Computational Intelligence, Volume 10, Number 4, 1994

THE COMPUTATIONAL COMPLEXITY OF THE UNIVERSAL RECOGNITION PROBLEM FOR PARALLEL MULTIPLE CONTEXT-FREE GRAMMARS

YUICHI KAJI[1], RYUICHI NAKANISHI, HIROYUKI SEKI[1]

Department of Information and Computer Sciences, Osaka University, Machikaneyama 1-3, Toyonaka, Osaka 560, Japan

TADAO KASAMI

Graduate School of Information Science, Nara Institute of Science and Technology, Takayama 8916-5, Ikoma, Nara 630-01, Japan

A number of grammatical formalisms have been proposed to describe the syntax of natural languages, and the universal recognition problems for some of those classes of grammars have been studied. A 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. In this paper, the computational complexities of the universal recognition problems for *parallel multiple context-free grammars*, *multiple context-free grammars*, and their subclasses are discussed.

Key words: universal recognition problem, parallel multiple context-free grammar, computational complexity, language acquisition.

1. INTRODUCTION

A number of grammatical formalisms have been introduced to describe the syntax of natural languages, and their mathematical properties have been extensively studied. Most of these studies mainly focused on the properties of the class of "languages" generated by those grammatical formalisms, but the properties of "grammars" themselves have not been studied so widely. In this paper, we discuss the computational complexities of the *universal recognition problems* for *parallel multiple context-free grammars*, *multiple context-free grammars* and their subclasses, and clarify characteristics of these grammars and relations between them.

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 G can generate w. 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) problems. A *fixed-language recognition (parsing) problem* for a language L is defined as follows: take a string w as an input, decide whether w belongs to L. The size of an input is $|w|$ only; a 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, the complexity of a fixed-language recognition problem is not an appropriate measure of syntactical complexity of interest. Fixed-language recognition is a problem for a language, while universal recognition is a one for a class of grammars.

The significance of universal recognition problems is described in Barton *et al.* (1987). Here we describe intuitively the relation between the "difficulty" of acquisition of a language and the universal recognition problem. Suppose that one learns a language from a description

[1]The authors are now with the Graduate School of Information Science, Nara Institute of Science and Technology.

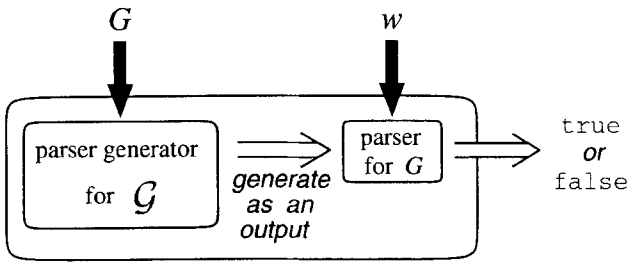


FIGURE 1. A model of acquisition of a language from a description of a grammar.

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 Fig. 1, using a parser generator for the class 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., $P_G \ll PG_{\mathcal{G}}$, the difficulty of acquisition of a language is estimated by the complexity of the universal recognition.

We emphasize that universal recognition is a problem for grammars. Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be classes of grammars and $\mathcal{L}_1$ and $\mathcal{L}_2$ be the classes of languages generated by grammars in $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively, and assume that the universal recognition problem for $\mathcal{G}_2$ is in a class $\mathcal{H}$ of computational complexity. Clearly $\mathcal{G}_1 \subseteq \mathcal{G}_2$ implies that the problem for $\mathcal{G}_1$ is also in $\mathcal{H}$ since every grammar G in $\mathcal{G}_1$ belongs to $\mathcal{G}_2$. On the other hand, $\mathcal{L}_1 \subseteq \mathcal{L}_2$ does not always imply that the universal recognition problem for $\mathcal{G}_1$ is in $\mathcal{H}$. For example, although the class of languages generated by generalized phrase structure grammars (GPSGs) (Gazdar *et al.* 1985) is included in the class of context-free languages (CFLs), and the universal recognition problem for context-free grammars (CFGs) is known to be solvable in deterministic polynomial time, the problem for GPSG is known to be EXP-POLY time-hard (Barton *et al.* 1987). Table 1 summarizes known results on the universal recognition problems for some classes of grammars, where RLFG (Nishino 1992) denotes a subclass of lexical functional grammars and IGs denotes indexed grammars (Aho 1968). As one can see from the table, the universal recognition 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

- It has enough generative power to describe the syntax of natural languages.
- The language acquisition from that class is tractable.

