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 (lfg's), 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.
cursively enumerable languages even though the underlying CFGs are restricted to regular grammars. In [Gazdar 1985](Kaplan 1982)(Nishino 1991), subclasses of IFLs were proposed in order to guarantee the recursiveness (and/or the efficient recognition) of languages generated by IFLs. However, these classes were shown to generate an \( \mathcal{NP} \)-complete language (Nakanishi 1992).

In this paper, three subclasses of IFLs called nc-IFLs, dc-IFLs and fc-IFLs are proposed, two of which can be recognized in polynomial time. Moreover, this paper clarifies the relations among the generative capacities of pMCFGs, FTSs and these subclasses of IFLs.

In nc-IFLs, a functional schema either specifies the value of a specific attribute, say \( \text{atr} \), immediately \( (\text{atr} = \text{val}) \) 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 \( v \) \( (\text{atr} = 1) \).

An nc-IFL is called a dc-IFL 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-IFL \( G \), 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-IFL. A multiset \( M \) of nonterminals of \( G \) is called an \( \text{SPN multiset} \) in \( G \) if the following condition holds:

Let \( M = \{\{A_1, A_2, \ldots , A_n\}\} \) 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 \( V = \{v_1, v_2, \ldots , v_n\} \) of \( t \) such that the label of \( v_1 \) is \( A_1 \) \( (1 \leq i \leq n) \) 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-IFL.

Our main result is that the generative capacity of nc-IFLs is equal to that of FTSs. As relations among proper subclasses of the above mentioned formalisms, it is shown that the generative capacities of dc-IFLs, deterministic FTSs and pMCFGs are equal to each other, and the generative capacity of fc-IFLs is equal to that of pMCFGs. 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 \((T^*)^{d_1(f)} \times (T^*)^{d_2(f)} \times \cdots \times (T^*)^{d_r(f)} \) 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 \( i \)-th argument of \( f \) for \( 1 \leq i \leq a(f) \).

(f1) For \( 1 \leq h \leq r(f) \), the \( h \)-th component of \( f \), denoted by \( f^h \), is defined as:

\[
f^h[x_{i1}, x_{i2}, \ldots , x_{id_i(f)}] = \alpha_{h,0} \cdot x_{i(h,0)} \cdot \alpha_{h,1} \cdot \cdots \cdot \alpha_{h,n_h-1} \cdot x_{i(h,n_h-1)} \cdot (h,n_h-1) \cdot \alpha_{h,n_h} \quad (2.1)
\]

where \( \alpha_{h,k} \in T^* \) for \( 0 \leq k \leq n_h \), \( 1 \leq \mu(h,j) \leq \alpha(f) \) and \( 1 \leq \eta(h,j) \leq \alpha(f) \) 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 $G_{EX1} = (N, T, F, P, S)$ be a pmcfg where $N = \{S, A\}$, $T = \{a, b\}$, $F = \{f_1, f_2\}$, $P = \{r_1 : S \rightarrow f_1, r_0 : S \rightarrow f|S|\}$, where $f_a = a, f[f(x)] = xx$. $G_{EX1}$ is a pmcfg but is not an mcfg since the function $f$ does not satisfy Right Linearity. The language generated by $G_{EX1}$ is

$$\{a^n \mid n \geq 0\},$$

which cannot be generated by any mcfg (see Lemma 6 of (Kasami 1988a)).

The empty string is denoted by $\varepsilon$.

**Example 2.2**: Let $G_{EX2} = (N, T, F, P, S)$ be a pmcfg, where $N = \{S, A\}$, $T = \{a, b\}$, $F = \{g[x_1, x_2] \rightarrow x_1x_2, f_a[x_1, x_2] \rightarrow (x_1a, x_2a), f_b[x_1, x_2] \rightarrow (x_1b, x_2b), f_e = (e, e)\}$, and $P = \{p_0 : S \rightarrow g[A], p_1 : A \rightarrow f_a[A], p_2 : A \rightarrow f_b[A], p_3 : A \rightarrow f_e\}$. Note that $G_{EX2}$ is an mcfg.

$L(G_{EX2}) = \{ww \mid w \in \{a, b\}^*\}$.

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 $w$.

