Restrictions on Monadic Context-Free Tree Grammars

Akio Fujiyoshi
Department of Computer and Information Sciences, Ibaraki University
4-12-1 Nakanarusawa, Hitachi, Ibaraki, 316-8511, Japan
fujiyoshi@cis.ibaraki.ac.jp

Abstract
In this paper, subclasses of monadic context-free tree grammars (CFTGs) are compared. Since linear, nondeleting, monadic CFTGs generate the same class of string languages as tree adjoining grammars (TAGs), it is examined whether the restrictions of linearity and nondeletion on monadic CFTGs are necessary to generate the same class of languages. Epsilon-freeness on linear, nondeleting, monadic CFTG is also examined.

1 Introduction
The context-free tree grammars (CFTGs) were introduced by W. C. Rounds (1970) as tree generating systems, the definition of which is a direct generalization of context-free grammars (CFGs) from strings to rooted, ordered, labeled trees. For the application of CFTGs to natural languages, many kinds of restrictions on CFTGs have been considered because the string languages generated by CFTGs are exactly indexed languages, whose emptiness problem and uniform membership problem are exponential time complete, i.e., nonrestricted CFTGs are formidable. One approach to define subclasses of CFTGs is to restrict the ranks of nonterminals. The rank of a nonterminal is a natural number assigned to each nonterminal by which the number of children of the node labeled by the nonterminal is fixed. Through this approach, the simplest model of CFTGs is regular tree grammars (RTGs) (Brainerd, 1969), where the ranks of nonterminals are all 0. The string languages generated by RTGs are the languages generated by context-free grammars (CFGs). Since recent research on natural languages has suggested that formalisms for natural languages need to generate a slightly larger class of languages than CFGs, this paper focuses on monadic CFTGs, where the ranks of nonterminals are either 0 or 1.

Another formalism of tree generating systems is tree adjoining grammars (TAGs) (Joshi et al., 1975; Joshi and Schabes, 1996; Abeillé and Rambow, 2000). TAGs have been widely studied relating them to natural languages, and it was shown that TAGs have the same generative power of string languages as other formalisms for natural languages developed independently such as head grammars, combinatorial categorial grammars and linear indexed grammars (Vijay-Shanker and Weir, 1994). It is also noteworthy that there are recognition algorithms for the string languages generated by TAGs that run in $O(n^6)$ and $O(M(n^2))$ time (Rajasekaran, 1996; Rajasekaran and Yooseph, 1998). From the view point of CFTG, the languages generated by TAGs were examined (Fujiyoshi and Kasai, 2000; Fujiyoshi, 2004; Mönnich, 1997), and it was shown that linear, nondeleting, monadic CFTGs generate the same class of string languages as TAGs and a strictly larger class of tree languages than TAGs. Linearity is a restriction on CFTGs that requires the number of occurrences of every variable in the right-hand side of a rule be no more than 1, and nondeletion requires all variables in the left-hand side of a rule occur at least once in the right-hand side. In other words, linear, nondeleting, monadic CFTGs are those with nonterminals of rank 0 and 1 only and with exactly one occurrence of a variable in every right-hand side of a rule for a nonterminal of rank 1.

In this paper, the subclasses of monadic CFTGs are compared to examine whether the restrictions of linearity and nondeletion on monadic CFTGs are necessary to generate the same class of string languages as TAGs. It is shown that nondeletion is unnecessary since for any linear, monadic CFTG, there exists an equivalent linear, nondeleting, monadic CFTG. On the other hand, it is shown that linearity is necessary since there exists a non-linear, monadic CFTG which is not weakly equivalent to any linear, monadic CFTG.

For the development of parsing algorithm, the property of epsilon-freeness is very important, and in this paper, epsilon-freeness on linear, monadic CFTGs is also considered. Epsilon-freeness is a restriction on grammars that requires no use of
epsilon-rules, that is, rules defined with the empty string. It is shown that for any linear, monadic CFTG, there exists an epsilon-free, linear, nondeleting, monadic CFTG that generate the same string language.

2 Preliminaries

In this section, some terms, definitions and former results which will be used in the rest of this paper are introduced.

2.1 Ranked Alphabets, Trees and Substitution

A ranked alphabet is a finite set of symbols in which each symbol is associated with a natural number, called the rank of a symbol. Let \( \Sigma \) be a ranked alphabet. For \( n \geq 0 \), it is defined that \( \Sigma_n = \{ a \in \Sigma \mid \text{the rank of } a \text{ is } n \} \).

