fe}, e\}$ and $\Delta = \{1, \wedge, \$, \#\}$. Since there are many rules in R, we will use an abbreviated notation. For example, the following four rules

$$q_a[r_{Te}(x)] \to 1\$, \quad q_a[r_{Te}(x)] \to 1\#$$
$$q_a[r_{Fe}(x)] \to 1\$, \quad q_a[r_{Fe}(x)] \to 1\#$$

are abbreviated as "$q_a[r_{Te}(x)] = q_a[r_{Fe}(x)] \to 1\$$ or $1\#$". By using this notation, define R as following rules (R1) through (R9):

(R1) $q_0[r_{SS}(x)] \to q_c[x] \wedge q_0[x]$.

(R2) $q_c[r_{SS}(x)] \to q_c[x]$.

By the rules (R1) and (R2), M transduces $q_0[u]$ into

$$\underbrace{q_c[r_{Sp_1}(u')] \wedge \cdots \wedge q_c[r_{Sp_1}(u')] \wedge q_0[r_{Sp_1}(u')]}_{m}. \tag{53}$$

As we explain later, each $q_c[\cdots]$ and $q_0[\cdots]$ derives one clause. Hence m clauses will be derived from $q_0[u]$.

(R3) $q_0[r_{ST}(x)] = q_0[r_{SF}(x)] = q_c[r_{ST}(x)] = q_c[r_{SF}(x)] \to$
$q_t[x]q_a[x]q_a[x]$ or $q_a[x]q_t[x]q_a[x]$ or $q_a[x]q_a[x]q_t[x]$.

By these rules, each $q_c[r_{Sp_1}(u')]$ ($q_0[r_{Sp_1}(u')]$) in (53) derives one of $q_t[u']q_a[u']q_a[u']$, $q_a[u']q_t[u']q_a[u']$ or $q_a[u']q_a[u']q_t[u']$.

(R4) $q_t[r_{TT}(x)] = q_t[r_{TF}(x)] \rightarrow 1q_t[x]$ or $1\$$.

(R5) $q_t[r_{Te}(x)] \rightarrow 1\$$.

(R6) $q_t[r_{FT}(x)] = q_t[r_{FF}(x)] \rightarrow 1q_t[x]$ or $1\#$.

(R7) $q_t[r_{Fe}(x)] \rightarrow 1\#$.

Suppose that a derivation from $q_t[u']$ has been proceeded and the current configuration is $q_t[r_{p_ip_{i+1}}(\cdots)]$. Now, transducer M has two choices (see rules (R4) and (R6));

- generate 1 and continue a translation of subtree, or
- generate $1\$$ if p_i is T, $1\#$ if p_i is F and complete a translation.

If M completes a translation and p_i is T (resp. F), then $q_t[u']$ has derived $1^i\$$ (resp. $1^i\#$). Note that this is a literal x_i (resp. $\bar{x}_i$) which becomes "true" under the assignment (52).

(R8) $q_a[r_{TT}(x)] = q_a[r_{TF}(x)] = q_a[r_{FT}(x)] = q_a[r_{FF}(x)] \rightarrow 1q_a[x]$ or $1\$$ or $1\#$.

(R9) $q_a[r_{Te}(x)] = q_a[r_{Fe}(x)] \rightarrow 1\$$ or $1\#$.

These are similar to the rules (R4) through (R7); $q_a[u']$ derives some literal but it is not guaranteed to become "true".

Now, the readers can easily verify that this fsts has a state-bound 2, and that this fsts can derive an arbitrary satisfiable Unary-3CNF.

