PARALLEL MULTIPLE CONTEXT-FREE GRAMMARS, FINITE-STATE TRANSLATION SYSTEMS, AND POLYNOMIAL-TIME RECOGNIZABLE SUBCLASSES OF LEXICAL-FUNCTIONAL GRAMMARS

Hiroyuki Seki †‡ Ryuichi Nakanishi † Yuichi Kaji †
Sachiko Ando † Tadao Kasami †‡

† Department of Information and Computer Sciences, Faculty of Engineering Science, Osaka University
1-1 Machikaneyama, Toyonaka, Osaka 560, Japan

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

Internet: seki@ics.es.osaka-u.ac.jp

Abstract

A number of grammatical formalisms were introduced to define the syntax of natural languages. Among them are parallel multiple context-free grammars (pmcfg's) and lexical-functional grammars (lfg's). Pmcfg's and their subclass called multiple context-free grammars (mcfg's) are natural extensions of cfg's, and pmcfg's are known to be recognizable in polynomial time. Some subclasses of lfg's have been proposed, but they were shown to generate an $\mathcal{NP}$-complete language. Finite state translation systems (fts') were introduced as a computational model of transformational grammars. In this paper, three subclasses of lfg's called nc-lfg's, dc-lfg's and fc-lfg's are introduced and the generative capacities of the above mentioned grammatical formalisms are investigated. First, we show that the generative capacity of fts' is equal to that of nc-lfg's. As relations among subclasses of those formalisms, it is shown that the generative capacities of deterministic fts', dc-lfg's, and pmcfg's are equal to each other, and the generative capacity of fc-lfg's is equal to that of mcfg's. It is also shown that at least one $\mathcal{NP}$-complete language is generated by fts'. Consequently, deterministic fts', dc-lfg's and fc-lfg's can be recognized in polynomial time. However, fts' (and nc-lfg's) cannot, if $\mathcal{P} \neq \mathcal{NP}$.

1 Introduction

A number of grammatical formalisms such as lexical-functional grammars (Kaplan 1982), head grammars (Pollard 1984) and tree adjoining grammars (Joshi 1975)(Vijay-Shanker 1987) were introduced to define the syntax of natural languages. On the other hand, there has been much effort to propose well-defined computational models of transformational grammars. One of these is the one to extend devices which operate on strings, such as generalized sequential machines (gsm's) to devices which operate on trees. It is fundamentally significant to clarify the generative capacities of such grammars and devices.

Parallel multiple context-free grammars (*pmcfg*'s) and *multiple context-free grammars* (*mcfg*'s) were introduced in (Kasami 1988a)(Seki 1991) as natural extensions of cfg's. The subsystem of *linear context-free rewriting systems* (*lcfrs'*) (Vijay-Shanker 1987) which deals with only strings is the same formalism as mcfg's. The class of cfl's is properly included in the class of languages generated by pmcfg's, which in turn is properly included in the one generated by mcfg's. The class of languages generated by pmcfg's is properly included in that of context-sensitive languages (Kasami 1988a). Pmcfg's have been shown to be recognized in polynomial time (Kasami 1988b)(Seki 1991).

A *tree transducer* (Rounds 1969) takes a tree as 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, it transforms an input tree into an output tree in a top-down way. A *finite state translation system* (*fts*) is a tree transducer with its input domain being the set of derivation trees of a cfg (Rounds 1969)(Thatcher 1967). A number of equivalence relations between the classes of yield languages generated by fts' and other computational models have been established (Engelfriet 1991)(Engelfriet 1980)(Weir 1992). Especially, it has been shown that the class of yield languages generated by finite-copying fts' equals to the class of languages generated by lcfrs' (Weir 1992), hence by mcfg's.

In *lexical-functional grammars* (*lfg*'s) (Kaplan 1982), associated with each node v of a derivation tree is a finite set F of pairs of attribute names and their values. F is called the *f-structure* of v. An lfg G consists of a cfg G_0 called the *underlying cfg* of G and a finite set P_{fs} of equations called *functional schemata* which specify constraints between the f-structures of nodes in a derivation tree. Functional schemata are attached to symbols in productions of G_0. It has been shown in (Nakanishi 1992) that the class of languages generated by lfg's is equal to that of recursively enumerable languages even though the underlying cfg's are restricted to regular grammars. In (Gazdar 1985)(Kaplan 1982)(Nishino 1991), subclasses of lfg's were proposed in order to guarantee the recursiveness (and/or the efficient recognition) of languages generated by lfg's. However, these classes were shown to generate an $\mathcal{NP}$-complete language (Nakanishi 1992).

In this paper, three subclasses of lfg's called *nc-lfg's*, *dc-lfg's* and *fc-lfg's* are proposed, two of which can be recognized in polynomial time. Moreover, this paper clarifies the relations among the generative capacities of pmcfg's, fts' and these subclasses of lfg's.

In nc-lfg's, a functional schema either specifies the value of a specific attribute, say *atr*, immediately ($\uparrow atr = val$) or specifies that the value of a specific attribute of a node v is equal to the whole f-structure of a child node of v ($\uparrow atr =\downarrow$).

An nc-lfg is called a *dc-lfg* if each pair of rules $p_1 : A \rightarrow \alpha_1$ and $p_2 : A \rightarrow \alpha_2$ whose left-hand sides are the same is inconsistent in the sense that there exists no f-structure that locally satisfies both of the functional schemata of p_1 and those of p_2. Intuitively, in a dc-lfg G, for each pair (t_1, t_2) of derivation trees in G, if the f-structure and nonterminal of the root of t_1 are the same as those of t_2, then t_1 and t_2 derive the same terminal string.

Let G be an nc-lfg. A multiset M of nonterminals of G is called an *SPN multiset* in G if the following condition holds:

> Let $M = \{\{A_1, A_2, \cdots, A_n\}\}$ be a multiset of nonterminals where different A_i's are not always distinct. There exist a derivation tree t and a subset of nodes $V = \{v_1, v_2, \cdots, v_n\}$ of t such that the label of v_i is A_i $(1 \leq i \leq n)$ and the f-structures of v_i's are the same with each other by functional schemata of G.

If the number of SPN multisets in G is finite, then G is called an *fc-lfg*.

Our main result is that the generative capacity of nc-lfg's is equal to that of fts'. As relations among proper subclasses of the above mentioned formalisms, it is shown that the generative capacities of dc-lfg's, deterministic fts' and pmcfg's are equal to each other, and the generative capacity of fc-lfg's is equal to that of mcfg's. It is also shown that a (nondeterministic) fts generates an $\mathcal{NP}$-complete language.

2 Parallel Multiple Context-Free Grammars

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 1988a)(Seki 1991).

(G1) N is a finite set of *nonterminal symbols*. A positive integer $d(A)$ is given for each nonterminal symbol $A \in N$.

(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. For a positive integer d, let $(T^*)^d$ denote the set of all the d-tuples of strings over T. For each $f \in F$ with arity $a(f)$, positive integers $r(f)$ and $d_i(f)$ $(1 \leq i \leq a(f))$ 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

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

denote the ith argument of f for $1 \leq i \leq a(f)$.