The set \( T_\Sigma \) (trees over \( \Sigma \)) is the smallest set of strings over \( \Sigma \), parentheses and commas such that:

1. \( T_\Sigma \subseteq T_\Sigma \) and \( a \in \Sigma_n \) for some \( n \geq 1 \), then \( a(\alpha_1, \alpha_2, \ldots, \alpha_n) \in T_\Sigma \).

2. Let \( \lambda \) be the empty string. Let \( \varepsilon \) be the special symbol which may be contained in \( \Sigma_0 \). The yield of a tree is a function from \( T_\Sigma \) into \( \Sigma^* \) defined as follows. For \( \alpha \in T_\Sigma \), (1) if \( \alpha = a \in (\Sigma_0 - \{ \varepsilon \}) \), yield(\( \alpha \)) = \( a \), (1') if \( \alpha = \varepsilon \), yield(\( \alpha \)) = \( \lambda \), and (2) if \( \alpha = a(\alpha_1, \alpha_2, \ldots, \alpha_n) \) for some \( a \in \Sigma_n \) and \( \alpha_1, \alpha_2, \ldots, \alpha_n \in T_\Sigma \), yield(\( \alpha \)) = yield(\( \alpha_1 \)) \cdot yield(\( \alpha_2 \)) \cdot \ldots \cdot yield(\( \alpha_n \)).

2.2 Context-Free Tree Grammars

The context-free tree grammars (CFTGs) were introduced by W. C. Rounds (1970) as tree generating systems. The definition of CFTGs is a direct generalization of context-free grammars (CFGs).

A context-free tree grammar (CFTG) is a four-tuple \( G = (N, \Sigma, P, S) \), where:

- \( N \) and \( \Sigma \) are disjoint ranked alphabets of non-terminals and terminals, respectively.
- \( P \) is a finite set of rules of the form

\[
A(x_1, x_2, \ldots, x_n) \rightarrow \alpha
\]

with \( n \geq 0 \), \( A \in N \) and \( \alpha \in T_{N \cup \Sigma}(X_n) \). For \( A \in N_0 \), rules are written as \( A \rightarrow \alpha \) instead of \( A() \rightarrow \alpha \).

- \( S \), the initial nonterminal, is a distinguished symbol in \( N_0 \).

For a CFTG \( G \), the one-step derivation \( \Rightarrow \) is the relation on \( T_{N \cup \Sigma} \times T_{N \cup \Sigma} \) such that for a tree \( \alpha \in T_{N \cup \Sigma} \), if \( \alpha = \alpha' [A(\alpha_1, \alpha_2, \ldots, \alpha_n)] \) for some \( \alpha' \in T_{N \cup \Sigma}(\{ X_1 \}) \cap T_{N \cup \Sigma}(\{ X_1 \}) \), \( A \in N \) and \( \alpha_1, \alpha_2, \ldots, \alpha_n \in T_{N \cup \Sigma} \), and \( A(x_1, x_2, \ldots, x_n) \rightarrow \beta \) is in \( P \), then \( \alpha \Rightarrow [\beta[\alpha_1, \alpha_2, \ldots, \alpha_n]] \). Figure 2 is an example of a one-step derivation where the rule \( A(x) \rightarrow \beta \) is applied to the tree \( \alpha = \alpha'[A(\alpha'')] \) and the tree \( \alpha'[\beta[\alpha'']] \) is obtained.

An \((n\text{-step})\) derivation is a finite sequence of trees \( \alpha_0, \alpha_1, \ldots, \alpha_n \in T_{N \cup \Sigma} \) such that \( n \geq 0 \) and \( \alpha_0 \Rightarrow \alpha_1 \Rightarrow \cdots \Rightarrow \alpha_n \). When there exists a derivation \( \alpha_0, \alpha_1, \ldots, \alpha_n \), it is written that \( \alpha_0 \Rightarrow^{*} \alpha_n \) or \( \alpha_0 \Rightarrow^{*} \alpha_n \).

The tree language generated by \( G \) is the set \( L(G) = \{ \alpha \in T_\Sigma \mid S \Rightarrow^{*} \alpha \} \). The string language generated by \( G \) is \( L_s(G) = \{ \text{yield}(\alpha) \mid \alpha \in L(G) \} \). Note that \( L_s(G) \subseteq \{ \Sigma_0 - \{ \varepsilon \} \}^* \).
Let \( G \) and \( G' \) be CFTGs. \( G \) and \( G' \) are equivalent if \( L(G) = L(G') \). \( G \) and \( G' \) are weakly equivalent if \( L_S(G) = L_S(G') \).

