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, combinatory 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; Möennich, 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 CTFGs 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 Σ be a ranked alphabet. For $n \geq 0$, it is defined that $\Sigma_n = \{a \in \Sigma |$ the rank of a is $n\}$.

The set T_Σ (*trees* over Σ) is the smallest set of strings over Σ, parentheses and commas such that (1) $\Sigma_0 \subseteq T_\Sigma$ and (2) if $\alpha_1, \alpha_2, \ldots, \alpha_n \in T_\Sigma$ and $a \in \Sigma_n$ for some $n \geq 1$, then $a(\alpha_1, \alpha_2, \ldots, \alpha_n) \in T_\Sigma$.

Let λ be the empty string. Let ε be the special symbol that may be contained in Σ_0. The *yield* of a tree is a function from T_Σ into Σ^* 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$(\alpha_2) \cdots$ yield(α_n).

Let X be the fixed countable set of variables $x_1, x_2, \ldots$. It is defined that $X_0 = \emptyset$ and for $n \geq 1$, $X_n = \{x_1, x_2, \ldots, x_n\}$. x_1 is situationally denoted by x. $T_\Sigma(X_n)$ is defined to be $T_{\Sigma \cup X_n}$ taking the ranks of elements in X are all 0. For $\alpha \in T_\Sigma(X_n)$ and $\beta_1, \beta_2, \ldots, \beta_n \in T_\Sigma(X)$, $\alpha[\beta_1, \beta_2, \ldots, \beta_n]$ is defined to be the result of substituting each β_i $(1 \leq i \leq n)$ for the occurences of the variable x_i in α.

A tree $\alpha \in T_\Sigma(X_n)$ is *linear* if no variable occurs more than once in α, and *nondeleting* if all variables in X_n occur at least once in α. The set of all linear trees and all nondeleting trees in $T_\Sigma(X_n)$ are denoted by $T_\Sigma(\lceil X_n \rceil)$ and $T_\Sigma(\lfloor X_n \rfloor)$, respectively.

In this papaer, the conventional way of illustrating trees is used. See Figure 1. The tree $A(b(a), a, B(E, d))$ is illustrated as (1). An arbitrary tree $\alpha \in T_\Sigma$ is illustrated as (2). When the variables of a tree $\beta \in T_\Sigma(X_3)$ occur in the order of x_1, x_2, x_3, x_1, the tree is illustrated as (3).

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).

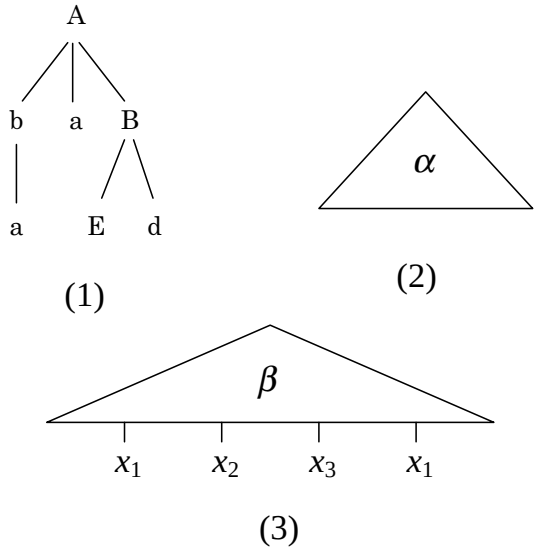

Figure 1: Trees

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

- N and Σ are disjoint ranked alphabets of *nonterminals* and *terminals*, respectively.
- P is a finite set of *rules* of the form

$$A(x_1, x_2, \ldots, x_n) \to \alpha$$

  with $n \geq 0$, $A \in N_n$ and $\alpha \in T_{N \cup \Sigma}(X_n)$. For $A \in N_0$, rules are written as $A \to \alpha$ instead of $A() \to \alpha$.
- S, the *initial nonterminal*, is a distinguished symbol in N_0.