(f1) For $1 \leq h \leq r(f)$, the hth component of f, denoted by $f^{[h]}$, is defined as;

$$f^{[h]}[\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_{a(f)}] = \alpha_{h,0} x_{\mu(h,0)\eta(h,0)} \alpha_{h,1} \cdots \alpha_{h,n_h-1} x_{\mu(h,n_h-1)\eta(h,n_h-1)} \alpha_{h,n_h} \quad (2.1)$$

where $\alpha_{h,k} \in T^*$ for $0 \leq k \leq n_h$, $1 \leq \mu(h,j) \leq a(f)$ and $1 \leq \eta(h,j) \leq d_{\mu(h,j)}(f)$ for $0 \leq j \leq n_h - 1$.

(G4) P is a finite set of *productions* of the form $A \rightarrow f[A_1, A_2, \ldots, A_{a(f)}]$ where $A, A_1, A_2, \ldots, A_{a(f)} \in N$, $f \in F$, $r(f) = d(A)$ and $d_i(f) = d(A_i)$ $(1 \leq i \leq a(f))$. If $a(f) = 0$, i.e., $f \in (T^*)^{r(f)}$, the production is called a *terminating production*, otherwise it is called a *nonterminating production*.

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

If all the functions of a pmcfg G satisfy the following Right Linearity condition, then G is called a *multiple context-free grammar* (*mcfg*).

[Right Linearity] For each x_{ij}, the total number of occurrences of x_{ij} in the right-hand sides of (2.1) from $h = 1$ through $r(f)$ is at most one.

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

(L1) If a terminating production $A \rightarrow \bar{\alpha}$ is in P, 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 $f[\bar{\alpha}_1, \bar{\alpha}_2, \ldots, \bar{\alpha}_{a(f)}] \in L_G(A)$.

Define $L(G) \triangleq L_G(S)$. $L(G)$ 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 2.1(Kasami 1988a): Let $G_{EX1} = (N, T, F, P, S), N = \{S\}, T = \{a\}, F = \{f_a, f\}, P = \{r_1 : S \to f_a,\ r_0 : S \to f[S]\}$, where $f_a = a, f[(x)] = xx$. G_{EX1} is a pmcfg but is not an mcfg since the function f does not satisfy Right Linearity. The language generated by G_{EX1} is $\{a^{2^n} \mid n \geq 0\}$, which cannot be generated by any mcfg (see Lemma 6 of (Kasami 1988a)). $\Box$

The empty string is denoted by ε.

Example 2.2: Let $G_{EX2} = (N, T, F, P, S)$ be a pmcfg, where $N = \{S, A\}$, $T = \{a, b\}$, $F = \{g[(x_1, x_2)] = x_1 x_2,\ f_a[(x_1, x_2)] = (x_1 a, x_2 a),\ f_b[(x_1, x_2)] = (x_1 b, x_2 b),\ f_\varepsilon = (\varepsilon, \varepsilon)\}$, and, $P = \{p_0 : S \to g[A],\ p_1 : A \to f_a[A],\ p_2 : A \to f_b[A],\ p_3 : A \to f_\varepsilon\}$. Note that G_{EX2} is an mcfg. $L(G_{EX2}) = \{ww \mid w \in \{a, b\}^*\}$. $\Box$

Lemma 2.1(Kasami 1988b)(Seki 1991): Let G be a pmcfg. For a given string w, it is decidable whether $w \in L(G)$ or not in time polynomial of $|w|$, where $|w|$ denotes the length of w. $\Box$

3 Finite State Translation Systems

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

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

Hereafter, a term in $\mathcal{T}_\Sigma(X)$ is also called a *tree*, and we use terminology of trees such as subtree, node and so on.

Let $G = (N, T, P, S)$ be a context-free grammar (cfg) where N, 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 in cfg G* is a term defined as follows.

(T1) Every $a \in T$ is a derivation tree in G.

(T2) Assume that there are a production $p : A \to X_1 \cdots X_n$ ($A \in N, X_1, \ldots, X_n \in N \cup T$) in P and n derivation trees $t_1, \ldots t_n$ whose roots are labeled with $p_1, \ldots, p_n$, respectively, and

- if $X_i \in N$, then p_i is a production $X_i \to \cdots$, whose left-hand side is X_i, and
- if $X_i \in T$, then $p_i = t_i = X_i$.

Then $p(t_1, \ldots, t_n)$ is a derivation tree in G.

(T3) There are no other derivation trees.

Let $\mathcal{R}(G)$ be the set of derivation trees in G, and $\mathcal{R}_S(G) \subseteq \mathcal{R}(G)$ be the set of derivation trees whose root is labeled with a production of which left-hand side is the initial symbol S. Clearly, $\mathcal{R}_S(G) \subseteq \mathcal{T}_\Sigma(\phi)$ holds. Remark that $\mathcal{R}_S(G)$ is a multi-sorted algebra, where the nonterminals are sorts, and the terminals and the labels of productions are operators.

A *tree transducer* (Rounds 1969) defines a mapping from trees to trees. Since we are mainly interested in the string language generated by a tree transducer, a "tree-to-string" version of transducer defined in (Engelfriet 1980) is used in this paper. 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 (1) Q is a finite set of *states*, (2) Σ is an *input ranked alphabet*, (3) Δ is an *output alphabet*, (4) $q_0 \in Q$ is the *initial state*, and (5) R is a finite set of *rules* of the form

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

where $q \in Q, \sigma \in \Sigma, n = \rho(\sigma)$ and $v \in (\Delta \cup Q[\{x_1, \ldots, x_n\}])^*$. If any different rules in R have different left-hand sides, then M is called *deterministic* (Engelfriet 1980).

A *configuration* of a yT-transducer is an element in $(\Delta \cup Q[\mathcal{T}_\Sigma(\phi)])^*$. *Derivation* of M is defined as follows. Let $t = \alpha_1 q[\sigma(t_1, \ldots, t_n)]\alpha_2$ be a configuration where $\alpha_1, \alpha_2 \in (\Delta \cup Q[\mathcal{T}_\Sigma(\phi)])^*$, $q \in Q$, $\sigma \in \Sigma$, $\rho(\sigma) = n$ and $t_1, \ldots, t_n \in \mathcal{T}_\Sigma(\phi)$. Assume that there is a rule $q[\sigma(x_1, \ldots, x_n)] \to v$ in R. Let t' be obtained from v by substituting $t_1, \ldots, t_n$ for $x_1, \ldots, x_n$, respectively, then we define $t \Rightarrow_M \alpha_1 t' \alpha_2$. Let $\Rightarrow_M^*$ be the reflexive and transitive closure of $\Rightarrow$. If $t \Rightarrow_M^* t'$, then we say t' *is derived from* t. If there is no $w \in \Delta^*$ such that $t \Rightarrow_M^* w$, then we say *no output is derived from* t.

A *tree-to-string finite state translation system* (*yT-fts* or *fts*) is defined by a yT-transducer M and a cfg G, written as (M, G) (Rounds 1969)(Thatcher 1967).

We define $yL(M, G)$, called the *yield language generated by yT-fts* (M, G), as

$$yL(M, G) \triangleq \{w \in \Delta^* \mid \exists t \in \mathcal{R}_S(G), q_0[t] \Rightarrow_M^* w\}$$

where Δ is an output alphabet and q_0 is the initial state of M. An fts is called *deterministic* (Engelfriet 1980) if the transducer M is deterministic.