### 3 Finite State Translation Systems

A set $\Sigma$ 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 $\sigma$. For a set $X$, we define free algebra $T_\Sigma(X)$ as the smallest set such that:

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

Hereafter, a term in $T_\Sigma(X)$ 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 $p : A \rightarrow X_1 \cdots X_n$ $(A \in N, X_1, \ldots, X_n \in N \cup T)$ 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 $X_i \rightarrow \cdot$, 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 $R(G)$ be the set of derivation trees in $G$, and $R_2(G) \subseteq R(G)$ be the set of derivation trees whose root is labeled with a production of which left-hand side is the initial symbol $S$. Clearly, $R_2(G) \subseteq T_\Sigma(\phi)$ holds. Remark that $R_2(G)$ 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) $\Sigma$ is an input ranked alphabet, (3) $\Delta$ 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

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

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 $(\Delta \cup Q[T_\Sigma(\phi)])^*$. Derivation of $M$ is defined as follows. Let $t = \alpha_1q[\sigma_1(t_1, \ldots, t_n)]\alpha_2$ be a configuration where $\alpha_1, \alpha_2 \in (\Delta \cup Q[T_\Sigma(\phi)])^*, q \in Q, \sigma \in \Sigma, \rho(\sigma) = n$ and $t_1, \ldots, t_n \in T_\Sigma(\phi)$. Assume that there is a rule $q[\sigma(x_1, \ldots, x_n)] \rightarrow v$ in $R$. Let $t'$ be obtained from $t$ by substituting $t_1, \ldots, t_n$ for $x_1, \ldots, x_n$, respectively, then we define $t \Rightarrow_M t'$. 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

$$yL(M, G) \triangleq \{w \in \Delta^* \mid \exists t \in R_2(G), q_0[t] \Rightarrow_M^* w\}$$

where $\Delta$ 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 \(\Delta\) 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 \(\alpha\) all the derivation steps which operate on \(t’\). This leads to the following new derivation which keeps \(t’\) un untouched;

\[
\alpha' : q_0[t] \Rightarrow^*_M w_1q_1[t’][w_2 \cdots w_nq_n[t’]w_{n+1}
\]

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

The state sequence of \(t’\) in derivation \(\alpha\) is defined to be \((q_1, \ldots, q_n)\). Derivation \(\alpha\) 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 \(w \in wL(M, G)\), 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 \(t\) in \(G\) 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, 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 \(\uparrow atr = 1\) \((atr \in N_{atr})\) 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:

\[
p : A \rightarrow B_1 B_2 \cdots B_q, \quad (4.2)
\]

where \(A \in N, B_1, B_2, \ldots, B_q \in N \cup T\). \(E_v\) 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 \(F = \{(atr_1, val_1), (atr_2, val_2), \ldots, (atr_k, val_k)\}\) where \(atr_1, atr_2, \ldots, atr_k\) are distinct attributes, and each of \(val_1, val_2, \ldots, val_k\) 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 \(p_0 : A \rightarrow B_1 \ldots B_q \in P_0\) is applied at \(v\), then \(F_v\) 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 \(i\)th 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 \(F_{sv}\), 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 \(l_\alpha = r_\alpha\) of \(p\), the values of \(l_\alpha\) and \(r_\alpha\) 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 \(\alpha\) is said to be derived from \(A\) and written as \(A \Rightarrow_{G_0}^* \alpha\). In this case, the tree \(t\) is called a derivation tree of \(A \Rightarrow_{G_0}^* \alpha\). We also call \(t\) a derivation tree (of \(A\)) in \(G\) simply.

The language generated by an nc-lfg \(G\), denoted by \(L(G)\), is defined as \(L(G) = \{w \in T^* | S \Rightarrow_{G_0}^* w\}\).

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

- $p_{11} : S \rightarrow A$ \{\top count = 1\} \{\top B \text{count} = 1\},
- $p_{12} : A \rightarrow a$ \{\top count = 1\} \{\top B \text{count} = 1\},
- $p_{13} : B \rightarrow c$ \{\top B \text{count} = 1\} \{\top d\},
- $p_{14} : \{\top A \text{count} = e\} \rightarrow \epsilon$,
- $p_{15} : \{\top A \text{count} = e\} \rightarrow \epsilon$.