### 2.3 Restrictions on CFTGs

A CFTG \( G = (N, \Sigma, P, S) \) is monadic if the rank of any nonterminal is 0 or 1, i.e., \( N = N_0 \cup N_1 \) and \( N_n = \emptyset \) for \( n \geq 2 \). \( G \) is linear if for any rule \( A(x_1, x_2, \ldots, x_n) \rightarrow \alpha \) in \( P \), \( \alpha \in T_{N \cup \Sigma}(\{X_n\}) \), and nondeleting if for any rule \( A(x_1, x_2, \ldots, x_n) \rightarrow \alpha \) in \( P \), \( \alpha \in T_{N \cup \Sigma}(\{X_n\}) \).

A CFTG \( G = (N, \Sigma, P, S) \) is epsilon-free if for any rule \( A(x_1, x_2, \ldots, x_n) \rightarrow \alpha \) in \( P \), the symbol \( \varepsilon \) doesn’t occur in \( \alpha \).

When \( G \) is monadic, all rules are either of the form \( A(x) \rightarrow \alpha \) with \( A \in N_1 \) and \( \alpha \in T_{N \cup \Sigma}(X_1) \) or of the form \( B \rightarrow \beta \) with \( B \in N_0 \) and \( \beta \in T_{N \cup \Sigma} \). When \( G \) is monadic, linear and nondeleting, for any rule \( A(x) \rightarrow \alpha \) with \( A \in N_1 \) in \( P \), there exists exactly one occurrence of \( x \) in \( \alpha \).

For linear, nondeleting, monadic CFTGs, the following results are known.

**Theorem 2.1** (Fujiyoshi and Kasai, 2000) The class of string languages generated by linear, nondeleting, monadic CFTGs coincides with the class of string languages generated by TAGs.

**Theorem 2.2** (Fujiyoshi and Kasai, 2000) For any linear, nondeleting, monadic CFTG, there exists a weakly equivalent linear, nondeleting, monadic CFTG \( G = (N, \Sigma, P, S) \) that satisfies the following conditions:

- For any \( a \in \Sigma \), the rank of \( a \) is either 0 or 2.
- For each \( A \in N_0 \), if \( A \rightarrow \alpha \) is in \( P \), then either \( \alpha = a \) with \( a \in \Sigma_0 \), or \( \alpha = B(C) \) with \( B \in N_1 \) and \( C \in N_0 \). See (1) and (2) in Figure 3.

- For each \( A \in N_1 \), if \( A \rightarrow \alpha \) is in \( P \), then \( \alpha \) is one of the following forms:
  \[ \alpha = B(C(x)) \]  with \( B, C \in N_1 \),
  \[ \alpha = b(C, x) \]  with \( b \in \Sigma_2 \) and \( C \in N_0 \), or
  \[ \alpha = b(x, C) \]  with \( b \in \Sigma_2 \) and \( C \in N_0 \).

See (3), (4) and (5) in Figure 3.

If a linear, nondeleting, monadic CFTG satisfies the condition of Theorem 2.2, it is said that the grammar is in **strong normal form**.

### 3 Linearity and Nondeletion on Monadic CFTGs

Because linear, nondeleting, monadic CFTGs generate the same class of string languages as TAGs, the question is whether the restrictions of linearity and nondeletion on monadic CFTGs are necessary to generate the same class of languages. First, it will be shown that nondeletion is unnecessary.

**Theorem 3.1** For any linear, monadic CFTG \( G \), there exists an equivalent linear, nondeleting, monadic CFTG \( G' \).

**Proof.** Let \( G = (N, \Sigma, P, S) \) be a linear, monadic CFTG. An equivalent linear, nondeleting, monadic CFTG \( G' = (N', \Sigma, P', S) \) can be constructed as follows.

The set of nonterminals is \( N' = N_0' \cup N_1' \) such that \( N_0' = N_0 \cup \{ \overline{A} | A \in N_1 \} \) and \( N_1' = N_1 \).