Engelfriet introduced a subclass of fts' called finite-copying fts' as follows (Engelfriet 1980): Let (M, G) be an fts with output alphabet Δ and initial state q_0, t be a derivation tree in G and t' be a subtree of t. Assume that there is a derivation $\alpha : q_0[t] \Rightarrow_M^* w$. Now, delete from this derivation α all the derivation steps which operates on t'. This leads to the following new derivation which keeps t' untouched;

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

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

The *state sequence of t' in derivation α* is defined to be $\langle q_{i_1}, \ldots, q_{i_n} \rangle$. Derivation α has *copying-bound* k if, for every subtree of t, the length of its state sequence is at most k. An fts (M, G) is a *finite-copying*, if there is a constant k and for each $w \in yL(M, G)$, there is a derivation tree t in G and a derivation $q_0[t] \Rightarrow_M^* w$ with copying-bound k. It is known that the determinism does not weaken the generative capacity of finite-copying fts' (Engelfriet 1980).

We note that an fts (M, G) can be considered to be a model of a transformational grammar: A deep-structure of a sentence is represented by a derivation tree of G, and M can be considered to transform the deep-structure into a sentence (or its surface structure).

4 Subclasses of Lexical-functional grammars

A simple subclass of lfg's, called *r-lfg*'s, is introduced in (Nishino 1992), which is shown to generate all the recursively enumerable languages (Nakanishi 1992). Here, we define a *nondeterministic copying lfg* (*nc-lfg*) as a proper subclass of r-lfg's. An nc-lfg is defined to be a 6-tuple $G = (N, T, P, S, N_{atr}, A_{atm})$ where: (1) N is a finite set of *nonterminal symbols*, (2) T is a finite set of *terminal symbols*, and (3) P is a finite set of *annotated productions*. Sometimes, a nonterminal symbol, a terminal symbol and an annotated production are abbreviated as a *nonterminal*, a *terminal* and a *production*, respectively. (4) $S \in N$ is the *initial symbol*, (5) N_{atr} is a finite set of *attributes*, and (6) A_{atm} is a finite set of *atoms*.

An equation of the form $\uparrow atr = \downarrow$ $(atr \in N_{atr})$ is called an *S* (*structure synthesizing*) *schema*, and an equation of the form $\uparrow atr = val$ $(atr \in N_{atr}, val \in A_{atm})$ is called a *V* (*immediate value*) *schema*. A *functional schema* is either an S schema or a V schema.

Each production $p \in P$ has the following form:

$$\begin{array}{lcllll} p : A & \rightarrow & B_1 & B_2 & \cdots & B_q, \\ E_V & & E_{S1} & E_{S2} & \cdots & E_{Sq} \end{array} \qquad (4.2)$$

where $A \in N$, $B_1, B_2, \cdots, B_q \in N \cup T$. E_V is a finite set of V schemata and E_{Sj} $(1 \leq j \leq q)$ is a singleton of an S schema. $A \rightarrow B_1 B_2 \cdots B_q$ in (4.2) is called the *underlying production of p*. Let P_0 be the set of all the underlying productions of P. Cfg $G_0 = (N, T, P_0, S)$ is called the *underlying cfg of G*.

An *f-structure* of G is recursively defined as a set $F = \{\langle atr_1, val_1 \rangle, \langle atr_2, val_2 \rangle, \cdots, \langle atr_k, val_k \rangle\}$ where $atr_1, atr_2, \cdots,$ and atr_k are distinct attributes, and each of $val_1, val_2, \cdots,$ and val_k is an atom or an f-structure. We say that val_i $(1 \leq i \leq k)$ is the *value of atr_i* in F and write $F.atr_i = val_i$.

For a cfg $G' = (N', T', P', S')$, derivation relations in G', denoted by $A \Rightarrow_{G'} \alpha$ and $A \Rightarrow_{G'}^* \alpha$ $(A \in N', \alpha \in (N' \cup T')^*)$ are defined in the usual way.

Suppose $G_0 = (N, T, P_0, S)$ is the underlying cfg of an nc-lfg $G = (N, T, P, S, N_{atr}, A_{atm})$. Let t be a derivation tree in G_0. (In 4.,7. and 8., the label of a leaf of a derivation tree is allowed to be a nonterminal.) Every internal node v in t has an f-structure, which is called the f-structure of v and written as F_v. If an underlying production $p_0 : A \rightarrow B_1 \cdots B_q \in P_0$ is applied at v, then v is labeled with either p_0 itself, or p $(\in P)$ of which p_0 is the underlying production, if necessary. Let v_i be the ith child of v $(1 \leq i \leq q)$. We define the values of both sides of a functional schema attached to the symbol in p (on v) as follows:

- the value of $\uparrow atr (atr \in N_{atr})$ is $F_v.atr$,
- the value of $\downarrow$ in an S schema is F_{v_i} if the S schema is attached to the $i (1 \leq i \leq q)$th symbol in the right-hand side of p, and
- the value of atom atm in a V schema is atm itself.

We say that v *satisfies* functional schemata if for each functional schema $l_{fs} = r_{fs}$ of p, the values of l_{fs} and r_{fs} on v are defined and equals with each other. In this case, it is also said that *F_v locally satisfies* the functional schemata of p.
NOTE: Because the meaning of a V schema is independent of the position where it is annotated, V schemata are attached to the left-hand side in this paper.

For a nonterminal $A \in N$ and a sentential form $\alpha \in (N \cup T)^*$, let t be a derivation tree of a derivation $A \Rightarrow_{G_0}^* \alpha$. If all internal nodes in t satisfy functional schemata, then α is said to *be derived from A* and written as $A \Rightarrow_G^* \alpha$. In this case, the tree t is called a *derivation tree of $A \Rightarrow_G^* \alpha$*. We also call t a derivation tree (of α) in G simply.

The *language generated by* an nc-lfg G, denoted by $L(G)$, is defined as $L(G) = \{w \in T^* \mid S \Rightarrow_G^* w\}$.
NOTE: In the definition of nc-lfg, even if

> "E_{Sj} $(1 \leq j \leq q)$ is a singleton of an S schema"

is replaced with

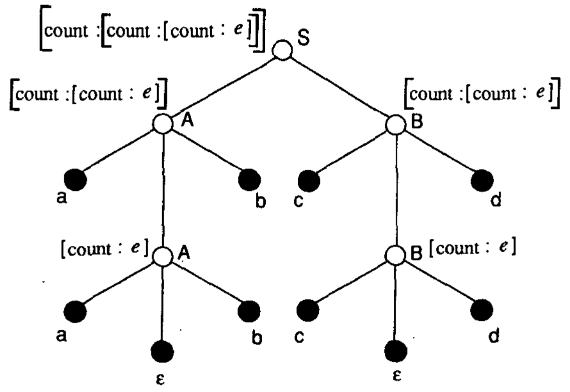

Figure 1: A derivation tree of *aabbccdd*

"E_{Sj} $(1 \leq j \leq q)$ is either a singleton of an S schema or an empty set",

the generative capacity of nc-lfg is not changed.