For a CFTG $\mathcal{G}$, the *one-step derivation* $\underset{\mathcal{G}}{\Rightarrow}$ 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 $\alpha' \in T_{N \cup \Sigma}(\lceil X_1 \rceil) \cap T_{N \cup \Sigma}(\lfloor X_1 \rfloor)$, $A \in N_n$ and $\alpha_1, \alpha_2, \ldots, \alpha_n \in T_{N \cup \Sigma}$, and $A(x_1, x_2, \ldots, x_n) \to \beta$ is in P, then $\alpha \underset{\mathcal{G}}{\Rightarrow} \alpha'[\beta[\alpha_1, \alpha_2, \ldots, \alpha_n]]$. Figure 2 is an example of a one-step derivation where the rule $A(x) \to \beta$ is applied to the tree $\alpha = \alpha'[A(\alpha'')]$ and the tree $\alpha'[\beta[\alpha'']]$ is obtained.

An (n-*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 $\alpha_0 \underset{\mathcal{G}}{\Rightarrow} \alpha_1 \underset{\mathcal{G}}{\Rightarrow} \cdots \underset{\mathcal{G}}{\Rightarrow} \alpha_n$. When there exists a derivation $\alpha_0, \alpha_1, \ldots, \alpha_n$, it is writen that $\alpha_0 \overset{n}{\underset{\mathcal{G}}{\Rightarrow}} \alpha_n$ or $\alpha_0 \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \alpha_n$.

The *tree language generated by* $\mathcal{G}$ is the set $L(\mathcal{G}) = \{\alpha \in T_\Sigma \mid S \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \alpha\}$. The *string language generated by* $\mathcal{G}$ is $L_S(\mathcal{G}) = \{\text{yield}(\alpha) \mid \alpha \in L(\mathcal{G})\}$. Note that $L_S(\mathcal{G}) \subseteq (\Sigma_0 - \{\varepsilon\})^*$.

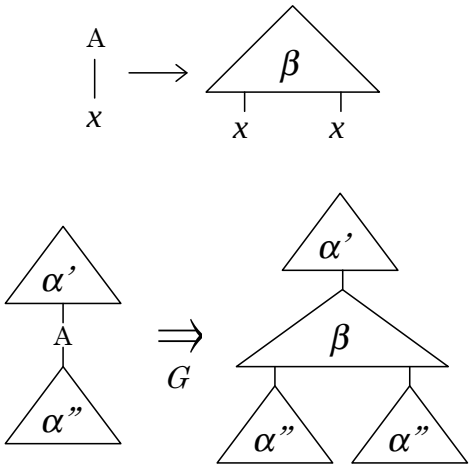

Figure 2: One-step derivation

Let $\mathcal{G}$ and $\mathcal{G}'$ be CFTGs. $\mathcal{G}$ and $\mathcal{G}'$ are *equivalent* if $L(\mathcal{G}) = L(\mathcal{G}')$. $\mathcal{G}$ and $\mathcal{G}'$ are *weakly equivalent* if $L_S(\mathcal{G}) = L_S(\mathcal{G}')$.

2.3 Restrictions on CFTGs

A CFTG $\mathcal{G} = (N, \Sigma, P, S)$ 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 $n \geq 2$. $\mathcal{G}$ is *linear* if for any rule $A(x_1, x_2, \ldots, x_n) \to \alpha$ in P, $\alpha \in T_{N\cup\Sigma}(\lceil X_n \rceil)$, and *nondeleting* if for any rule $A(x_1, x_2, \ldots, x_n) \to \alpha$ in P, $\alpha \in T_{N\cup\Sigma}(\lfloor X_n \rfloor)$.

A CFTG $\mathcal{G} = (N, \Sigma, P, S)$ is *epsilon-free* if for any rule $A(x_1, x_2, \ldots, x_n) \to \alpha$ in P, the symbol ε doesn't occur in α.

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

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 $\mathcal{G} = (N, \Sigma, P, S)$ 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 $A \to \alpha$ 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.

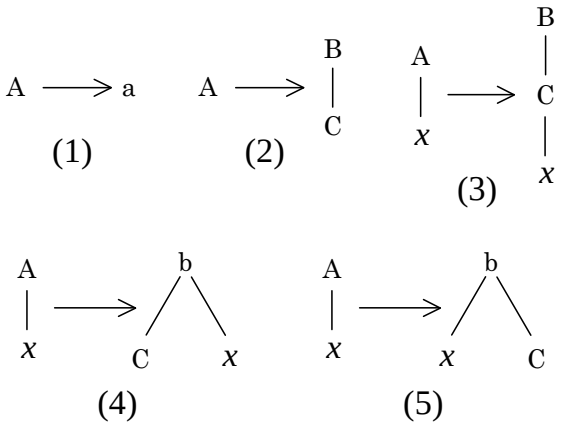

Figure 3: Strong normal form

- For each $A \in N_1$, if $A(x) \to \alpha$ is in P, then α 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*[1].