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

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

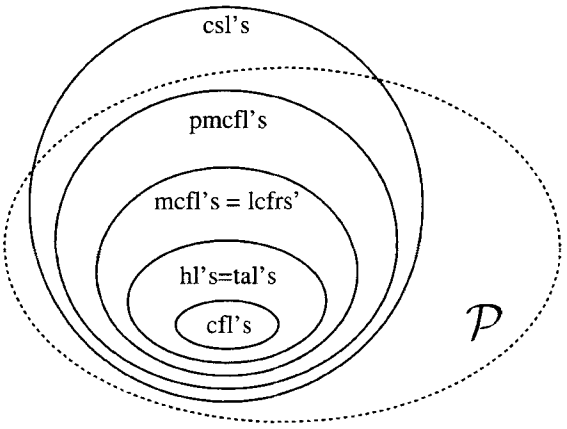
TABLE 1. Known Results.

Grammars	Complexity of Universal Recognition
CFGs	$\mathcal{P}$-complete (Jones and Laaser 1976)
GPSGs (Gazdar *et al.* 1985)	EXP-POLY time-hard (Barton *et al.* 1987)
RLFGs (Nishino 1992)	$\mathcal{NP}$-hard (Barton *et al.* 1987)
IGs (Aho 1968)	EXP-POLY time-complete (Tanaka and Kasai 1985)
CSGs	PSPACE-complete (Karp 1992)

Parallel multiple context-free grammars (*PMCFGs*) (Kasami *et al.* 1987) are a subclass of generalized context-free grammars introduced by Pollard (1984) in order to describe the syntax of natural languages, and *multiple context-free grammars* (*MCFGs*) (Kasami *et al.* 1987) are a subclass of PMCFGs. PMCFGs and MCFGs can be considered as a natural extension of CFGs. 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 1. The class of languages generated by PMCFGs and MCFGs are called *parallel multiple context-free languages* (*PMCFLs*) and *multiple context-free languages* (*MCFLs*), respectively. It was shown in Kasami *et al.* (1987) that MCFLs properly include CFLs, and are properly included in PMCFLs, which in turn are properly included in the class of context-sensitive languages (CSLs). It was also shown in Kasami *et al.* (1987) and Seki *et al.* (1991) that MCFLs are equal to the class of languages generated by the subsystem of linear context-free rewriting systems (LCFRs) (Vijay-Shanker *et al.* 1987) which deals with only strings, and it properly includes the class of tree-adjoining languages (TALs) (Joshi *et al.* 1975) and that of head languages (HLs) (Pollard 1984), which were also introduced to describe the syntax of natural languages (see Figure 1). It has been already shown (Seki *et al.* 1984) that PMCFLs (and hence MCFLs) are included in the class $\mathcal{P}$ of the computational complexity; i.e., the fixed-language recognition problem for any language generated by a PMCFG is solvable in deterministic polynomial time. To the authors' knowledge, no grammatical formalisms $\mathcal{G}$ have been introduced such that the class of languages generated by $\mathcal{G}$ properly includes PMCFLs and is properly included in $\mathcal{P}$. That is, PMCFGs have the strongest generative power among the known classes of grammars which define tractable classes of languages. This is one of the reasons why we are interested in the universal recognition problem for PMCFGs.