Example 4.1: Let $G_{EX3} = (N, T, P, S, N_{atr}, A_{atm})$ be an nc-lfg where $N = \{S, A, B\}$, $T = \{a, b, c, d\}$, $N_{atr} = \{count\}$, $A_{atm} = \{e\}$, and productions in P are;

$$p_{11} : S \to \underset{\{\uparrow count = \downarrow\}}{A} \; \underset{\{\uparrow count = \downarrow\}}{B},$$

$$p_{12} : A \to a \; \underset{\{\uparrow count = \downarrow\}}{A} \; b,$$

$$p_{13} : B \to c \; \underset{\{\uparrow count = \downarrow\}}{B} \; d,$$

$$p_{14} : \underset{\{\uparrow count = e\}}{A} \to \varepsilon,$$

$$p_{15} : \underset{\{\uparrow count = e\}}{B} \to \varepsilon.$$

The language generated by G_{EX3} is $L(G_{EX3}) = \{a^n b^n c^n d^n \mid n \geq 0\}$. Figure 1 shows a derivation tree of $S \Rightarrow^*_{G_{EX3}} aabbccdd$ in G_{EX3}. ▯

Example 4.2: Let $G_{EX4} = (N, T, P, S, N_{atr}, A_{atm})$ be an nc-lfg where $N = \{S\}$, $T = \{a\}$, $N_{atr} = \{log\}$, $A_{atm} = \{e\}$, and productions in P are;

$$p_{21} : S \to \underset{\{\uparrow log = \downarrow\}}{S} \; \underset{\{\uparrow log = \downarrow\}}{S},$$

$$p_{22} : \underset{\{\uparrow log = e\}}{S} \to a.$$

The language generated by G_{EX4} is $L(G_{EX4}) = \{a^{2^n} \mid n \geq 0\}$. ▯

Example 4.3: Let $G_{EX5} = (N, T, P, S, N_{atr}, A_{atm})$ be an nc-lfg where $N = \{S, S', A, B\}$, $T = \{$*the, woman, men, and, drinks, smoke, respectively*$\}$, $N_{atr} = \{num, list\}$, $A_{atm} = \{sg, pl, nil\}$, and productions in P are;

$$p_{30} : S \to \underset{\{\uparrow list = \downarrow\}}{S'} \; respectively,$$

$$p_{31} : \underset{\{\uparrow num = sg\}}{S'} \to the\ woman\ and\ \underset{\{\uparrow list = \downarrow\}}{A}\ drinks\ and\ \underset{\{\uparrow list = \downarrow\}}{B},$$

$$p_{32} : \underset{\{\uparrow num = pl\}}{S'} \to the\ men\ and\ \underset{\{\uparrow list = \downarrow\}}{A}\ smoke\ and\ \underset{\{\uparrow list = \downarrow\}}{B},$$

$$p_{33} : \underset{\{\uparrow num = sg\}}{A} \to the\ woman\ and\ \underset{\{\uparrow list = \downarrow\}}{A},$$

$$p_{34} : \underset{\{\uparrow num = pl\}}{A} \to the\ men\ and\ \underset{\{\uparrow list = \downarrow\}}{A},$$

$$p_{35} : \underset{\left\{\begin{array}{c}\uparrow num = sg \\ \uparrow list = nil\end{array}\right\}}{A} \to the\ woman,$$

$$p_{36} : \underset{\left\{\begin{array}{c}\uparrow num = pl \\ \uparrow list = nil\end{array}\right\}}{A} \to the\ men,$$

$$p_{37} : \underset{\{\uparrow num = sg\}}{B} \to drinks\ and\ \underset{\{\uparrow list = \downarrow\}}{B},$$

$$p_{38} : \underset{\{\uparrow num = pl\}}{B} \to smoke\ and\ \underset{\{\uparrow list = \downarrow\}}{B},$$

$$p_{39} : \underset{\left\{\begin{array}{c}\uparrow num = sg \\ \uparrow list = nil\end{array}\right\}}{B} \to drinks,$$

$$p_{310} : \underset{\left\{\begin{array}{c}\uparrow num = pl \\ \uparrow list = nil\end{array}\right\}}{B} \to smoke.$$

G_{EX5} generates "respectively" sentences such as "the woman and the men drinks and smoke respectively". ▯

For a set X of functional schemata, X is *consistent* iff neither the following (1) nor (2) holds.

(1) $\{\uparrow atr = val_1, \uparrow atr = val_2\} \subseteq X$
for some $atr \in N_{atr}$ and some $val_1, val_2 \in A_{atm}$ such that $val_1 \neq val_2$.

(2) $\{\uparrow atr = val, \uparrow atr = \downarrow\} \subseteq X$
for some $atr \in N_{atr}$ and some $val \in A_{atm}$.

Productions $p_1, \cdots, p_n$ are *consistent* iff $\cup_{1 \leq i \leq n} E^{(i)}$ is consistent where $E^{(i)}$ is the set of functional schemata of p_i. If productions are not consistent then they are called *inconsistent.*

An nc-lfg G is called a *deterministically copying lfg* (*dc-lfg*), if any two productions $A \to \alpha_1$ and $A \to \alpha_2$ whoes left-hand sides are the same are inconsistent.

Suppose $G = (N, T, P, S, N_{atr}, A_{atm})$ is an nc-lfg. Let $\{\{e_1, e_2, \cdots, e_n\}\}$ denote the multiset which consists of elements $e_1, e_2, \cdots, e_n$ that are not necessarily distinct. An *SPN* (*SubPhrase Nonterminal*) *multiset* in G is recursively defined as the following 1 through 3:

1. $\{\{S\}\}$ is an SPN multiset.
2. Suppose that $\{\{A_1, A_2, \cdots, A_h\}\}$ $(A_1, A_2, \cdots, A_h \in N)$ is an SPN multiset. Let $A_1 \to \alpha_1$,

$\cdots$, $A_h \to \alpha_h$ be consistent productions. For each $atr \in N_{atr}$, let MS_{atr} be the multiset consisting of all the nonterminals which appear in $\alpha_1, \cdots, \alpha_h$ and have an S schema $\uparrow atr = \downarrow$. If MS_{atr} is not empty, then MS_{atr} is also an SPN multiset.

3. There is no other SPN multiset.

An nc-lfg such that the number of SPN multisets in G is finite is called a *finite-copying lfg (fc-lfg)*.

Example 4.4: Consider G_{EX3} in Example 4.1. Productions p_{12} and p_{14} are inconsistent with each other and so are p_{13} and p_{15}. SPN multisets in G_{EX3} are $\{\{S\}\}$ and $\{\{A, B\}\}$. Hence G_{EX3} is a dc-lfg and is an fc-lfg. G_{EX5} is also a dc-lfg and is an fc-lfg by the similar reason. Similarly, G_{EX4} in Example 4.2 is a dc-lfg. SPN multisets in G_{EX4} are $\{\{S\}\}$, $\{\{S, S\}\}$, $\{\{S, S, S, S\}\}$, $\cdots$. Hence G_{EX4} is not an fc-lfg. $\square$

NOTE : $L(G_{EX5})$ is generated by a tree adjoining grammar. Suppose that a sentence has three or more phrases which have co-occurrence relation like the one between the subject phrase and the verb phrase in the "respectively" sentence. Tree adjoining grammars can not generate such syntax while fc-lfg's or dc-lfg's can, although the authors do not know a natural language which has such syntax so far.

By Lemma 2.1 and Theorem 8.1, fc-lfg's are polynomial-time recognizable. Hence, it is desirable that whether a given lfg G is an fc-lfg or not is decidable. Fortunately, it is decidable by the following lemma.

Lemma 4.1: For a given nc-lfg G, it is decidable whether the number of SPN multisets in G is finite or infinite.