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 $\mathcal{G}$, there exists an equivalent linear, nondeleting, monadic CFTG $\mathcal{G}'$.*

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

The set of nonterminal is $N' = N'_0 \cup N'_1$ such that $N'_0 = N_0 \cup \{\overline{A} \mid A \in N_1\}$ and $N'_1 = N_1$. For the preparation of the definition of P', for $\alpha \in T_{N\cup\Sigma}(X_1)$ we define $\Pi(\alpha) \subset T_{N'\cup\Sigma}(X_1)$ as the smallest set satisfying the following conditions:

- $\alpha \in \Pi(\alpha)$D

[1]We say "strong" because a grammar in this normal form only preserves weak equivalence.

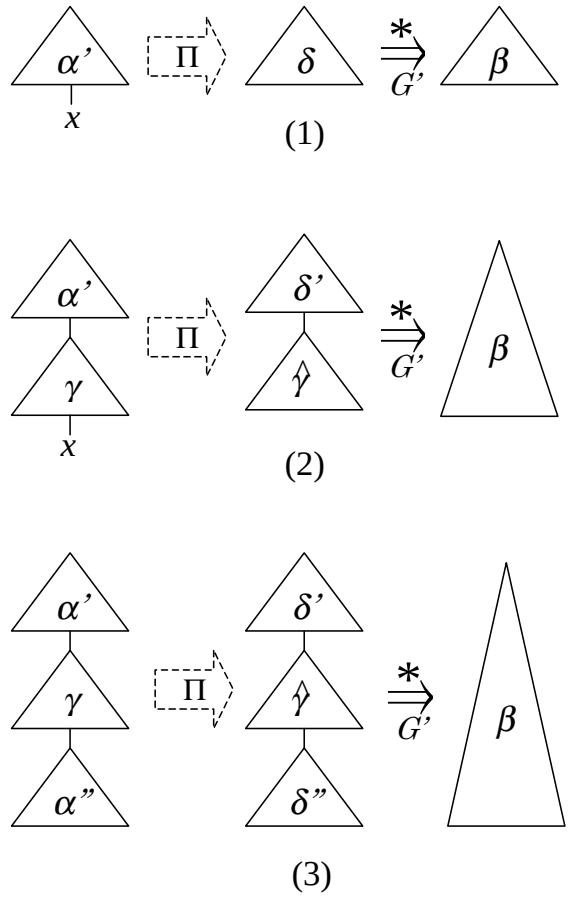

Figure 4: The three different cases

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

The set of rules is defined as follows.

$$\begin{aligned} P' = \{&A \to \hat{\alpha} \mid A \in N_0,\, A \to \alpha \in P,\, \hat{\alpha} \in \Pi(\alpha)\} \\ \cup\ \{&A(x) \to \hat{\alpha} \mid A \in N_1,\, A(x) \to \alpha \in P, \\ &\hat{\alpha} \in \Pi(\alpha) \cap T_{N' \cup \Sigma}(\lfloor X_1 \rfloor)\} \\ \cup\ \{&\overline{A} \to \hat{\alpha} \mid A \in N_1,\, A(x) \to \alpha \in P, \\ &\hat{\alpha} \in \Pi(\alpha) \cap T_{N' \cup \Sigma}\} \end{aligned}$$

Because of the construction of N' and P', $\mathcal{G}'$ is monadic and nondeleting.

To show the equivalence of $\mathcal{G}$ and $\mathcal{G}'$, 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:

> $\alpha \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \beta$ if and only if there exitsts $\hat{\alpha} \in \Pi(\alpha)$ such that $\hat{\alpha} \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \beta$.