The language generated by $G_{XS}$ is $L(G_{XS}) = \{a^n b^n c^n d^n \mid n \geq 0\}$.

**Example 4.2:** Let $G_{X4} = (N,T,P,S,N_{atr},A_{atm})$ be an nc-lfg where $N = \{S\}$, $T = \{a\}$, $N_{atr} = \{\log\}$, $A_{atm} = \{e\}$, and productions in $P$ are;

- $p_{21} : S \rightarrow \{\top log = 1\}$ \{\top log = 1\},
- $p_{22} : \{\top log = e\} \rightarrow a$.

The language generated by $G_{X4}$ is $L(G_{X4}) = \{a^n \mid n \geq 0\}$.

**Example 4.3:** Let $G_{X5} = (N,T,P,S,N_{atr},A_{atm})$ be an nc-lfg where $N = \{S,S',A,B\}$, $T = \{\text{the}, \text{woman}, \text{men}, \text{and}, \text{drinks}, \text{smoke}, \text{respectively}\}$, $N_{atr} = \{\text{num}, \text{list}\}$, $A_{atm} = \{\text{sg}, \text{pl}, \text{nil}\}$, and productions in $P$ are;

- $p_{30} : S \rightarrow S'$ \{\top list = 1\} \{\top list = 1\},
- $p_{31} : S' \rightarrow \text{the woman and } A \text{ drinks and } B$ \{\top num = \text{sg}\} \{\top list = 1\} \{\top list = 1\},
- $p_{32} : S' \rightarrow \text{the man and } A \text{ smoke and } B$ \{\top num = \text{pl}\} \{\top list = 1\} \{\top list = 1\},
- $p_{33} : A \rightarrow \text{the woman and } A$ \{\top num = \text{sg}\} \{\top list = 1\},
- $p_{34} : A \rightarrow \text{the man and } A$ \{\top num = \text{pl}\} \{\top list = 1\},
- $p_{35} : \{\top num = \text{sg}\} \{\top list = \text{nil}\} \rightarrow \text{the woman}$,
- $p_{36} : \{\top num = \text{pl}\} \{\top list = \text{nil}\} \rightarrow \text{the man}$,
- $p_{37} : B \rightarrow \text{drinks and } B$ \{\top num = \text{sg}\} \{\top list = 1\},
- $p_{38} : B \rightarrow \text{smoke and } B$ \{\top num = \text{pl}\} \{\top list = 1\},
- $p_{39} : B \rightarrow \text{drinks}$ \{\top num = \text{sg}\} \{\top list = \text{nil}\},
- $p_{40} : B \rightarrow \text{smoke}$ \{\top num = \text{pl}\} \{\top list = \text{nil}\},
- $p_{41} : A \rightarrow \text{the woman}$ \{\top num = \text{sg}\} \{\top list = \text{nil}\},
- $p_{42} : B \rightarrow \text{drinks}$ \{\top num = \text{pl}\} \{\top list = \text{nil}\},
- $p_{43} : B \rightarrow \text{smoke}$ \{\top num = \text{sg}\} \{\top list = \text{nil}\}.

$G_{X5}$ 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. $\{\top atr = \text{val}_1, \top atr = \text{val}_2\} \subseteq X$ for some $\text{atr} \in N_{atr}$ and some $\text{val}_1, \text{val}_2 \in A_{atm}$ such that $\text{val}_1 \neq \text{val}_2$.

2. $\{\top atr = \text{val}, \top atr = 1\} \subseteq X$ for some $\text{atr} \in N_{atr}$ and some $\text{val} \in A_{atm}$.

Productions $p_1, \ldots, p_n$ are consistent iff $\bigcup_{1 \leq i \leq n} E_{(i)}$ is consistent where $E_{(1)}$ is the set of functional schemata of $p_1$. 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 $A \rightarrow A_1$ and $A \rightarrow A_2$ whose left-hand sides are the same are inconsistent.