Proof. The problem can be reduced to the boundedness problem of Petri nets, which is known to be decidable (Peterson 1981). $\square$

5 Overview of the Results

Let $\mathcal{L}_{nc\text{-}lfg}$, $\mathcal{L}_{dc\text{-}lfg}$ and $\mathcal{L}_{fc\text{-}lfg}$ denote the classes of languages generated by nc-lfg's, dc-lfg's and fc-lfg's, respectively, and let $y\mathcal{L}_{fts}$, $y\mathcal{L}_{d\text{-}fts}$ and $y\mathcal{L}_{fc\text{-}fts}$ denote the classes of yield languages generated by fts', deterministic fts' and finite-copying fts', respectively. Let $\mathcal{L}_{pmcfg}$ and $\mathcal{L}_{mcfg}$ be the classes of languages generated by pmcfg's and mcfg's, respectively. Also let $\mathcal{L}_{tag}$ be the class of language generated by tree adjoining grammars.

Inclusion relations among these classes of languages are summarized in Figure 2. An equivalence relation $*1$ is shown in (Weir 1992). Relations $*2$ are new results which we prove in this paper. We also note that all the inclusion relations are proper; indeed,

$$\{a_1^n a_2^n a_3^n a_4^n \mid n \geq 0\} \in D - E$$
$$\{a_1^n a_2^n \cdots a_{2m-1}^n a_{2m}^n \mid n \geq 0\} \in C - D \text{ for } m \geq 3,$$
(by (Vijay-Shanker 1987).)
$$\{a^{2^n} \mid n \geq 0\} \in B - C,$$
(by (Kasami 1988a)(Seki 1991).)

A relation $B \subsetneq A$ is shown in (Engelfriet 1980). By Lemma 2.1, all languages in the region enclosed with the bold line are recognizable in polynomial time. On the other hand, it is shown in this paper that Unary-3SAT, which is known to be $\mathcal{NP}$-complete (Nakanishi 1992), is in A. Hence, if $\mathcal{P} \neq \mathcal{NP}$, then Unary-3SAT $\in A - B$ and the languages generated by fts' (or equivalently, nc-lfg's) are not recognizable in polynomial time in general.

6 Generative Capacity of fts'

6.1 Deterministic fts'

Here, the proof of an inclusion relation $y\mathcal{L}_{d\text{-}fts} \subseteq \mathcal{L}_{pmcfg}$ is sketched.

Let (M, G) be a deterministic yT-fts where $M = (Q, \Sigma, \Delta, q_1, R)$ and $G = (N, T, P, S)$. We assume that $Q = \{q_1, \ldots, q_\ell\}$, $T = \{a_1, \ldots, a_n\}$ and $P = \{p_1, \ldots, p_m\}$. Since the input for M is the set of derivation trees of G, we assume that $\Sigma = \{p_1, \ldots, p_m, a_1, \ldots, a_n\}$ without loss of generality.

We will construct a pmcfg $G' = (N', T', F', P', S')$ such that $yL(M, G) = L(G') \cap \Delta^*$. Since $\mathcal{L}_{pmcfg}$ is closed under the intersection with a regular set (Kasami 1988a)(Seki 1991), it follows that $yL(M, G) \in \mathcal{L}_{pmcfg}$. Let $T' = \Delta \cup \{b\}$ where b is a newly introduced symbol and let

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

where $d(R_i) = d(A_j) = \ell$ for $1 \leq i \leq m$ and $1 \leq j \leq n$. Productions and functions of G' will be constructed to have the following property.

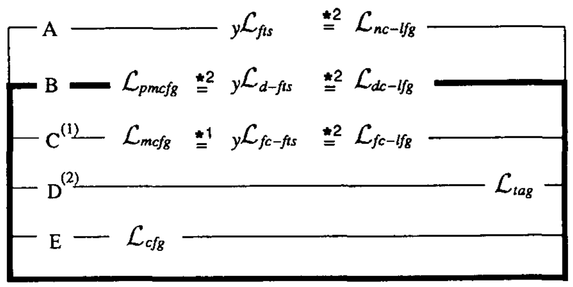
A
$y\mathcal{L}_{fts}$ $\overset{*2}{=}$ $\mathcal{L}_{nc\text{-}lfg}$
B
$\mathcal{L}_{pmcfg}$ $\overset{*2}{=}$ $y\mathcal{L}_{d\text{-}fts}$ $\overset{*2}{=}$ $\mathcal{L}_{dc\text{-}lfg}$
$C^{(1)}$
$\mathcal{L}_{mcfg}$ $\overset{*1}{=}$ $y\mathcal{L}_{fc\text{-}fts}$ $\overset{*2}{=}$ $\mathcal{L}_{fc\text{-}lfg}$
$D^{(2)}$
$\mathcal{L}_{tag}$
E
$\mathcal{L}_{cfg}$

Figure 2: Inclusion relations between classes of languages. (1) : The class of language generated by lcfrs' is equal to C. (2) : The class of language generated by head grammars is equal to D.

Property 6.1: There is $(\alpha_1, \ldots, \alpha_\ell) \in L_{G'}(R_h)$ (resp. $L_{G'}(A_h)$) such that

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

if and only if there is a derivation tree t of G such that the root is p_h (resp. a_h) and

$$\begin{cases} q_{s_j}[t] \Rightarrow^*_M \alpha_{s_j} & (1 \leq j \leq u) \\ \text{no output is derived from } q_{t_j}[t] & (1 \leq j \leq v). \end{cases}$$

□

The basic idea is to simulate the move of tree transducer M which is scanning a symbol p_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.

We define RS(X) $(X \in N \cup T)$ as follows.

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

Productions and functions are defined as follows.

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

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

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

- if the transducer M has no rule whose left-hand side is $q_i[p_h(x_1, \ldots, x_k)]$, then

$$f^{[i]}_{p_h}[\bar{x}_1, \ldots, \bar{x}_k] \triangleq b, \qquad (6.3)$$

- if M has a rule $q_i[p_h(x_1, \ldots, x_k)] \to \alpha_{i,0}\ q_{\eta(i,0)}[x_{\mu(i,0)}]\alpha_{i,1} \cdots \alpha_{i,n_i-1} q_{\eta(i,n_i-1)}[x_{\mu(i,n_i-1)}]\ \alpha_{i,n_i}$, then

$$f^{[i]}_{p_h}[\bar{x}_1, \ldots, \bar{x}_k] \triangleq \alpha_{i,0} x_{\mu(i,0)\eta(i,0)} \alpha_{i,1} \cdots \alpha_{i,n_i-1} x_{\mu(i,n_i-1)\eta(i,n_i-1)} \alpha_{i,n_i} \qquad (6.4)$$

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 $q_i[p_h(\cdots)]$ and hence the above construction is well defined.)

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

- if M has no rule whose left-hand side is $q_i[a_h]$, then $f^{[i]}_{a_h} \triangleq b$.
- if M has a rule $q_i[a_h] \to \alpha_i$, then $f^{[i]}_{a_h} \triangleq \alpha_i$.

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

The pmcfg G' constructed above satisfies Property 6.1. Its proof is found in (Kaji 1992) and omitted in this paper. By Property 6.1, we obtain the following lemma.

Lemma 6.1: $y\mathcal{L}_{d\text{-}fts} \subseteq \mathcal{L}_{pmcfg}$. □

The reverse inclusion relation $\mathcal{L}_{pmcfg} \subseteq y\mathcal{L}_{d\text{-}fts}$ can be shown in a similar way, and the following theorem holds.