We start with proving "only-if" part. Let $\alpha \overset{k}{\underset{\mathcal{G}}{\Rightarrow}} \beta$. If $k = 0$, then clearly $\alpha = \beta$, $\alpha \in \Pi(\alpha)$ and $\alpha \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \beta$. For $k \geq 1$, assume that the statement holds for any derivation of length less than k. If a rule of the form $A \to \gamma$ with $A \in N_0$ is used at the first step, the proof is rather simple, so we only prove the other case. Suppose that a rule $A(x) \to \gamma$ with $A \in N_1$ is used at the first step and $\alpha = \alpha'[A(\alpha'')] \underset{\mathcal{G}}{\Rightarrow} \alpha'[\gamma[\alpha'']] \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \beta$ for some $\alpha' \in T_{N \cup \Sigma}(\lceil X_1 \rceil) \cap T_{N \cup \Sigma}(\lfloor X_1 \rfloor)$ and $\alpha'' \in T_{N \cup \Sigma}$. By the induction hypothesis, there exist $\delta \in \Pi(\alpha'[\gamma[\alpha'']])$ such that $\delta \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \beta$. Here, we have to think of the three different cases: (1) $\delta \in \Pi(\alpha')$, (2) δ can be written as $\delta'[\hat{\gamma}]$ for some $\delta' \in \Pi(\alpha')$ and $\hat{\gamma} \in \Pi(\gamma)$, and (3) δ can be written as $\delta'[\hat{\gamma}[\delta'']]$ for some $\delta' \in \Pi(\alpha')$, $\hat{\gamma} \in \Pi(\gamma)$ and $\delta'' \in \Pi(\alpha'')$. See Figure 4. In the case (1), $\delta \in \Pi(\alpha)$ and $\delta \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \beta$. In the case (2), $\overline{A} \to \hat{\gamma}$ is in P' and therefore, $\delta'[\,\overline{A}\,] \in \Pi(\alpha)$ and $\delta'[\,\overline{A}\,] \underset{\mathcal{G}'}{\Rightarrow} \delta'[\hat{\gamma}] \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \beta$. And in the case (3), $A(x) \to \hat{\gamma}$ is in P' and therefore, $\delta'[A(\delta'')] \in \Pi(\alpha)$ and $\delta'[A(\delta'')] \underset{\mathcal{G}'}{\Rightarrow} \delta'[\hat{\gamma}[\delta'']] \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \beta$.

The "if" part is proved as follows. Let $\hat{\alpha} \overset{k}{\underset{\mathcal{G}'}{\Rightarrow}} \beta$ for some $\hat{\alpha} \in \Pi(\alpha)$. If $k = 0$, then clearly $\hat{\alpha} = \beta$, $\alpha = \hat{\alpha}$ and $\alpha \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \beta$. For $k \geq 1$, assume that the statement holds for any derivation of length less than k. The rule used at the first step is one of the following forms: (1) $A \to \hat{\gamma}$ with $A \in N_0$, (2) $A(x) \to \hat{\gamma}$ with $A \in N_1$, or (3) $\overline{A} \to \hat{\gamma}$ with $\overline{A} \in N_0' - N_0$. The proof of the case (1) is similar to the proofs of the other cases, so we start proving the case (2). In the case (2), $\hat{\alpha} = \hat{\alpha}'[A(\hat{\alpha}'')] \underset{\mathcal{G}'}{\Rightarrow} \hat{\alpha}'[\hat{\gamma}[\hat{\alpha}'']] \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \beta$ for some $\hat{\alpha}' \in T_{N' \cup \Sigma}(\lceil X_1 \rceil) \cap T_{N' \cup \Sigma}(\lfloor X_1 \rfloor)$ and $\hat{\alpha}'' \in T_{N' \cup \Sigma}$. By the definition of P', $A(x) \to \gamma$ is in P such that $\hat{\gamma} \in \Pi(\gamma)$. By the induction hypothesis, for any $\delta \in T_{N \cup \Sigma}$ such that $\hat{\alpha}'[\hat{\gamma}[\hat{\alpha}'']] \in \Pi(\delta)$, $\delta \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \beta$. By the definition of Π, there exists $\alpha' \in T_{N \cup \Sigma}(X_1)$ and $\alpha'' \in T_{N \cup \Sigma}$ such that $\alpha = \alpha'[A(\alpha'')] \underset{\mathcal{G}}{\Rightarrow} \alpha'[\gamma[\alpha'']]$, and $\hat{\alpha}'[\hat{\gamma}[\hat{\alpha}'']] \in \Pi(\alpha'[\gamma[\alpha'']])$. Therefore, $\alpha \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \beta$. And in the case (3), $\hat{\alpha} = \hat{\alpha}'[\,\overline{A}\,] \underset{\mathcal{G}'}{\Rightarrow} \hat{\alpha}'[\hat{\gamma}] \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \beta$ for some $\hat{\alpha}' \in T_{N' \cup \Sigma}(\lceil X_1 \rceil) \cap T_{N' \cup \Sigma}(\lfloor X_1 \rfloor)$. By the definition of P', $A(x) \to \gamma$ is in P such that $\hat{\gamma} \in \Pi(\gamma)$. By the induction hypothesis, for any $\delta \in T_{N \cup \Sigma}$ such that $\hat{\alpha}'[\hat{\gamma}] \in \Pi(\delta)$, $\delta \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \beta$. By the definition of Π, there exists $\alpha' \in T_{N \cup \Sigma}(X_1)$ and $\alpha'' \in T_{N \cup \Sigma}$ such that $\alpha = \alpha'[A(\alpha'')] \underset{\mathcal{G}}{\Rightarrow} \alpha'[\gamma[\alpha'']]$, and $\hat{\alpha}'[\hat{\gamma}] \in \Pi(\alpha'[\gamma[\alpha'']])$. Therefore, $\alpha \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \beta$.