For the preparation of the definition of \( P' \), for \( \alpha \in T_{N \cup \Sigma}(X_1) \) we define \( \Pi(\alpha) \subseteq T_{N' \cup \Sigma}(X_1) \) as the smallest set satisfying the following conditions:

- \( \alpha \in \Pi(\alpha) \cup \Pi(\alpha) \)

\(^1\)We say “strong” because a grammar in this normal form only preserves weak equivalence.
The set of rules is defined as follows.

Figure 4: The three different cases

- If $\beta \in \Pi(\alpha)$ and $\beta = \beta'[B(\beta'')]$ for some $B \in N_1, \beta' \in T_{N\cup\Sigma}([X_1]) \cap T_{N\cup\Sigma}([X_1])$ and $\beta'' \in T_{N\cup\Sigma}$, then $\beta'[B] \in \Pi(\alpha)$.

The set of rules is defined as follows.

$$P' = \{ A \rightarrow \hat{\alpha} \mid A \in N_0, A \rightarrow \alpha \in P, \hat{\alpha} \in \Pi(\alpha) \}$$

$$\cup \{ (A(x) \rightarrow \hat{\alpha} \mid A \in N_1, A(x) \rightarrow \alpha \in P, \hat{\alpha} \in \Pi(\alpha) \cap T_{N\cup\Sigma}([X_1]) \}$$

$$\cup \{ \overrightarrow{A} \rightarrow \hat{\alpha} \mid A \in N_1, A(x) \rightarrow \alpha \in P, \hat{\alpha} \in \Pi(\alpha) \cap T_{N\cup\Sigma} \}$$

Because of the construction of $N'$ and $P'$, $G'$ is monadic and nondeleting.

To show the equivalence of $G$ and $G'$, we prove the following statement holds for any $\alpha \in T_{N\cup\Sigma}$ and $\beta \in T_{\Sigma}$ by induction on the length of derivations:

$$\frac{\alpha \xrightarrow{G} \beta}{\alpha \xrightarrow{G'} \beta} \text{ if and only if there exists } \hat{\alpha} \in \Pi(\alpha) \text{ such that } \frac{\hat{\alpha} \xrightarrow{G'} \beta}{\hat{\alpha} \xrightarrow{G} \beta}.$$
monadic CFTGs to generate the same class of languages. The answer is negative. The following example is a non-linear, monadic CFTG that generates a string language that no linear, monadic CFTG can generate.

Example 3.2 The following is an example of a non-linear, monadic CFTG that generates the string language $L_{w^4} = \{w w w w | w \in \{a, b\}^+\}$. $G = (N, \Sigma, P, S)$ where $N = \{S, A\}$, the ranks of $S$ and $A$ are 0 and 1, respectively, $\Sigma = \{a, b, c, d\}$, the ranks of $a, b, c$ and $d$ are 0, 0, 2 and 4, respectively, and $P$ consists of the following rules:

$$
S \rightarrow A(a), \quad S \rightarrow A(b), \quad A(x) \rightarrow d(xxxx), \\
A(x) \rightarrow A(c(xa)), \quad A(x) \rightarrow A(c(xb)).
$$

Because $G$ has the rule $A(x) \rightarrow d(xxxx)$, $G$ is not linear.

Theorem 3.3 There exists a monadic CFTG which is not weakly equivalent to any linear, monadic CFTG.

Proof. It is known that the string language $L_{w^4}$ in Example 3.2 cannot be generated by any TAG. It cannot be generated by any linear, monadic CFTG, neither.

4 Epsilon-Freeness on Linear, Monadic CFTGs

According to our definition of CFTGs, they are allowed to generate trees with the special symbol $\varepsilon$, which is treated as the empty string while taking the yields of trees. In this section, it will be seen that for any linear, monadic CFTG, there exists a weakly equivalent epsilon-free, linear, nondeleting, monadic CFTG. Because any epsilon-free CFTG cannot generate a tree with $\varepsilon$, it is clear that for a CFTG with epsilon-rules, there generally doesn't exist an equivalent epsilon-free CFTG.

Theorem 4.1 For any linear, monadic CFTG $G = (N, \Sigma, P, S)$, if $\lambda \not\in L_S(G)$, then there exists a weakly equivalent epsilon-free, linear, nondeleting, monadic CFTG $G'$. If $\lambda \in L_S(G)$, then there exists $G'$ whose epsilon-rule is only $S \rightarrow \varepsilon$.

Proof. Since it is enough to show the existence of a weakly equivalent grammar, without loss of generality, we may assume that $G$ is in strong normal form. We may also assume that the initial nonterminal $S$ doesn't appear in the right-hand side of any rule in $P$.