Theorem 6.2: $y\mathcal{L}_{d\text{-}fts} = \mathcal{L}_{pmcfg}$. □

6.2 Nondeterministic fts'

In this section, the generative capacity of nondeterministic yT-fts' is investigated, from the viewpoint of computational complexity. We have already shown that $y\mathcal{L}_{d\text{-}fts} = \mathcal{L}_{pmcfg}$, and hence every language in this class can be recognized in time polynomial of the length of an input string. Our result here is: there is a nondeterministic fts that generates an $\mathcal{NP}$-complete language. In the following, a language called *Unary-3SAT*, which is $\mathcal{NP}$-complete (Nakanishi 1992), is considered, and then it is shown to belong to $y\mathcal{L}_{fts}$.

A *Unary-3CNF* is a (nonempty) 3CNF in which the subscripts of variables are represented in unary. A positive literal x_i in a 3CNF 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 a Unary-3CNF

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

Unary-3SAT is the set of all satisfiable Unary-3CNF's.

Next, we construct a nondeterministic yT-fts (M, G) that generates Unary-3SAT. Define a cfg $G = (N, T, P, S)$ where $N = \{S, T, F\}$, $T = \{e\}$ and the productions in P are as follows:

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

Let $M = (Q, \Sigma, \Delta, q_0, R)$ where

$$\begin{aligned} Q &= \{q_0, q_c, q_t, q_a\}, \\ \Sigma &= \{r_{SS}, \ldots, r_{Fe}\}, \\ \Delta &= \{1, \wedge, \$, \#\}. \end{aligned}$$

Since there are many rules in R, we will use an abbreviated notation. For example, following four rules

$$\begin{aligned} q_a[r_{Te}(x)] &\rightarrow 1\$,\ q_a[r_{Te}(x)] \rightarrow 1\# \\ q_a[r_{Fe}(x)] &\rightarrow 1\$,\ q_a[r_{Fe}(x)] \rightarrow 1\# \end{aligned}$$

are abbreviated as "$q_a[r_{Te}(x)] = q_a[r_{Fe}(x)] \rightarrow 1\$$ or $1\#$". By using this notation, the rules in R are defined as follows.

$$\begin{aligned} q_0[r_{SS}(x)] &\rightarrow q_c[x] \wedge q_0[x] \\ q_c[r_{SS}(x)] &\rightarrow q_c[x] \\ q_0[r_{ST}(x)] = q_0[r_{SF}(x)] &= q_c[r_{ST}(x)] \\ = q_c[r_{SF}(x)] &\rightarrow q_t[x]q_a[x]q_a[x] \text{ or} \\ & \quad q_a[x]q_t[x]q_a[x] \text{ or } q_a[x]q_a[x]q_t[x] \\ q_t[r_{TT}(x)] = q_t[r_{TF}(x)] &\rightarrow 1q_t[x] \text{ or } 1\$ \\ q_t[r_{Te}(x)] &\rightarrow 1\$ \\ q_t[r_{FT}(x)] = q_t[r_{FF}(x)] &\rightarrow 1q_t[x] \text{ or } 1\# \\ q_t[r_{Fe}(x)] &\rightarrow 1\# \\ q_a[r_{TT}(x)] = q_a[r_{TF}(x)] &= q_a[r_{FT}(x)] \\ = q_a[r_{FF}(x)] &\rightarrow 1q_a[x] \text{ or } 1\$ \text{ or } 1\# \\ q_a[r_{Te}(x)] = q_a[r_{Fe}(x)] &\rightarrow 1\$ \text{ or } 1\#. \end{aligned}$$

The readers can easily verify that this yT-fts generates Unary-3SAT.

7 Equivalence of $\mathcal{L}_{nc\text{-}lfg}$ and $y\mathcal{L}_{fts}$

First, we show $\mathcal{L}_{nc\text{-}lfg} \subseteq y\mathcal{L}_{fts}$. For a given nc-lfg $G = (N, T, P, S, N_{atr}, A_{atm})$, an equivalent fts (M, G') is constructed in the following way.

Let t be a derivation tree in lfg G and the f-structure of the root node of t be $F = \{\langle atr_1, F_1\rangle, \cdots, \langle atr_n, F_n\rangle\}$. F is represented by a derivation tree $\tau = p'_{sp}(\tau_1, \cdots, \tau_n)$ in G', where $\tau_i\ (1 \le i \le n)$ is a derivation tree in G' which represents F_i recursively. And sp is a set of productions such that F locally satisfies the functional schemata of all productions in sp. M transforms τ into the yield of t, i.e., the terminal string obtained by concatenating the labels of leaves, in a top-down way.

[TRANS 7.1] Let $N = \{A_1, \cdots, A_m\}$, $S = A_1$ and $N_{atr} = \{atr_1, \cdots, atr_n\}$. Define SP as the set of all consistent subsets of P.

Step 1: $G' = (N', \{d\}, P', S')$, where $N' = \{S_{sp} | sp \in SP\} \cup \{S'\}$ and

$$\begin{aligned} P' = \{ & p'_{sp} : S_{sp} \rightarrow \underbrace{S' \cdots S'}_{n}\} \\ \cup \{ & p'_{guess_{sp}} : S' \rightarrow S_{sp} | sp \in SP\} \\ \cup \{ & p'_{term} : S' \rightarrow \underbrace{d \cdots d}_{n}\}. \end{aligned}$$

For a derivation tree τ in G' and a node v where p'_{sp} is applied, the subtree rooted by the ith child of v represents the value of attribute atr_i.

Step 2: $M = (Q, \Sigma, T, q_1, R)$ is defined as follows.

Define $Q = \{q_1, \ldots, q_m\}$. A state $q_j\ (1 \le j \le m)$ corresponds to nonterminal A_j in N. Define $\Sigma = \{p'_{sp} \,|\, sp \in SP\} \cup \{p'_{guess_{sp}} \,|\, sp \in SP\} \cup \{p'_{term}\} \cup \{d\}$ where $\rho(p'_{sp}) = \rho(p'_{guess_{sp}}) = \rho(p'_{term}) = n$ and $\rho(d) = 0$. And define R by the following (i) through (iii).

(i) $q_j[p'_{guess_{sp}}(x)] \rightarrow q_j[x]\ (1 \le j \le m)$ belongs to R for each $sp \in SP$.

(ii) Let τ be a derivation tree in G'. When p'_{sp} is the production applied at the root of τ and a state of M is q_{μ_0}, M chooses a production p whose left-hand side is A_{μ_0}, if exists, in sp.

NOTE : Since productions in sp are consistent, there is an f-structure, which locally satisfies the functional schemata of all productions in sp.

For each production $p \in sp$ in SP

$$\begin{array}{lllllll} p : A_{\mu_0} \rightarrow \alpha_0 & A_{\mu_1} & \alpha_1 \ldots \alpha_{L-1} & A_{\mu_L} & \alpha_L \\ \quad E_V & \{\uparrow atr_{\nu_1} = \downarrow\} & \ldots & \{\uparrow atr_{\nu_L} = \downarrow\} \end{array}$$

where $A_{\mu_l} \in N$ and $\alpha_l \in T^*(0 \le l \le L)$, the following rule belongs to R:

$$\begin{aligned} & q_{\mu_0}[p'_{sp}(x_1, \cdots, x_n)] \\ & \quad \rightarrow \alpha_0 q_{\mu_1}[x_{\nu_1}]\alpha_1 \ldots \alpha_{L-1} q_{\mu_L}[x_{\nu_L}]\alpha_L. \quad (7.5) \end{aligned}$$

(iii) No other rule belongs to R. $\square$

Next, $y\mathcal{L}_{fts} \subseteq \mathcal{L}_{nc\text{-}lfg}$ is shown. For a given fts (M, G), the following algorithm constructs an nc-lfg G' such that $L(G') = yL(M, G)$.