We also note that we can define some subclasses of PMCFGs and MCFGs 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 $(m+1)$-MCFGs properly includes the class of languages generated by m-MCFGs. The degree of a PMCFG G is defined as the maximum of the sizes of production rules of G (see Section 2 for its formal definition). Modified head grammars, which were shown to have the same generative power as head grammars, are a proper subclass of MCFGs with dimension 2 and degree 6. It is interesting to make clear the complexity of the universal recognition problem for PMCFGs, and to observe how the complexity varies when some restrictions are put on the grammars.

This paper is organized as follows. We give formal definitions of PMCFG and MCFG together with some examples in Section 2. In Section 3, we summarize our technical results on



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

FIGURE 2. Inclusion relations among some classes of languages.

the computational complexities of the universal recognition problems for PMCFGs, MCFGs and their subclasses, based on Kaji (1992) and Kaji *et al.* (1992*a*, 1992*b*), and discuss their implication in natural language processing. In the appendix, we present the main ideas behind some of the proofs of the results.

2. DEFINITIONS

A *parallel multiple context-free grammar* (*PMCFG*) is defined to be a 5-tuple $G = (N, T, F, P, S)$ which satisfies the following conditions (G1) through (G5) (Kasami *et al.* 1988; Seki 1991).

(G1) N is a finite set of *nonterminal symbols*, and a positive integer $d(A)$ is given for each nonterminal symbol $A \in N$. Define the *dimension of* G as $\max\{d(A) \mid A \in N\}$.

(G2) T is a finite set of *terminal symbols* which is disjoint with N.

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

$$\bar{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id_i(f)}) \qquad (1 \leq i \leq a(f))$$

denote the ith argument of 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 represented by a concatenation of some terminal strings in 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 $\alpha_{hk} \in T^*$ $(0 \leq k \leq v_h(f))$ and $z_{hk} \in X$ $(1 \leq k \leq v_h(f))$.

(G4) P is a finite set of *rewriting rules* (or shortly *rules*) of the form $A \to f[A_1, A_2, \ldots, A_{a(f)}]$ where $A, A_1, A_2, \ldots, A_{a(f)} \in N$, $f \in F$, $r(f) = d(A)$ 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 T. A rule with a function f such that $a(f) = 0$ is called a *terminating rule*. We write $A \to f$ for a terminating rule $A \to f[]$.

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

If all the functions of a PMCFG G satisfy the following condition (f2), then G is called a *multiple context-free grammar (MCFG)*.

(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 capacity of grammars (Kasami *et al.* 1987). Condition (f2) inhibits these copy operations.

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

(L1) If there is a terminating rule $A \to f$ such that $f = \bar{\alpha} \in (T^*)^{d(A)}$, then $\bar{\alpha} \in L_G(A)$.

(L2) If $A \to f[A_1, A_2, \ldots, A_{a(f)}] \in P$ and $\bar{\alpha}_i \in L_G(A_i)$ $(1 \leq i \leq a(f))$, then $f[\bar{\alpha}_1, \bar{\alpha}_2, \ldots, \bar{\alpha}_{a(f)}] \in L_G(A)$.

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. If G is an MCFG, $L(G)$ is called the *multiple context-free language (MCFL) generated by* G.

Example 1. Let $G_1 = (N, T, F, P, S)$ where $N = \{A, B, S\}(d(A) = d(B) = 2, d(S) = 1)$, $T = \{a, b, c, d\}$, $F = \{f, g_A, g_B, h\}$ and the rules in P be

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

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). 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. Let $G_2 = (N, T, F, P, S)$, $N = \{S\}$, $T = \{a\}$, $F = \{f, f_a\}$ and the rules in P be

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

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 Kasami *et al.* 1988).

Hereafter, we will define subclasses of PMCFGs and MCFGs. If all the functions f of a PMCFG (resp. MCFG) G satisfy the following condition (f3), then G is a *PMCFG with nonerasing condition* (resp. *MCFG with nonerasing condition*).

(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 nonerasing 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 (Vijay-Shanker *et al.* 1987), a subclass which deals only with tuples of strings. From a grammatical viewpoint, PMCFGs (MCFGs) with nonerasing condition are a proper subclass of PMCFGs (MCFGs). But it has been already shown that condition (f3) does not weaken the generative power of PMCFGs (MCFGs) (Seki *et al.* 1991).