Theorem 4.5: Unary-3SAT is in $yL(\text{NMFSTS}_2)$. □

5 Conclusions

Computational complexities of the universal recognition problems have been investigated in this dissertation. It has been shown that the problems for PMCFG, NEPMCFG, m-PMCFG ($m \geq 1$), and PMCFG$_e$ ($e \geq 3$) are EXP-POLY time-complete, PSPACE-complete, $\mathcal{NP}$-complete, and $\mathcal{P}$-complete, respectively. Complexities of the problems for MCFG and its subclasses are almost identical to that for PMCFG, except that $\mathcal{NP}$-completeness holds for m-MCFG with $m \geq 2$ (see Figure 2). Based on these theoretical results, characteristics and relations among these subclasses are discussed. Furthermore, the relation between language acquisition and universal recognition is discussed in some detail. The variation of complexities of the universal recognition problems confirms our intuitive understanding, from theoretical aspects, that there is a trade-off relation between the generative power of grammars and the difficulty of acquisition of languages. Though the universal recognition problems are intractable for most subclasses of PMCFG, it is tractable for the class of pmcfg's (mcfg's) with bounded degree. It can be concluded that the class of pmcfg's (mcfg's) with bounded degree satisfies properties (i) through (iii) introduced in the introduction. Thereby, it is a favorable model to describe the syntax of natural languages. There may be other grammatical, or mathematical computational models of which generative (or computational) power equal to PMCFG. In fact, in the latter half of this dissertation, FSTS is shown to have the same generative power as PMCFG. Clarifying complexities of universal recognition problems for these models, and comparing the results to that for PMCFG will bring fruitful implications from both of theoretical and practical viewpoints.

In the latter half of the dissertation, the generative power of FSTS and its subclasses are investigated. It is an interesting result that deterministic FSTS and PMCFG, which were proposed from quite different viewpoints and developed independently, have the same generative power. Remark that we can define hierarchies in the class of languages generated by PMCFG (and hence deterministic FSTS) via dimension of PMCFG, and via state-bound of deterministic FSTS. It is interesting to clarify the relation between these two hierarchy. As for nondeterministic FSTS, it has been shown that there is a nondeterministic monadic fsts with

state-bound 2 which generates an $\mathcal{NP}$-complete language. Among the natural subclasses of nondeterministic FSTS, nondeterministic monadic FSTS with state-bound 2 is quite a small one. The fact that there is an $\mathcal{NP}$-complete language in the class of yield languages generated by nondeterministic monadic FSTS with state-bound 2 implies that nondeterminism brings an essential computational power in FSTS. It is notable that the relation between the class $\mathcal{P}$ and the class of yield languages generated by monadic FSTS with state-bound 1 remains unclear. Hence it will be interesting subject to clarify the relation among $\mathcal{P}$, the class of languages generated by PMCFG, and that of monadic FSTS with state-bound 1.

References

[1] Aho A.V.: "Indexed Grammars", J. Assoc. Comput. Machinary, **15**, pp.647–671 (1968).

[2] Barton G.E., Berwick R.C. and Ristad E.S.: "Computational Complexity and Natural Language", The MIT Press (1987).

[3] Chandra A. and Stockmeyer L.: "Alternation", Proc. of the 17th FOCS, pp.98–108 (1976).

[4] Earley J.: "An Efficient Context-Free Parsing Algorithm", Ph.D. dissertation, Dept. of Computer Science, Carnegie-Mellon University (1970).

[5] Engelfriet J.: "The Complexity of Languages Generated by Attribute Grammars", SIAM J. Comput., **15**, 1, pp.70–86 (Feb. 1986).

[6] Engelfriet J. and Heyker L.: "The String Generating Power of Context-Free Hypergraph Grammars", J. Comput. & Syst. Sci., **43**, pp.328–360 (1991).

[7] Engelfriet J., Rosenberg G. and Slutzki G.: "Tree Transducers, L Systems, and Two-Way Machines", J. Comput. & Syst. Sci., **20**, pp.150–202 (1980).

[8] Gazdar G., Klein E., Pullum G. and Sag I.: "Generalized Phrase Structure Grammar", Basil Blackwall (1985).

[9] Hopcroft J.E. and Ullman J.D.: "Introduction to Automata Theory, Languages, and Computation", Addison-Wesley (1979).