[TRANS 7.2] Suppose that a given fts (M, G) is $G = (N, T, P, S)$ and $M = (Q, \Sigma, \Delta, q_1, R)$ where $Q = \{q_1, q_2, \cdots, q_m\}$. Let n be the maximum length of the right-hand side of a production in P. Define an nc-lfg $G' = (N', \Delta, P', S', N_{atr}, A_{atm})$ as follows.

Step 1:
$$\begin{aligned} N' &= \{C^{[j]} \,|\, C \in N, 1 \le j \le m\} \\ & \quad \cup \{a^{[j]} \,|\, a \in T, 1 \le j \le m\}, \\ S' &= S^{[1]}, \\ N_{atr} &= \{atr_i \,|\, 1 \le i \le n\} \cup \{rule\}, \text{ and} \\ A_{atm} &= \{p \,|\, p \text{ is the label of} \\ & \qquad \text{a production in } P\}. \end{aligned}$$

A derivation tree $t = p(t_1, \cdots, t_h)$ in G is represented by an f-structure $\{\langle rule, p\rangle, \langle atr_1, F_1\rangle, \cdots, \langle atr_h, F_h\rangle\}$ of G' where $F_i\ (1 \le i \le h)$ is an f-structure which represents the subtree t_i recursively.

Each pair of a symbol (either nonterminal or terminal) X of G and a state q_j of M is represented by a single nonterminal $X^{[j]}$ in G'.

Step 2: A move when M at state q_j reads a symbol p which is the label of a production $p : C \rightarrow \cdots$, can be simulated by a production in G' whose left-hand side is $\underset{\{\uparrow rule = p\}}{C^{[j]}}$.

Formally, the set P' of productions of G' is constructed as follows.

(i) Let $p : C \rightarrow X_1 \cdots X_h$ be a production in P where $C \in N$, $X_i \in N \cup T$ $(1 \leq i \leq h)$, and let:

$$q_j[p(x_1, \cdots, x_h)] \rightarrow \alpha_{j0} q_{\eta_{j1}}[x_{\nu_{j1}}]\alpha_{j1} \cdots q_{\eta_{jL_j}}[x_{\nu_{jL_j}}]\alpha_{jL_j}$$

be a rule in R where $\alpha_{jk} \in \Delta^*$ $(0 \leq k \leq L_j)$, $q_{\eta_{jl}} \in Q$, and $x_{\nu_{jl}} \in \{x_1, \cdots, x_h\}$ $(1 \leq l \leq L_j)$.
Then, the following production belongs to P':

$$\underset{\{\uparrow rule = p\}}{C^{[j]}} \rightarrow \alpha_{j0} \underset{\{\uparrow atr_{\nu_{j1}} = \downarrow\}}{X_{\nu_{j1}}^{[\eta_{j1}]}} \alpha_{j1} \cdots \underset{\{\uparrow atr_{\nu_{jL_j}} = \downarrow\}}{X_{\nu_{jL_j}}^{[\eta_{jL_j}]}} \alpha_{jL_j}.$$

(ii) Let $q_j[a] \rightarrow \beta_j$ be a rule in R where $a \in T$ and $\beta_j \in \Delta^*$. Then the production $a^{[j]} \rightarrow \beta_j$ belongs to P'.

(iii) No other production belongs to P'. □

By **TRANS 7.1** and **TRANS7.2**, the following theorem is obtained. A formal proof is found in (Nakanishi 1993).

Theorem 7.1: $\mathcal{L}_{nc\text{-}lfg} = y\mathcal{L}_{fts}$. □

Corollary 7.2: $\mathcal{L}_{dc\text{-}lfg} = y\mathcal{L}_{d\text{-}fts}$.

Proof. In **TRANS 7.1**, if G is a dc-lfg, then no $sp \in SP$ contains distinct productions whose left-hand sides are the same and hence the constructed transducer M becomes deterministic by the construction. Conversely, in **TRANS 7.2**, if M is deterministic, then there exist no consistent productions p'_1 and p'_2 in P' whose left-hand sides are the same and hence the constructed nc-lfg is a dc-lfg. □

8 Equivalence of $\mathcal{L}_{fc\text{-}lfg}$ and $\mathcal{L}_{mcfg}$

To prove $\mathcal{L}_{fc-lfg} \subseteq \mathcal{L}_{mcfg}$, we give an algorithm which translates a given fc-lfg $G = (N, T, P, S, N_{atr}, A_{atm})$ into an mcfg G' such that $L(G') = L(G)$.

[TRANS 8] We explain the algorithm by using the fc-lfg G_{EX3} in Example 4.1. An mcfg $G' = (N', T, F, P', S)$ is constructed as follows.

Step 1: N' = (*the set of nonterminals which has a one-to-one correspondence with the set of SPN multisets in G*)
= $\{\langle S\rangle, \langle A, B\rangle\}$
(for G_{EX3} in Example 4.1)
$P' = \phi$, and
$F = \phi$.

Step 2: For each SPN multiset $M_0 = \{\{A_1, A_2, \cdots, A_k\}\}$ of G, consider every tuple $(p_1, p_2, \cdots, p_k)$ of productions in P whose left-hand sides are $A_1, A_2, \cdots, A_k$ respectively and which are consistent. (Suppose that, if we write an SPN multiset as $\{\{A_1, A_2, \cdots, A_k\}\}$, then A_j's are arranged according to a predefined total order $\leq$ on N, that is, $A_1 \leq A_2 \leq \cdots \leq A_k$ hold.) For an SPN multiset $\{\{A, B\}\}$ in G_{EX3}, the following two pairs of productions have to be considered:

$$\begin{cases} p_{12} : A \rightarrow a \underset{\{\uparrow count = \downarrow\}}{A} b \\ p_{13} : B \rightarrow c \underset{\{\uparrow count = \downarrow\}}{B} d, \end{cases}$$

$$\begin{cases} p_{14} : \underset{\{\uparrow count = e\}}{A} \rightarrow \varepsilon \\ p_{15} : \underset{\{\uparrow count = e\}}{B} \rightarrow \varepsilon. \end{cases}$$

For $(p_1, p_2, \cdots, p_k)$, a production p' and a function f of G' are constructed and added to P' and F, respectively as follows.

The multiset M of the nonterminals appearing in the right-hand side of some p_j $(1 \leq j \leq k)$ are partitioned into multisets $M_1, M_2, \cdots, M_h$ with respect to the S schemata attached to the nonterminals in p_j's. That is, $(M_1, M_2, \cdots, M_h)$ are the coarsest partition of M such that for each M_u $(1 \leq u \leq h)$, the following condition holds.

Each nonterminal in M_u has the same S schema.