Suppose $G = (N,T,P,S,N_{atr},A_{atm})$ is an nc-lfg. Let $\{e_1, e_2, \ldots, e_n\}$ denote the multiset which consists of elements $e_1, e_2, \ldots, e_n$ 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 $\{\{A_1, A_2, \ldots, A_k\}\}$ is an SPN multiset. Let $A_1 \rightarrow A_1,$
... $A_h \rightarrow \alpha_h$ be consistent productions. For each $\alpha \in N_{\text{atr}}$, let $M_{\text{atr}}$ be the multiset consisting of all the nonterminals which appear in $\alpha_1, \ldots, \alpha_h$ and have an $S$ schema $\uparrow \alpha \downarrow = \iota$. If $M_{\text{atr}}$ is not empty, then $M_{\text{atr}}$ 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 $G_{EX\land}$ 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 $G_{EX\land}$ are $\{\{S\}\}$ and $\{\{A, B\}\}$. Hence $G_{EX\land}$ is a dc-lfg and is also an fc-lfg by the similar reason. Similarly, $G_{EX\land}$ in Example 4.2 is a dc-lfg. SPN multisets in $G_{EX\land}$ are $\{\{S\}\}$, $\{\{S, S\}\}$, $\{\{S, S, S, S\}\}$, ... Hence $G_{EX\land}$ is not an fc-lfg.

**NOTE :** $L(G_{EX\land})$ 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).

**5 Overview of the Results**

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

\[
\begin{align*}
\{a^n &a_2^n a_3^n a_4^n a_5^n \mid n \geq 0\} \not\subseteq D - E \\
\{a^n a_2^n \cdots a_{2m-1}^n a_{2m}^n \mid n \geq 0\} &\not\subseteq C - D \text{ for } m \geq 3, \\
&\text{(by Vijay-Shanker 1987.)} \\
\{a^{2^n} \mid n \geq 0\} &\not\subseteq B - C, \\
&\text{(by Kasami 1988a)(Seki 1991.)}
\end{align*}
\]

A relation $B \subseteq A$ 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 NP-complete (Nakanishi 1992), is in A. Hence, if $P \neq NP$, 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 $y\mathcal{L}_{\text{d-fts}} \subseteq \mathcal{L}_{\text{pmc}}$ 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 $Q = \{q_1, \ldots, q_t\}$, $T = \{a_1, \ldots, a_n\}$ 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.

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).

**Figure 2:** Inclusion relations between classes of languages. (1) : The class of language generated by icfr's is equal to C. (2) : The class of language generated by head grammars is equal to D.
Property 6.1: There is \((a_1, \ldots, a_t) \in L_G(R_h)\) (resp. \(L_G(A_h)\)) such that
\[
\begin{align*}
\text{if and only if there is a derivation tree } t \text{ of } G \text{ such that the root is } & p_h \text{ (resp. } a_h) \text{ and } \\
\{q_{a_j}[t] \Rightarrow_{M} \alpha_{a_j} \mid 1 \leq j \leq u\} \\
\{\text{no output is derived from } q_{a_j}[t] \mid 1 \leq j \leq v\}.
\end{align*}
\]
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 \(i\)th 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.
\[
RS(X) = \begin{cases} \{R_h \mid \text{the left-hand side of } p_h \text{ is } X\} & \text{if } X \in N \\ \{A_h\} & \text{if } X = a_h \in T. \end{cases}
\]
Productions and functions are defined as follows.

Step 1: For each production \(p_h : Y_0 \rightarrow Y_1 \cdots Y_k\) \((Y_0 \in N, Y_u \in N \cup T \text{ for } 1 \leq u \leq k)\) of cfg \(G\), construct nonterminating productions
\[
R_h \rightarrow f_{p_h}[Z_1, \ldots, Z_k]
\]
for every \(Z_u \in RS(Y_u)\) \((1 \leq u \leq k)\), 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
  \[
  f_{p_h}^{[i]}[x_1, \ldots, x_k] \triangleq b, \quad (6.3)
  \]
- if \(M\) has a rule \(q_i[p_h(x_1, \ldots, x_k)] \rightarrow \alpha_{i,0} q_{r(i,0)}[x_{\mu(i,0)}] \alpha_{i,1} \cdots \alpha_{i,n_i-1} q_{r(i,n_i-1)}[x_{\mu(i,n_i-1)}] \alpha_{i,n_i}\), then
  \[
  f_{p_h}^{[i]}[x_1, \ldots, x_k] \triangleq \alpha_{i,0} x_{\mu(i,0)} \alpha_{i,1} \cdots \alpha_{i,n_i-1} x_{\mu(i,n_i-1)} \alpha_{i,n_i}, \quad (6.4)
  \]
where \(x_u = (x_{u1}, \ldots, x_{ut})\) \((1 \leq u \leq k)\).