Lemma 3. For a given PMCFG (resp. MCFG), we can construct a weakly equivalent PMCFG (resp. MCFG) with nonerasing condition.

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

The last subclass we introduce is a PMCFG with bounded degree. (Refer to expression (2) of (f1)). The *degree* of a function f is defined as $\sum_{h=1}^{r(f)}(v_h(f)+1)$, which equals 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. Note that in these classes of grammars, a degree e is treated as a constant. For $e \geq 1$, the class of PMCFGs with degree e is a proper subclass of PMCFGs with degree $e+1$, 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, the class of PMCFGs with degree e is a subclass of e-PMCFGs and hence the dimension of each grammar in that class can be treated as a constant. But in the class of m-PMCFGs, 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 1 has dimension 2 and degree 6 (rule r_0 has the maximum degree) and the PMCFG G_2 in Example 2 has dimension 1 and degree 3.

Lastly, the size of a PMCFG $G = (N, T, F, P, S)$ is defined. 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 $|N|$, $|T|$, $\sum_{f\in F}|f|$ and $\sum_{r\in P}|r|$.

3. RESULTS AND DISCUSSION

In this section, we present our results on the computational complexities for PMCFGs, MCFGs, and their subclasses. The subclasses we investigated are

- PMCFGs (with no constraint)
- PMCFGs with nonerasing condition
- m-PMCFGs
- PMCFGs with degree e

and their corresponding classes of MCFGs. Our results are summarized in Table 2. The main ideas behind the proofs are summarized in the Appendix.

The problems for PMCFGs and MCFGs without any constraint are both EXP-POLY time-complete, where EXP-POLY time is the class of complexity which consists of the problems solvable in deterministic $c^{p(n)}$-time for a constant c and a polynomial $p(n)$. If we restrict the input grammars to those which satisfy the nonerasing condition, the complexities drop into PSPACE-complete, whereas the nonerasing condition does not weaken the generative power of PMCFGs or MCFGs (see Lemma 3).

The difference between these complexities can be understood intuitively as follows: Let's go back to the language acquisition example in the introduction. 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 PMCFGs, an omission of a word corresponds to an erasing of a variable on the right-hand side of expression (1). Thereby, it is quite suitable to our intuition that the universal recognition problem for PMCFGs with no constraint is more difficult than that for PMCFGs with nonerasing condition. As mentioned above, nonerasing condition does not weaken the generative power of grammars. Now, one may think to transform a PMCFG G that violates the nonerasing condition into another weakly equivalent PMCFG G' that satisfies the nonerasing condition, and apply a polynomial space universal recognition algorithm to G'. But elimination of rules that violate the nonerasing condition may make the grammar size exponentially larger than the original one (Kasami *et al.* 1988; Seki *et al.* 1991), and hence the complexity of the universal recognition problem cannot be reduced by such a transformation.

The problems for m-PMCFGs ($m \geq 1$) and for m-MCFGs ($m \geq 2$) are both $\mathcal{NP}$-complete. Note that the problem for 1-MCFGs is not $\mathcal{NP}$-complete since 1-MCFGs equal CFGs, and hence they are solvable in deterministic polynomial time (see Table 1). The problems for PMCFGs and MCFGs with degree e are solvable in $O(|G||w|^{e+1})$-time and $O(|G||w|^{e})$-time, respectively, and they are $\mathcal{P}$-complete if $e \geq 3$.

Generally speaking, there is not much difference between the complexities of the problems for PMCFGs and for MCFGs with the same restriction (compare PMCFGs and MCFGs 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 we described in Section 1, 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 PMCFGs (or MCFGs) with fixed degree. The remaining question is, "Do the PMCFGs 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 (HGs) (Pollard 1984) and tree-adjoining grammars (TAGs) (Joshi *et al.* 1975) have been widely accepted as grammatical formalisms to describe the syntax of natural

TABLE 2. Complexities of the Universal Recognition Problems.