Because $\Pi(S) = \{S\}$, $L(\mathcal{G}) = L(\mathcal{G}')$. □

Next, consideration will be given to whether the restriction of linearity can be removed from

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 monadic CFTG that generates the string language $L_{w^4} = \{wwww \mid w \in \{a,b\}^+\}$. $\mathcal{G} = (N, \Sigma, P, S)$ 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:

$$S \to A(a),\ S \to A(b),\ A(x) \to d(xxxx),$$
$$A(x) \to A(c(xa)), \text{ and } A(x) \to A(c(xb)).$$

Because $\mathcal{G}$ has the rule $A(x) \to d(xxxx)$, $\mathcal{G}$ 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 ε, 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 ε, 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 $\mathcal{G} = (N, \Sigma, P, S)$, if $\lambda \notin L_S(\mathcal{G})$, then there exists a weakly equivalent epsilon-free, linear, nondeleting, monadic CFTG $\mathcal{G}'$. If $\lambda \in L_S(\mathcal{G})$, then there exists $\mathcal{G}'$ whose epsilon-rule is only $S \to \varepsilon$.*

Proof. Since it is enough to show the existence of a weakly equivalent grammar, without loss of generality, we may assume that $\mathcal{G}$ 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 $E_0 = \{A \in N_0 | A \to \varepsilon \in P\}$ and $E_1 = \emptyset$. We repeat the following operations to E_0 and E_1 until no more operations are possible:

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

$$E_0 = \{A \in N_0 | \exists \alpha \in T_\Sigma, A \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \alpha, \text{yield}(\alpha) = \lambda\}$$

We construct $\mathcal{G}' = (N', \Sigma', P', S)$ as follows. The set of nonterminals is $N' = N'_0 \cup N'_1$ such that $N'_0 = N_0 \cup \{\overline{A} | A \in N_1\}$ 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 $A \to \varepsilon$.
- If $S \in E_0$, then $S \to \varepsilon$ is in P'.
- If $A \to B(C)$ is in P and $C \in E_0$, then $A \to \overline{B}$ is in P'.
- If $A(x) \to B(C(x))$ is in P, then $\overline{A} \to B(\overline{C})$ is in P'.
- If $A(x) \to b(C, x)$ or $A(x) \to b(x, C)$ is in P and $C \in E_0$, then $A(x) \to c(x)$ is in P'.
- If $A(x) \to b(C, x)$ or $A(x) \to b(x, C)$ is in P, then $\overline{A} \to c(C)$ is in P'.

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

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

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

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

We start with "only if" part. For 0-step derivations, (i), (ii) and (iii) clearly hold since there doesn't exists $\alpha' \in T_\Sigma$ nor $\alpha' \in T_\Sigma(X_1)$ for each statement.

We consider the cases for 1-step derivations.