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 \(a_h \in T_i\), construct a terminating production \(A_h \rightarrow f_{a_h}\) where \(f_{a_h}\) is defined as follows: For \(1 \leq i \leq \ell\),

- if \(M\) has no rule whose left-hand side is \(q_i[a_h]\), then \(f_{a_h} \triangleq b\).
- if \(M\) has a rule \(q_i[a_h] \rightarrow \alpha_i\), then \(f_{a_h}^{[i]} \triangleq \alpha_i\).

Step 3: For each \(R_h \in RS(S)\), construct \(S' \rightarrow f_{\text{first}}[R_h]\) where \(f_{\text{first}}[x_1, \ldots, x_l] \triangleq x_1\). 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: \(y_{\mathcal{L}_{d,fts}} \subseteq \mathcal{L}_{\text{pmcfg}}\).

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

Theorem 6.2: \(y_{\mathcal{L}_{d,fts}} = \mathcal{L}_{\text{pmcfg}}\).

6.2 Nondeterministic \(\mathcal{L}_{d,fts}\)
In this section, the generative capacity of nondeterministic \(\mathcal{L}_{d,fts}\) is investigated, from the viewpoint of computational complexity. We have already shown that \(y_{\mathcal{L}_{d,fts}} = \mathcal{L}_{\text{pmcfg}}\), 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 \(\mathcal{L}_{d,fts}\) that generates an \(NP\)-complete language. In the following, a language called \(\text{Unary-3SAT}\), which is \(NP\)-complete (Nakanishi 1992), is considered, and then it is shown to belong to \(y_{\mathcal{L}_{d,fts}}\).

A \(\text{Unary-3CNF}\) is a (nonempty) \(3\)CNF in which the subscripts of variables are represented in unary. A positive literal \(x_i\) in a \(3\)CNF is represented by \(1i\) in a \(\text{Unary-3CNF}\). Similarly, a negative literal \(-x_i\) is represented by \(12i\). For example, a \(3\)CNF
\[(x_1 \lor x_2 \lor -x_3) \land (x_4 \lor -x_1 \lor -x_2)\]
is represented by a \(\text{Unary-3CNF}\)
\[
1\$11\$11# \land 11\$1\#11#.
\]
\(\text{Unary-3SAT}\) is the set of all satisfiable \(\text{Unary-3CNF}\)s.

Next, we construct a nondeterministic \(\mathcal{L}_{d,fts}\) \((M, G)\) that generates \(\text{Unary-3SAT}\). Define a cfg \(G = (N, T, P, S)\) where \(N = \{S, T, F\}\), \(T = \{e\}\) and the productions in \(P\) are as follows:
\[
\begin{align*}
r_{SS} & : S \rightarrow S \\
r_{TT} & : T \rightarrow T \\
r_{SF} & : S \rightarrow F \\
r_{TF} & : T \rightarrow F \\
r_{FE} & : F \rightarrow e
\end{align*}
\]
and the productions in \(P\) are as follows:
\[
\begin{align*}
r_{SS} & : S \rightarrow S \\
r_{TT} & : T \rightarrow T \\
r_{SF} & : S \rightarrow F \\
r_{TF} & : T \rightarrow F \\
r_{FE} & : F \rightarrow e
\end{align*}
\]
Let $M = (Q, \Sigma, \Delta, q_0, R)$ where
\[ Q = \{q_0, q_c, q_t, q_a\}, \]
\[ \Sigma = \{\sigma_S, \ldots, \sigma_F\}, \]
\[ \Delta = \{1, \land, \$, \#\}. \]