	PMCFGs	MCFGs
(with no constraint)	EXP-POLY time-complete	
with nonerasing 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 $O(\|G\|\|w\|^{e+1})$-time	solvable in $O(\|G\|\|w\|^{e})$-time

languages. It has already been shown that the class of languages generated by HGs and TAGs coincide, and it is a proper subclass of languages generated by the MCFGs with dimension 2 and degree 6 (Seki *et al.* 1991). 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, the generative capacity and the difficulty of acquisition are ambivalent.

ACKNOWLEDGMENT

We'd like to thank three anonymous referees for their helpful advice and encouragements. We also thank Dr. Giorgio Satta for his valuable information about related works and known results. We express our great appreciation to the chair, Professor Aravind K. Joshi for the invitation to the workshop and his kind support.

This paper is partially based on Kaji (1992), Kaji (1992*a*, 1992*b*).

REFERENCES

AHO, A. V. 1968. Indexed grammars. J. Assoc. Comput., **15**, 647–671.

BARTON, G. E., R. C. BERWICK, and E. S. RISTAD. 1987. Computational complexity and natural language. The MIT Press.

CHANDRA, A. and L. STOCKMEYER. 1976. Alternation. *In* Proc. 17th FOCS, pp. 98–108.

GAZDAR, G., E. KLEIN, G. PULLUM, and I. SAG. 1985. Generalized Phrase Structure Grammar. Basil Blackwell.

JONES, N. D., and W. T. LAASER. 1976. Complete problems for deterministic polynomial time. Theoretical Computer Sci., **3**(1):105–118.

JOSHI, A. K., L. LEVY, and M. TAKAHASHI. 1975. Tree adjunct grammars. J. Computer System Sci., **10**(1):136–163.

KAJI, Y. 1992. The computational complexity of the universal recognition problem for parallel multiple context-free grammars. Master's thesis, Osaka University.

KAJI, Y., R. NAKANISHI, H. SEKI, and T. KASAMI. 1992*a*. The universal recognition problems for multiple context-free grammars and for linear context-free rewriting systems. IEICE Trans. Information and Systems, **E75-D**(1):78–88.

KAJI, Y., R. NAKANISHI, H. SEKI, and T. KASAMI. 1992*b*. The universal recognition problems for parallel multipl context-free grammars and for their subclasses. IEICE Trans. Information and Systems, **E75-D**(7):499–508

KARP, R. M. 1972. Reducibility among combinatorial problems. *In* Complexity of Computer Computations Plenum Press, pp. 85–104.

KASAMI, T., H. SEKI, and M. FUJII. 1987. Generalized context-free grammars, multiple context-free grammars and head grammars. Technical Report, Osaka University. Also in Preprint of WG on Natural Language of IPSJ, **87-NL**-63-1.

KASAMI, T., H. SEKI, and M. FUJII. 1988. Generalized context-free grammars and multiple context-free grammars. Trans. IEICE, **J71-D-I**(5):758–765 (Japanese).

NISHINO, T. 1992. Relating attribute grammars and lexical-functional grammars. Information Sciences, **66**, 1–22.

POLLARD, C. J. 1984. Generalized phrase structure grammars, head grammars and natural language. Ph.D. dissertation, Stanford University.

SEKI, H., T. MATSUMURA, M. FUJII, and T. KASAMI. 1991. On multiple context-free grammars. Theoretical Computer Science, **88**(2):191–229.

TANAKA, S., and T. KASAI. 1985. The emptiness problem for indexed languages is exponential time complete. Trans. IEICE, **68-D**(10):1727–1734 (Japanese).

VIJAY-SHANKER, K., D. J. WEIR, and A. K. JOSHI. 1987. Characterizing structural descriptions produced by various grammatical formalisms. *In* Proc. 25th meeting of Assoc. Comput. Ling., pp. 104–111.