[10] Jones N.D. and Laaser W.T.: "Complete Problems for Deterministic Polynomial time", Theoretical Computer Science, **3**, 1, pp.105–118 (1976).

[11] Joshi A.K., Levy L. and Takahashi M.: "Tree Adjunct Grammars", J. Comput. & Syst. Sci., **10**, 1, pp.136–163 (1975).

[12] Karp R.M.: "Reducibility among Combinatorial Problems", Complexity of Computer Computations, pp.85–104, Plenum Press (1972).

[13] Kasami T., Seki H. and Fujii M.: "Generalized Context-Free Grammars, Multiple Context-Free Grammars and Head Grammars", Technical Report, Osaka University (1987), also in : Preprint of WG on Natural Language of IPSJ, **87-NL**-63-1 (Sept. 1987).

[14] Kasami T., Seki H. and Fujii M.: "Generalized Context-Free Grammars and Multiple Context-Free Grammars", Trans. IEICE, **J71-D-I**, 5, pp.758–765 (May 1988) (in Japanese).

[15] Kasami T., Seki H. and Fujii M.: "On the Membership Problem for Head Languages and Multiple Context-Free Languages", Trans. IEICE, **J71-D-I**, 6, pp. 935–941 (June 1988) (in Japanese).

[16] Matsumura T., Seki H., Fujii M. and Kasami T.: "On the Generative Power of Multiple Context-Free Grammars and Head Grammars", Trans. IEICE, **J73-D-I**, 5, pp.473–483 (May 1990) (in Japanese).

[17] Nakanishi R., Seki H. and Kasami T.: "On the Generative Capacity of Lexical-Functional Grammars", IEICE Trans. Inf. and Syst., **75-D**, 7, pp.509–516 (July 1992).

[18] Nishino T.: "Relating Attribute Grammars and Lexical-Functional Grammars", Research Report TDU-IS-7, Tokyo Denki University (June 1988) (also in Information Sciences, **66**, pp.1–22 (1992)).

[19] Pollard C.J.: "Generalized Phrase Structure Grammars, Head Grammars and Natural Language", Ph.D. dissertation, Stanford University (1984).

[20] Rounds W.C.: "Context-Free Grammars on Trees", Proc. of ACM Symp. on Theory of Computing, pp.143–148 (May 1969).

[21] Rounds W.C.: "Complexity of Recognition in Intermediate-Level Languages", IEEE 14th Annual Symp. on SWAT., pp.145–158 (Oct. 1973).

[22] Seki H., Matsumura T., Fujii M. and Kasami T.: "On Multiple Context-Free Grammars", Theoretical Computer Science, **88**, 2, pp.191–229 (Oct. 1991).

[23] Tanaka S. and Kasai T.: "The Emptiness Problem for Indexed Languages is Exponential Time Complete", Trans. IEICE, **68-D**, 10, pp.1727–1734 (Oct. 1985) (in Japanese).

[24] Thatcher J.W.: "Characterizing Derivation Trees of Context-Free Grammars through a Generalization of Finite Automata Theory", J. Comput. & Syst. Sci., **1**, pp.317–322 (Dec. 1967).

[25] Vijay-Shanker K., Weir D.J. and Joshi A.K.: "Tree Adjoining and Head Wrapping", Proc. 11th Intl. Conf. on Comput. Ling., pp.202–207 (1986).

[26] Vijay-Shanker K., Weir D.J. and Joshi A.K.: "Characterizing Structural Descriptions Produced by Various Grammatical Formalisms", Proc. of 25th Annual Meeting of Assoc. for Comput. Ling., pp.104–111 (1987).

[27] Weir D.J. : "Characterizing Mildly Context-Sensitive Grammar Formalisms", Ph.D. thesis, University of Pennsylvania (1988).

[28] Weir D.J.: "Linear Context-Free Rewriting Systems and Deterministic Tree-Walking Transducers", Proc. of 30th Annual Meeting of Assoc. for Comput. Ling. (June 1992).