We first construct subsets of nonterminals $E_0$ and $E_1$ as follows. For initial values, we set $E_0 = \{A \in N_0 | A \rightarrow \varepsilon \in P\}$ and $E_1 = \emptyset$. We repeat the following operations to $E_0$ and $E_1$ until no more operations are possible:

- If $A \rightarrow B(C)$ with $B \in E_1$ and $C \in E_0$ is in $P$, then add $A \in N_0$ to $E_0$.
- If $A(x) \rightarrow b(C, x)$ with $C \in E_0$ is in $P$, then add $A \in N_1$ to $E_1$.
- If $A(x) \rightarrow b(x, C)$ with $C \in E_0$ is in $P$, then add $A \in N_1$ to $E_1$.
- If $A(x) \rightarrow B(C(x))$ with $B, C \in E_1$ is in $P$, then add $A \in N_1$ to $E_1$.

In the result, $E_0$ satisfies the following.

$E_0 = \{A \in N_0 | \exists \alpha \in T_\Sigma, A \xrightarrow{f} \alpha, \text{yield}(\alpha) = \lambda\}$

We construct $G' = (N', \Sigma', P', S)$ as follows. The set of nonterminals is $N' = N_0' \cup N_1'$ such that $N_0' = N_0 \cup \{A | A \in N_1\}$ and $N_1' = N_1$. The set of terminal is $\Sigma' = \Sigma \cup \{c\}$, where $c$ is a new symbol of rank 1. The set of rules $P'$ is the smallest set satisfying following conditions:

- $P'$ contains all rules in $P$ except rules of the form $A \rightarrow \varepsilon$.
- If $S \in E_0$, then $S \rightarrow \varepsilon$ is in $P'$.
- If $A \rightarrow B(C)$ is in $P$ and $C \in E_0$, then $A \rightarrow \overline{B}$ is in $P'$.
- If $A(x) \rightarrow B(C(x))$ is in $P$, then $\overline{A} \rightarrow B(\overline{C})$ is in $P'$.
- If $A(x) \rightarrow b(C, x)$ or $A(x) \rightarrow b(x, C)$ is in $P$ and $C \in E_0$, then $A(x) \rightarrow c(x)$ is in $P'$.
- If $A(x) \rightarrow b(C, x)$ or $A(x) \rightarrow b(x, C)$ is in $P$, then $\overline{A} \rightarrow c(C)$ is in $P'$.

To show $L_S(G') = L_S(G)$, we prove the following (i), (ii) and (iii) hold by induction on the length of derivations:

(i) For $A \in N_0$, $A \xrightarrow{G'} \alpha'$ and $\alpha' \in T_\Sigma$ if and only if $A \xrightarrow{G} \alpha$ for some $\alpha \in T_\Sigma$ such that $\text{yield}(\alpha) = \text{yield}(\alpha') = \lambda$.

(ii) For $A \in N_1$, $A(x) \xrightarrow{G'} \alpha'$ and $\alpha' \in T_\Sigma(X_1)$ if and only if $A(x) \xrightarrow{G} \alpha$ for some $\alpha \in T_\Sigma(X_1)$ such that $\text{yield}(\alpha) = \text{yield}(\alpha')$.

(iii) For $\overline{A} \in N'_0 - N_0$, $\overline{A} \xrightarrow{G'} \alpha'$ and $\alpha' \in T_\Sigma$ if and only if $A(x) \xrightarrow{G} \alpha$ for some $\alpha \in T_\Sigma(X_1)$ such that $\text{yield}(\alpha[\varepsilon]) = \text{yield}(\alpha') = \lambda$. 

We start with “only if” part. For 0-step derivations, (i), (ii) and (iii) clearly hold since there doesn’t exist $\alpha' \in T_S$ nor $\alpha' \in T_S(1)$ for each statement.

We consider the cases for 1-step derivations.

[Proof of (i)] If $A \not\vdash \alpha'$ and $\alpha' \in T_S$, then $\alpha' = a$ for some $a \in \Sigma_0$ and the rule $A \rightarrow a$ in $P'$ has been used. Therefore, $A \rightarrow a$ is in $P$ and $A \not\vdash a$.