A. APPENDIX

Let C be a class of complexity. If we show that

- The universal recognition problem for PMCFGs belongs to C, and
- The problem for MCFGs is C-hard,

then we can conclude that the problems for PMCFGs and for MCFGs are both C-complete.

Since the proofs are lengthy and we cannot introduce all of them in this paper, we only introduce reduction algorithms which are used to prove the hardness results in Table 2. The correctness of each algorithm and the proofs of containment part are omitted here (see Kaji 1992 and Kaji *et al.* 1992 for details).

A.1. EXP-POLY time-completeness

It is known that a language belongs to the class of EXP-POLY time if and only if it is accepted by a polynomial space-bounded alternating Turing machine (ATM) (Chandra and Stockmeyer 1976). We will introduce an algorithm which reduces an acceptance problem of a polynomial space-bounded ATM into the universal recognition problem for MCFGs.

An ATM M is defined by a 9-tuple $M = (Q, \Sigma, \Gamma, B, \delta, q_S, Q_F, Q_U, Q_E)$ where each component denotes a set of states, a set of input symbols, a set of tape symbols, a blank symbol, a transition function, an initial state, a set of final states, a set of universal states, and a set of existential states, respectively. It is assumed that Q_F, Q_U, and Q_E are disjoint and $Q = Q_F \cup Q_U \cup Q_E$. 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 tape head. 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) through (c).

(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) Let $I = (q, k, \alpha)$ be an ID with $q \in Q_U$. If every I' satisfying $I \vdash I'$ belong 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).

The ATM M *accepts* a string t if and only if its initial ID $(q_S, 1, t)$ belongs to ACC(M).

Now, we introduce the idea behind the algorithm. Fix a polynomial p and a $p(n)$ space-bounded ATM M. For convenience, we assume $\Gamma = \{c_1, \ldots, c_{|\Gamma|}\}$ without loss of generality. Define a pairing function as $\langle k, c_j \rangle = (k-1)|\Gamma| + j$ $(1 \leq k \leq p(n), 1 \leq j \leq |\Gamma|)$.

To a string on the tape $\alpha \in \Gamma^{p(n)}$, we relate the following vector $\bar{v}$ which have $p(n)|\Gamma|$ components; for $1 \leq k \leq p(n)$, if the kth symbol of α is c then the $\langle k, c \rangle$th component of $\bar{v}$ is ε, and $\langle k, c' \rangle$th component $\bar{v}$ is 1^+ (a string consists of "1") for $c' \neq c$, for $1 \leq k \leq p(n)$. For example, let $\Gamma = \{c_1, c_2, c_3\}$ and $p(n) = 3$. A string $c_2c_1c_3$ is represented by a $3 \times 3 = 9$-vector $(1, \varepsilon, 1, \varepsilon, 1, 1, 1, 1, \varepsilon)$.

For $q \in Q$ and k $(1 \leq k \leq p(n))$, we will introduce a nonterminal A_{qk} with $d(A_{qk}) = p(n)|\Gamma|$, and construct rules to satisfy the following proposition.

Proposition A.1. $(q, k, \alpha) \in \mathrm{ACC}(M)$ iff $\exists \bar{v} \in L_G(A_{qk})$ which represents α.

Algorithm A.1.