Since there are many rules in $R$, we will use an abbreviated notation. For example, following four rules
\[ q_a[r_{Te}(x)] \rightarrow 15, \]
\[ q_t[r_{Te}(x)] \rightarrow 15, \]
\[ q_o[r_{Fe}(x)] \rightarrow 15, \]
\[ q_a[r_{Fe}(x)] \rightarrow 15, \]
are abbreviated as "$q_a[r_{Te}(x)] = q_t[r_{Te}(x)] = 15$". By using this notation, the rules in $R$ are defined as follows.
\[ q_0[r_{SS}(x)] \rightarrow q_0[x] \land q_0[z] \]
\[ q_0[r_{ST}(x)] = q_0[r_{SP}(x)] = q_0[r_{SP}(x)] = q_0[r_{ST}(x)] = q_0[r_{SP}(x)] \]
\[ q_t[r_{TT}(x)] = q_t[r_{TF}(x)] = q_t[r_{TF}(x)] = q_t[r_{TF}(x)] = q_t[r_{TF}(x)] \]
\[ q_a[r_{Te}(x)] = q_a[r_{Te}(x)] = q_a[r_{Te}(x)] = q_a[r_{Te}(x)] = q_a[r_{Te}(x)] \]

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

### 7 Equivalence of $L_{nc-lfg}$ and $yL_{fts}$

First, we show $L_{nc-lfg} \subseteq yL_{fts}$. For a given $nc$-lfg $G = (N, T, P, S, N_{at}, A_{atm})$, 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 $F = \{(\text{atr}_1, F_1), \ldots, (\text{atr}_n, F_n)\}$. $F$ is represented by a derivation tree $\tau = p_{sp}(\tau_1, \ldots, \tau_n)$ in $G'$, where $\tau_i (1 \leq i \leq n)$ 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 $\tau$ into the yield of $t$, i.e., the terminal string obtained by concatenating the labels of leaves, in a top-down way.

**Step 1:** $G' = (N', \{d\}, P', S')$, where $N' = \{S_{sp} | sp \in SP\} \cup \{S\}$ and
\[ P' = \{p_{sp}' : S_{sp} \rightarrow S' \cdots S'\} \]
\[ \cup \{p_{guess}' : S' \cdots S' | sp \in SP\} \]
\[ \cup \{p_{term} : S' \rightarrow d \cdots d\}. \]

For a derivation tree $\tau$ in $G'$ and a node $v$ where $p_{sp}'$ is applied, the subtree rooted by the $i$th child of $v$ represents the value of attribute $\text{atr}_i$.

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

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

(i) $q_j[p_{guess}(x)] \rightarrow q_j(x) (1 \leq j \leq m)$ belongs to $R$ for each $sp \in SP$.

(ii) Let $\tau$ be a derivation tree in $G'$. When $p_{sp}'$ is the production applied at the root of $\tau$ and a state of $M$ is $q_{0o}$, $M$ chooses a production $p$ whose left-hand side is $A_{0o}$ 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 $SP$
\[ p : A_{0o} \rightarrow \alpha_0 A_{0o} \alpha_1 \ldots \alpha_{L-1} A_{0o} \alpha_L \]
\[ Ev \{ \uparrow atr_{v_1} = 1 \} \ldots \{ \uparrow atr_{v_L} = 1 \} \]

where $A_{0o} \in N$ and $\alpha_l \in T^* (0 \leq l \leq L)$, the following rule belongs to $R$:
\[ q_{0o}[p_{guess}(x)] \rightarrow q_{0o}[x] (1 \leq j \leq m) \]

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

Next, $yL_{fts} \subseteq L_{nc-lfg}$ 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 $Q = \{q_1, q_2, \ldots, q_m\}$. Let $n$ be the maximum length of the right-hand side of a production in $P$. Define an $nc$-lfg $G' = (N', \Delta, P', S')$ as follows.

**Step 1:** $N' = \{C_{[j]} | G \in N, 1 \leq j \leq m\} \cup \{a_{[j]} | a \in T, 1 \leq j \leq m\}$
\[ S' = S_{[1]}, \]
\[ N_{at} = \{atr_{i} | 1 \leq i \leq n\} \cup \{\text{rule}\}, \]
\[ A_{atm} = \{p | p\} \text{ is the label of a production in } P\} \]

A derivation tree $\tau = p(t_1, \ldots, t_h)$ in $G$ is represented by an f-structure $\{\{\text{rule}, p\}, (atr_1, F_1), \ldots, (atr_h, F_h)\}$ of $G'$ where $F_i (1 \leq i \leq h)$ 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 $a[j]$

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:

$$q_j[p(x_1, \cdots, x_h)] \rightarrow \alpha_{q_j} \alpha_{j_1} \cdots \alpha_{j_L_j} [x_{j_1}, \cdots, x_{j_L_j}]$$

be a rule in $R$ where $\alpha_{q_j} \in \Delta^*$ $(0 \leq j \leq L_j)$, $q_{j_1} \in Q$, and $x_{j_1} \in \{x_1, \cdots, x_h\}$ $(1 \leq i \leq L_j)$.