[Proof of (ii)] If $A(X) \not\vdash \alpha'$ and $\alpha' \in T_S(X_1)$, then $\alpha' = c(x)$ and the rule $A(x) \rightarrow c(x)$ in $P'$ has been used. By the definition of $P'$, $A(x) \rightarrow b(C, x)$ or $A(x) \rightarrow b(x, C)$ is in $P$ for some $C \in E_0$. There exists $\gamma \in T_S$ such that $C \not\vdash \gamma$ and yield($\gamma$) = $\lambda$. Therefore, $A(X) \not\vdash b(x, C)$ or $A(X) \not\vdash x$. By the definition of $P'$, there exists $\beta' \in T_S(X_1)$ such that $B(X) \not\vdash \beta'$ and $C \not\vdash \gamma$. By the induction hypothesis of (ii), there exists $\gamma \in T_S$ such that $C \not\vdash \gamma$ and yield($\gamma$) = $\lambda$. Therefore, $A(X) \not\vdash \beta'$.

[Proof of (i)] If $A \not\vdash \alpha'$, then the rule used at the first step is one of the following form: (1) $A \rightarrow B(C)$ or (2) $A \rightarrow B$. In the case (1), $A \not\vdash B(C)$ and $\beta' = \alpha'$ for some $\beta' \in T_S(X_1)$ and $\gamma' \in T_S$ such that $B(X) \not\vdash \beta'$ and $C \not\vdash \gamma'$. By the induction hypothesis of (ii), there exists $\gamma \in T_S$ such that $C \not\vdash \gamma$ and yield($\gamma$) = $\lambda$. By the definition of $P'$, $A(X) \rightarrow b(C, x)$ or $A(X) \rightarrow b(x, C)$ is in $P$. Therefore, $A(X) \not\vdash b(C, x)$ or $A(X) \not\vdash b(x, C)$. Therefore, $A \not\vdash a$.

[Proof of (ii)] If $A \not\vdash \alpha'$, then the rule used at the first step is one of the following form: (1) $A \rightarrow B(C)$ or (2) $A \rightarrow c(C)$. In the case (1), $A \not\vdash B(C)$ and $\beta' = \alpha'$ for some $\beta' \in T_S(X_1)$ and $\gamma' \in T_S$ such that $B(X) \not\vdash \beta'$ and $C \not\vdash \gamma'$. By the induction hypothesis of (ii), there exists $\gamma \in T_S$ such that $C \not\vdash \gamma$ and yield($\gamma$) = $\lambda$. By the definition of $P'$, $A(X) \rightarrow b(C, x)$ or $A(X) \rightarrow b(x, C)$ is in $P$. Therefore, $A(X) \not\vdash b(C, x)$ or $A(X) \not\vdash b(x, C)$. Therefore, $A \not\vdash a$.

The “if” part is similarly proved as follows. For 0-step derivations, (i), (ii) and (iii) clearly hold since there doesn’t exist $\alpha \in T_S$ nor $\alpha \in T_S(X_1)$ for each statement.

The cases for 1-step derivations are proved.

[Proof of (i)] If $A \not\vdash \alpha$ and $\alpha \in T_S$, then $\alpha = a$ for some $a \in \Sigma_0$ and the rule $A \rightarrow a$ in $P$ has been used. Therefore, $A \rightarrow a$ is in $P'$ and $A \not\vdash a$.

[Proof of (ii)] If $A \not\vdash \alpha'$, then the rule used at the first step is one of the following form: (1) $A \rightarrow B(C)$ or (2) $A \rightarrow B$. In the case (1), $A \not\vdash B(C)$ and $\beta' = \alpha'$ for some $\beta' \in T_S(X_1)$ and $\gamma' \in T_S$ such that $B(X) \not\vdash \beta'$ and $C \not\vdash \gamma'$. By the induction hypothesis of (ii), there exists $\gamma \in T_S$ such that $C \not\vdash \gamma$ and yield($\gamma$) = $\lambda$. By the definition of $P'$, $A(X) \rightarrow b(C, x)$ or $A(X) \rightarrow b(x, C)$ is in $P$. Therefore, $A(X) \not\vdash b(C, x)$ or $A(X) \not\vdash b(x, C)$. Therefore, $A \not\vdash a$.

For $k \geq 2$, assume that (i), (ii) and (iii) holds for any derivation of length less than $k$.