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

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

1. Let $T = \{1\}$ and $w = \varepsilon$.
2. Let $N = \{S\} \cup \{A_{qk} \mid q \in Q, 1 \leq k \leq p(n)\}$ where $d(A_{qk}) = p(n)|\Gamma|$ $(q \in Q, 1 \leq k \leq p(n))$.
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 rule corresponds to the initial ID $(q_S, 1, t_1 \cdots t_n B \cdots B)$.
4. For each $q \in Q_F$, add terminating rule $A_{qk} \to (\varepsilon, \varepsilon, \ldots, \varepsilon)$ to P for each k $(1 \leq k \leq p(n))$. Remind the condition (a) of the definition of ACC(M) and Proposition A.1.
5. For $q \in Q$ and $c \in \Gamma$, $\delta(q, c)$ can be written as
$$\delta(q, c) = \{(q_i', c_i', R) \mid 1 \leq i \leq n_R\} \cup \{(q_j'', c_j'', L) \mid 1 \leq j \leq n_L\}.$$
For $(q_i', c_i', R) \in \delta(q, c)$, a function $f_{cc_i'k}$ is defined for each k $(1 \leq k < p(n))$ as follows.
$$\begin{aligned} f_{cc_i'k}^{\langle r, b \rangle}[\bar{x}] &= x_{\langle r, b \rangle} \qquad (r \neq k, b \in \Gamma) \\ f_{cc_i'k}^{\langle k, c \rangle}[\bar{x}] &= x_{\langle k, c_i' \rangle} \\ f_{cc_i'k}^{\langle k, b \rangle}[\bar{x}] &= 1 \qquad (b \neq c). \end{aligned}$$

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

$$\begin{aligned} g_{cc_j''k}^{\langle r,b\rangle}[\bar{x}] &= x_{\langle r,b\rangle} \qquad (r \neq k, b \in \Gamma) \\ g_{cc_j''k}^{\langle k,c\rangle}[\bar{x}] &= x_{\langle k,c_j''\rangle} \\ g_{cc_j''k}^{\langle k,b\rangle}[\bar{x}] &= 1 \qquad (b \neq c). \end{aligned}$$

If $q \in Q_E$, then let $F = F \cup \{f_{cc_i'k} \mid 1 \leq i \leq n_R, 1 \leq k \leq p(n)\} \cup \{g_{cc_j'k} \mid 1 \leq j \leq n_L, 1 \leq k \leq p(n)\}$ and add the following rules into P;

$$\begin{aligned} A_{qk} &\rightarrow f_{cc_i'k}[A_{q_i'k+1}] \qquad (1 \leq i \leq n_R, 1 \leq k < p(n)) \\ A_{qk} &\rightarrow g_{cc_j''k}[A_{q_j''k-1}] \qquad (1 \leq j \leq n_L, 1 < k \leq p(n)). \end{aligned} \tag{3}$$

The role of these productions can be understood by the following example. Let

$$\begin{aligned} I_1 &= (q, k, b_1 b_2 \cdots b_{k-1} c b_{k+1} \cdots b_{p(n)}), \\ I_2 &= (q_1', k+1, b_1 b_2 \cdots b_{k-1} c_1' b_{k+1} \cdots b_{p(n)}). \end{aligned}$$

If $I_2 \in \mathrm{ACC}(M)$ then $I_1 \in \mathrm{ACC}(M)$. If we assume that Proposition A.1 holds for I_2, then there is a vector $\bar{v}_{I_2} \in L_G(A_{q_1'k+1})$ such that $\langle k, c_1'\rangle$th component and $\langle j, b_j\rangle$th components $(j \neq k)$ are ε's. By substituting $\bar{v}_{I_2}$ for $A_{q_1'k+1}$ on the right-hand side of (3), we obtain $\bar{v}_{I_1} \in L_G(A_{qk})$ such that $\langle k, c\rangle$th components and $\langle j, b_j\rangle$th components $(j \neq k)$ are ε's, which implies the right-hand side of Proposition A.1.

If $q \in Q_U$, then define h_{ck} as

$$\begin{aligned} &h_{ck}[\bar{y}_1, \ldots, \bar{y}_{n_R}, \bar{z}_1, \ldots, \bar{z}_{n_L}] \\ &\quad = \mathrm{CONCAT}_{n_R+n_L}^{p(n)|\Gamma|}[f_{cc_1'k}[\bar{y}_1], \ldots, f_{cc_{n_R}'k}[\bar{y}_{n_R}], g_{cc_1''k}[\bar{z}_1], \ldots, g_{cc_{n_L}''k}[\bar{z}_{n_L}]] \end{aligned}$$