Then, the following production belongs to $P'$:

$$C[j] \rightarrow \alpha_{q_j} X_{j_1}^{[n_1]} \cdots X_{j_h}^{[n_h]} \alpha_{j_1} \cdots \alpha_{j_L_j} \{\text{rule } = p\}$$

(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 TRANS 7.2, the following theorem is obtained. A formal proof is found in (Nakanishi 1993).

**Theorem 7.1:** $L_{nc-lfg} = yL_{fts}$.

**Corollary 7.2:** $L_{dc-lfg} = yL_{d-fts}$.

**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 $L_{fc-lfg}$ and $L_{mcfg}$**

To prove $L_{fc-lfg} \subseteq L_{mcfg}$, we give an algorithm which translates a given fc-lfg $G = (N,T,P,S,N_{atm},A_{atm})$ 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:** $N' = \{ \text{the set of nonterminals which has a one-to-one correspondence with the set of SPN multisets in } G \} = \{ \{S\}, \{A,B\} \}$ (for $G_{EX3}$ in Example 4.1) $P' = \emptyset$, and $F = \emptyset$.

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 $A_i$'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:

$$\begin{align*}
p_{12} & : A \rightarrow a \\
p_{13} & : B \rightarrow b
\end{align*}$$

By 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.
sides are A and B, and their S schemata are the same. Thus, we construct the following mcfg production:

\[(A, B) \rightarrow f_1 [(A, B)]\]

where \(f_1 [(x_1, x_2)] = (ax_1b, cx_2d)\).

Consider the following pair of productions as another example:

\[
\begin{align*}
\{ \rho_1 : p_1 : A &\rightarrow a \} \quad \{ \rho_2 : p_2 : B &\rightarrow A \} \\
\{ \rho_1 : p_1 : A &\rightarrow A \} \quad \{ \rho_2 : p_2 : B &\rightarrow C \}
\end{align*}
\]

\[
\{ \rho_1 : p_1 : A &\rightarrow C \} \quad \{ \rho_2 : p_2 : B &\rightarrow D \}
\]

The multiset of nonterminals in the right-hand sides are partitioned into \(M_1 = \{(A, B)\}\) (for \(\rho_1\)) and \(M_2 = \{(C, D)\}\) (for \(\rho_2\)). For \((p_1, p_2)\), the following mcfg production is constructed:

\[
(A, B) \rightarrow g [(A, B), (C, D, D)]
\]

where \(g [(x_1, x_2), (x_21, x_22)] = (ax_1b, cx_2d)\).

Example 8.1: TRANS 8 translates fc-lfg \(G_{1X3}\) in Example 4.1 into an equivalent mcfg \(G_{1X3}'\), \(F = \{f_1 [(x_1, x_2)] = x_1x_2, f_2 [(x_1, x_2)] = (ax_1b, cx_2d), f_3 = (e, e)\}\), and, \(P' = \{(S) \rightarrow f_0 [(A, B)], (A, B) \rightarrow f_1 [(A, B)], (A, B) \rightarrow f_2\}\).

Theorem 8.1: \(\mathcal{L}_{\text{mcfg}} = \mathcal{L}_{\text{fc-lfg}}\).

Proof: \(\mathcal{L}_{\text{fc-lfg}} \subseteq \mathcal{L}_{\text{mcfg}}\) 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., 20:190-202.

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, COMP92-94.

Kaplan, R. and Bresnan, J. 1982. "Lexical-Functional Grammar", The Mental Representation of Grammatical Relations, J.Bresnan (ed.), MIT press:173-291.

Kasami, T. et al. 1988a. "Generalized Context-Free Grammars and Multiple Context-Free Grammars", Trans. IEICE, J71-D-I, 5:759-765.

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-0202.

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", Proc. of 25th meeting of Assoc. Comput. Ling. :104-111.

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.

Vijay-Shanker, K., Weir, D.J. and Joshi, A.K. 1987. "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.