[Proof of (ii)] If $A \not\vdash \alpha'$, then the rule used at the first step is one of the following form: (1) $A \rightarrow B(C)$ or (2) $A \rightarrow B$. In the case (1), $A \not\vdash B(C)$ and $\beta' = \alpha'$ for some $\beta' \in T_S(X_1)$ and $\gamma' \in T_S$ such that $B(X) \not\vdash \beta'$ and $C \not\vdash \gamma'$. By the induction hypothesis of (ii), there exists $\gamma \in T_S$ such that $C \not\vdash \gamma$ and yield($\gamma$) = $\lambda$. By the definition of $P'$, $A(X) \rightarrow b(C, x)$ or $A(X) \rightarrow b(x, C)$ is in $P$. Therefore, $A(X) \not\vdash b(C, x)$ or $A(X) \not\vdash b(x, C)$. Therefore, $A \not\vdash a$.

[Proof of (ii)] If $A \not\vdash \alpha''$, then the rule used at the first step is one of the following form: (1) $A \rightarrow B(C)$ or (2) $A \rightarrow B$. In the case (1), $A \not\vdash B(C)$ and $\beta' = \alpha'$ for some $\beta' \in T_S(X_1)$ and $\gamma' \in T_S$ such that $B(X) \not\vdash \beta'$ and $C \not\vdash \gamma'$. By the induction hypothesis of (ii), there exists $\gamma \in T_S$ such that $C \not\vdash \gamma$ and yield($\gamma$) = $\lambda$. By the definition of $P'$, $A(X) \rightarrow b(C, x)$ or $A(X) \rightarrow b(x, C)$ is in $P$. Therefore, $A(X) \not\vdash b(C, x)$ or $A(X) \not\vdash b(x, C)$. Therefore, $A \not\vdash a$.
exists ε-freeness can be assumed when their gener-

Therefore, the induction hypothesis of (ii), there exists

yield the induction hypothesis of (ii), there exists

5 Conclusions

[Proof of (ii)] If \( A(x) \xrightarrow{\beta} \alpha \), then the rule used at

the first step is one of the following form: (1)

\( A(x) \rightarrow B(C(x)) \), (2) \( A(x) \rightarrow b(C, x) \) or (3)

\( A(x) \rightarrow b(x, C) \). The proof of the case (1) is di-

rect from the induction hypothesis. In the case (2),

\( A(x) \xrightarrow{\beta} b(C, x) \xrightarrow{\gamma} b(\gamma, x) = \alpha \) for some \( \gamma \in T_{\Sigma} \)
such that \( C \xrightarrow{\beta} \gamma \). Here, we have to think of the two
cases: (a) \( \text{yield}(\gamma) \neq \lambda \) and (b) \( \text{yield}(\gamma) = \lambda \).

(a) If \( \text{yield}(\gamma) \neq \lambda \), then by the induction hypot-

thesis of (i), there exists \( \gamma' \in T_{\Sigma} \) such that

\( C \xrightarrow{\beta} \gamma' \) and \( \text{yield}(\gamma') = \text{yield}(\gamma) \). By the def-

inition of \( P' \), \( A(x) \rightarrow b(C, x) \) is in \( P' \). Therefore,

\( A(x) \xrightarrow{\beta} b(C, x) \xrightarrow{\gamma} b(\gamma, x) \) =

\( \text{yield}(b(\gamma, x)) \). (b) If \( \text{yield}(\gamma) = \lambda \), then \( C \in E_0 \)

and \( A(x) \rightarrow c(x) \) is in \( P' \). Therefore, \( A(x) \xrightarrow{\beta} c(x) \)

and \( \text{yield}(c(x)) = \text{yield}(b(\gamma, x)) \). The proof of the
case (3) is similar to that of the case (2).

[Proof of (iii)] If \( A(x) \xrightarrow{\beta} \alpha \), then the rule used at

the first step is one of the following form: (1)

\( A(x) \rightarrow B(C(x)) \), (2) \( A(x) \rightarrow b(C, x) \) or (3)

\( A(x) \rightarrow b(x, C) \). In the case (1),

\( A(x) \xrightarrow{\beta} B(C(x)) \xrightarrow{\gamma} \beta[\gamma] = \alpha \) for some \( \beta, \gamma \in T_{\Sigma}(X_1) \) such that \( B(x) \xrightarrow{\beta} \beta' \) and \( C \xrightarrow{\beta'} \gamma \). By

the definition of \( P' \), \( \overline{\alpha} \rightarrow B(C) \) is in \( P' \). By

the induction hypothesis of (ii), there exists \( \beta' \in T_{\Sigma}(X_1) \) such that \( B(x) \xrightarrow{\beta} \beta' \) and \( \text{yield}(\beta') = \text{yield}(\beta) \). By the induction hypothesis of (iii), there

exists \( \gamma' \in T_{\Sigma} \) such that \( C \xrightarrow{\beta'} \gamma' \) and \( \text{yield}(\gamma') = \text{yield}(\gamma|\varepsilon) \). Therefore, \( \overline{\alpha} \xrightarrow{\beta'} B(C) \xrightarrow{\gamma'} \beta'[\gamma'] \) and