By the definition, each M_u $(1 \leq u \leq h)$ is an SPN multiset in G. Construct a production of mcfg $p' : \bar{M_0} \rightarrow f[\bar{M_1}, \bar{M_2}, \cdots, \bar{M_h}]$ where $\bar{M_u}$ is the nonterminal of G' which corresponds to M_u $(1 \leq u \leq h)$. Add p' to P' and f to F where f is defined as follows. Suppose

$$p_j : A_j \rightarrow \alpha_{j0} B_{j1} \alpha_{j1} \cdots B_{jL_j} \alpha_{jL_j} \ (1 \leq j \leq k)$$

where $A_j \in N$, $B_{jl} \in N$ $(1 \leq l \leq L_j)$ and $\alpha_{jl} \in T^*$ $(0 \leq l \leq L_j)$, and let

$$M_u = \{\{C_{u1}, C_{u2}, \cdots, C_{us_u}\}\} \ (1 \leq u \leq h)$$

where $C_{uv} \in N$ $(1 \leq v \leq s_u)$. Then, for $1 \leq j \leq k$, the jth component $f^{[j]}$ of f is:

$$f^{[j]}(\bar{x_1}, \bar{x_2}, \cdots, \bar{x_h}) \triangleq \alpha_{j0} y_{j1} \alpha_{j1} y_{j2} \cdots y_{jL_j} \alpha_{jL_j}$$

where $\bar{x_u} = (x_{u1}, x_{u2}, \cdots, x_{us_u})$ $(1 \leq u \leq h)$. For j $(1 \leq j \leq k)$ and l $(1 \leq l \leq L_j)$, if $B_{jl} = C_{uv}$ then $y_{jl} \triangleq x_{uv}$. Note that, since M_u's are a partition of M, f satisfies Right Linearity (see 2.) and hence G' is an mcfg. For example, consider the above (p_{12}, p_{13}). The nonterminals appearing in the right-hand sides are A and B, and their S schemata are the same. Thus, we construct the following mcfg production:

$$\langle A, B\rangle \to f_1\,[\langle A, B\rangle]$$

$$\text{where } f_1[(x_1, x_2)] = (ax_1b, cx_2d).$$

Consider the following pair of productions as another example:

$$\left\{\begin{array}{llllll} p'_1 : A \to & a & B & b & D & \\ & & \{\uparrow atr_1 = \downarrow\} & & \{\uparrow atr_2 = \downarrow\} & \\ p'_2 : B \to & A & C & c & D & \\ & \{\uparrow atr_1 = \downarrow\} & \{\uparrow atr_2 = \downarrow\} & & \{\uparrow atr_2 = \downarrow\} & \end{array}\right.$$

The multiset of nonterminals in the right-hand sides are partitioned into $M_1 = \{\{A, B\}\}$ (for atr_1) and $M_2 = \{\{C, D, D\}\}$ (for atr_2). For (p'_1, p'_2), the following mcfg production is constructed:

$$\langle A, B\rangle \to g\,[\langle A, B\rangle, \langle C, D, D\rangle]$$

where $g\,[(x_{11}, x_{12}), (x_{21}, x_{22}, x_{23})] \triangleq (ax_{12}bx_{22}, x_{11}x_{21}cx_{23})$. □

Example 8.1: **TRANS 8** translates fc-lfg G_{EX3} in Example 4.1 into an equivalent mcfg $G'_{EX3} = (N', T, F, P', S')$ where N', S' are those illustrated in **TRANS 8**, $F = \{f_0[(x_1, x_2)] = x_1x_2,\ f_1[(x_1, x_2)] = (ax_1b, cx_2d),\ f_2 = (\varepsilon, \varepsilon)\}$, and, $P' = \{\langle S\rangle \to f_0[\langle A, B\rangle],\ \langle A, B\rangle \to f_1[\langle A, B\rangle],\ \langle A, B\rangle \to f_2\}$. □

Theorem 8.1: $\mathcal{L}_{mcfg} = \mathcal{L}_{fc-lfg}$.

Proof : $\mathcal{L}_{fc-lfg} \subseteq \mathcal{L}_{mcfg}$ can be proved by **TRANS 8**. Conversely, for a given mcfg G, an fc-lfg G' such that $L(G') = L(G)$ can be constructed in a similar way to **TRANS 8**. Details are found in (Ando 1992). □

9 Conclusion

In this paper, we introduce three subclasses of lfg's, two of which can be recognized in polynomial time. Also this paper clarifies the relations between the generative capacities of those subclasses, pmcfg's and fts'.

References

Ando, S. et al. 1992. "Subclasses of Lexical-Functional Grammars Which Are Recognizable in Polynomial Time", IEICE Technical Report, **COMP92-44**.

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

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

Joshi, A.K., Levy, L. and Takahashi, M. 1975 "Tree Adjunct Grammars", J. of Comput. & Syst. Sci., **10**:136–163.

Gazdar, G. and Pullum, G.K. 1985. "Computationally Relevant Properties of Natural Languages and Their Grammars", New Generation Computing, **3**:273–306.

Kaji, Y. et al. 1992. "Parallel Multiple Context-Free Grammars and Finite State Translation Systems ", IEICE Technical Report, **COMP92-34**.

Kaplan, R. and Bresnan, J. 1982. "Lexical-Functional Grammar", *The Mental Representation of Grammatical Relations*, J.Bresnan (ed.), MIT press:173–281.

Kasami, T. et al. 1988a. "Generalized Context-Free Grammars and Multiple Context-Free Grammars", Trans. IEICE, **J71-D-I**, 5:758–765.

Kasami, T. et al. 1988b. "On the Membership Problem for Head Language and Multiple Context-Free Languages, Trans. IEICE, **J71-D-I**, 6:935–941.

Nakanishi, R. et al. 1993. "On the Generative Capacity of Tree Translation Systems and Lexical Functional-Grammars", Technical Paper of FAI, Japanese Society for Artificial Intelligence, **SIG-FAI-9202**.

Nakanishi, R. et al. 1992. "On the Generative Capacity of Lexical-Functional Grammars", IEICE Trans. Inf. and Syst., **75-D**, 7:509–516.

Nishino, T. 1991. "Mathematical Analysis of Lexical-Functional Grammars —Complexity, Parsability, and Learnability—", Language Research Institute, Seoul National University.

Nishino, T. 1992. "Relating Attribute Grammars and Lexical-Functional Grammars", Information Sciences, **66**:1–22.

Peterson, J.L. 1981. "Petri Net Theory and the Modeling of Systems", Prentice-Hall.

Pollard, C.J. 1984. "Generalized Phrase Structure Grammars, Head Grammars and Natural Language", Ph.D. dissertation, Stanford University.

Rounds, W.C. 1969. "Context-Free Grammars on Trees", Proc. of ACM STOC:143–148.

Seki, H. et al. 1991. "On Multiple Context-Free Grammars", Theoretical Computer Science, **88**, 2:191-229.

Thatcher, J.W. 1967. "Characterizing Derivation Trees of Context-Free Grammars through a Generalarization of Finite Automata Theory". J. Comput. & Syst. Sci., **1**:317–322.

Vijay-Shanker, K. 1987. "A Study of Tree Adjoining Grammars", Ph.D. thesis, University of Pennsylvania.

Vijay-Shanker, K., Weir, D.J. and Joshi, A.K. 1987. "Characterizing structural descriptions produced by various grammatical formalisms", Proc. of 25th meeting of Assoc. Comput. Ling.:104–111.

Weir, D.J. 1988. "Characterizing Mildly Context-Sensitive Grammar Formalisms", Ph.D. thesis, University of Pennsylvania.

Weir, D.J. 1992. "Linear Context-Free Rewriting Systems and Deterministic Tree-Walking Transducers", Proc. of 30th meeting of Assoc. Comput. Ling.