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

$$\begin{aligned} &\mathrm{CONCAT}_r^s[(x_{11}, x_{12}, \ldots, x_{1s}), \ldots, (x_{r1}, x_{r2}, \ldots, x_{rs})] \\ &\qquad = (x_{11} x_{21} \cdots x_{r1}, \ldots, x_{1s} x_{2s} \cdots x_{rs}). \end{aligned}$$

Add h_{ck} to F and add

$$A_{qk} \rightarrow h_{ck}[A_{q_1'k+1}, \ldots, A_{q_{n_R}'k+1}, A_{q_1''k-1}, \ldots, A_{q_{n_L}''k-1}] \tag{4}$$

to P for $1 \leq k \leq p(n)$. Note that the right-hand side of this rule is a componentwise concatenation of the arguments.

For the correctness of this algorithm, see Lemmas 4 and 5 of Kaji *et al.* (1992*a*). For the proof of containment in EXP-POLY time, see Lemma 3 of Kaji *et al.* (1992*a*).

A.2. PSPACE-completeness

By modifying Algorithm A.1, we can obtain an algorithm which reduces the acceptance problem of a nondeterministic polynomial space-bounded Turing machine into the universal recognition problem for MCFGs with nonerasing condition. We only note the key point of the modification.

- Instead of an ATM, a nondeterministic polynomial space-bounded Turing machine is considered. Hence we don't have to deal with universal states.
- Remember that MCFG G constructed in Algorithm A.1 does not satisfy condition (f3). For the MCFG constructed here to satisfy the condition, an extra component $f_{cc'k}^{p(n)|\Gamma|+1}$ is introduced and its value 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 $f_{cc'k}^{h}$ for any $1 \leq h \leq p(n)|\Gamma|$. Similarly, the extra component $g_{cc''k}^{p(n)|\Gamma|+1}$ is introduced for each $g_{cc''k}$.
- The extra component derives, as it stands, some string whose length can increase in proportion to the length of the derivation. To inhibit this, the roles of ε's and 1's are interchanged.

There are a few small modifications which are omitted here. For a full description of the algorithm, see Algorithm 3 in Kaji *et al.* (1992*a*).

We remark that the construction here does not work in the case of an ATM. In the case of an ATM, we have realized the ACC(M) condition for universal states by a componentwise concatenation of nonterminals (see (4) in the previous section). This technique was effective since a concatenation of ε's is also an ε. But now we have exchanged 1 and ε, which violates this concatenation trick for universal states.

The proof of containment in PSPACE is shown in Lemma 6 in Kaji *et al.* (1992*a*).

A.3. $\mathcal{NP}$-completeness

We introduce a reduction algorithm from 3SAT (the satisfiability problem of 3-conjunctive normal form Boolean expressions) to the universal recognition problem for 2-MCFGs. Since any 2-MCFG is also an m-MCFG ($m \geq 2$), it can be concluded that the problem for MCFGs with fixed dimension is $\mathcal{NP}$-hard.

Algorithm A.2

input: 3-CNF Boolean expression E.
output: 2-MCFG $G = (N, T, F, P, S)$ 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 ($1 \leq i \leq q$, $1 \leq j \leq 3$).

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.
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 $\neg p_i$.
3. Let $T = \{\$, \#, 1\}$ and $N = \{S, X, Y, A\}$, where $d(X) = d(A) = 2$, $d(Y) = d(S) = 1$.
4. Define f as follows and let $F = \{f\}$.

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

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 following:

(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 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.)

5. The rewriting rules are defined as

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

6. Define the input string w as

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

For the correctness of this algorithm, see Lemma 4 of Kaji *et al.* (1992*b*). For the proof of containment in $\mathcal{NP}$, see Lemma 3 of Kaji *et al.* (1992*b*).

A.4. $\mathcal{P}$-completeness

The $\mathcal{P}$-hardness of the universal recognition problem for MCFGs with fixed degree is an obvious consequence of the result that the problem for CFGs is $\mathcal{P}$-hard. For the proof of containment in $\mathcal{P}$, see Lemma 6 of Kaji *et al.* (1992*b*).