\( \text{yield}(\beta'[\gamma']) = \text{yield}(\beta[\gamma|\varepsilon]) \). In the case (2),

\( A(x) \xrightarrow{\beta} b(C, x) \xrightarrow{\gamma} b(\gamma, x) = \alpha \) for some \( \gamma \in T_{\Sigma} \)
such that \( C \xrightarrow{\beta} \gamma \) and \( \text{yield}(\gamma) \neq \lambda \). By

the definition of \( P' \), \( \overline{\alpha} \rightarrow c(C) \) is in \( P' \). By

the induction hypothesis of (i), there exists \( \gamma' \in T_{\Sigma} \) such that \( C \xrightarrow{\beta'} \gamma' \) and \( \text{yield}(\gamma') = \text{yield}(\gamma) \).

Therefore, \( \overline{\alpha} \xrightarrow{\beta'} c(C) \xrightarrow{\gamma'} \gamma' \) and \( \text{yield}(\gamma') = \text{yield}(\gamma|\varepsilon) \). The proof of the case (3) is similar to

that of the case (2).

By (i), we have the result \( L_S(G') = L_S(G) \). □

5 Conclusions

In this paper, the desirable features of linear

monadic CFTGs have been discovered: the re-

striction of nondeletion doesn’t affect their gener-

ative power of tree languages, and the restriction of

epsilon-freeness can be assumed when their gener-

ation of string languages is considered. The key to

the proofs of this paper was the simplicity of the def-

inition of linear, monadic CFTGs and their normal

form.

Recently, the class of grammars called mildly

context-sensitive grammars has been studied very

actively, to which TAGs and other well-established

formalisms for natural languages belong. Since it

is not difficult to study formal properties of linear,

monadic CFTGs, they are helpful tools for the study of

mildly context-sensitive grammars.

References

Anne Abeillé and Owen Rambow, editors. 2000.

Tree adjoining grammars: formalisms, linguis-
ic analysis and processing. CSLI Publications,

Stanford, California.

Walter S. Brainerd. 1969. Tree generating regular

systems. Information & Control, 14(2):217–231.

Akio Fujiyoshi and Takumi Kasai. 2000. Spinal-

formed context-free tree grammars. Theory of

Computing Systems, 33(1):59–83.

Akio Fujiyoshi. 2004. Epsilon-free grammars and

lexicalized grammars that generate the class of

the mildly context-sensitive languages. In 7th In-
ternational Workshop on Tree Adjoining Gram-

mar and Related Formalisms: Proceedings of the

Workshop, Vancouver, pages 16–23.

Aravind K. Joshi and Yves Schabes. 1996. Handbook of Formal Languages, volume 3, chap-
ter Tree-adjoining grammars, pages 69–124.

Springer, Berlin.

Aravind K. Joshi, Leon S. Levy, and Masako Taka-

hashi. 1975. Tree adjunct grammars. J. Com-
puter & System Sciences, 10(1):136–163.

Uwe Möennich. 1997. Adjunction as substitution: an

algebraic formulation of regular, context-free and

tree adjoining languages. In G. V. Morrill G-

J. Kruijff and R. T. Oehrle, editors, Formal Gram-
mars 1997: Proceedings of the Conference, Aix-
en-Provence, pages 169–178.

Sanguthevar Rajasekaran and Shibu Yooseph.

1998. TAL recognition in \( O(M(n^2)) \) time. J.

Computer & System Sciences, 56(1):83–89.

Sanguthevar Rajasekaran. 1996. Tree-adjoining

language parsing in \( O(n^6) \) time. SIAM J. Com-

put., 25(4):862–873.

William C. Rounds. 1970. Mapping and gram-

mars on trees. Mathematical Systems Theory,

4(3):257–287.

K. Vijay-Shanker and David J. Weir. 1994. The

equivalence of four extensions of context-

free grammars. Mathematical Systems Theory,

27(6):511–546.