[Proof of (i)] If $A \underset{\mathcal{G}'}{\Rightarrow} \alpha'$ and $\alpha' \in T_\Sigma$, then $\alpha' = a$ for some $a \in \Sigma_0$ and the rule $A \to a$ in P' has been used. Therefore, $A \to a$ is in P and $A \underset{\mathcal{G}}{\Rightarrow} a$.

[Proof of (ii)] If $A(x) \underset{\mathcal{G}'}{\Rightarrow} \alpha'$ and $\alpha' \in T_\Sigma(X_1)$, then $\alpha' = c(x)$ and the rule $A(x) \to c(x)$ in P' has been used. By the definition of P', $A(x) \to b(C, x)$ or $A(x) \to b(x, C)$ is in P for some $C \in E_0$. There exists $\gamma \in T_\Sigma$ such that $C \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \gamma$ and yield$(\gamma) = \lambda$. Therefore, $A(x) \underset{\mathcal{G}}{\Rightarrow} b(C, x) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} b(\gamma, x)$ or $A(x) \underset{\mathcal{G}}{\Rightarrow} b(x, C) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} b(x, \gamma)$, and yield$(b(\gamma, x)) =$ yield$(b(x, \gamma)) =$ yield$(c(x))$.

[Proof of (iii)] There doesn't exists $\alpha' \in T_\Sigma$ such that $\overline{A} \underset{\mathcal{G}'}{\Rightarrow} \alpha'$.

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

[Proof of (i)] If $A \underset{\mathcal{G}'}{\overset{k}{\Rightarrow}} \alpha'$, then the rule used at the first step is one of the follwoing form: (1) $A \to B(C)$ or (2) $A \to \overline{B}$. In the case (1), $A \underset{\mathcal{G}'}{\Rightarrow} B(C) \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} \beta'[\gamma'] = \alpha'$ for some $\beta' \in T_\Sigma(X_1)$ and $\gamma' \in T_\Sigma$ such that $B(x) \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} \beta'$ and $C \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} \gamma'$. By the induction hypothesis of (ii), there exists $\beta \in T_\Sigma(X_1)$ such that $B(x) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \beta$ and yield$(\beta) =$ yield(β'). By the induction hypothesis of (i), there exists $\gamma \in T_\Sigma$ such that $C \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \gamma$ and yield$(\gamma) =$ yield(γ'). By the definition of P', $A \to B(C)$ is in P. Therefore, $A \underset{\mathcal{G}}{\Rightarrow} B(C) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \beta[\gamma]$ and yield$(\beta[\gamma]) =$ yield$(\beta'[\gamma'])$. In the case (2), $A \underset{\mathcal{G}'}{\Rightarrow} \overline{B} \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} \alpha'$. By the definition of P', $A \to B(C)$ is in P for some $C \in E_0$. There exists $\gamma \in T_\Sigma$ such that $C \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \gamma$ and yield$(\gamma) = \lambda$. By the induction hypothesis of (iii), there exists $\beta \in T_\Sigma(X_1)$ such that $B(x) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \beta$ and yield$(\beta[\varepsilon]) =$ yield(α'). Therefore, $A \underset{\mathcal{G}}{\Rightarrow} B(C) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \beta[\gamma]$ and yield$(\beta[\gamma]) =$ yield(α').

[Proof of (ii)] If $A(x) \underset{\mathcal{G}'}{\overset{k}{\Rightarrow}} \alpha'$, then the rule used at the first step is one of the follwoing form: (1) $A(x) \to B(C(x))$, (2) $A(x) \to b(C, x)$ or (3) $A(x) \to b(x, C)$. Becasue these rule are in P, the proofs are direct from the induction hypothesis like the proof of the case (1) of (i).

[Proof of (iii)] If $\overline{A} \underset{\mathcal{G}'}{\overset{k}{\Rightarrow}} \alpha'$, then the rule used at the first step is one of the follwoing form: (1) $\overline{A} \to B(\overline{C})$ or (2) $\overline{A} \to c(C)$. In the case (1), $\overline{A} \underset{\mathcal{G}'}{\Rightarrow} B(\overline{C}) \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} \beta'[\gamma'] = \alpha'$ for some $\beta' \in T_\Sigma(X_1)$ and $\gamma' \in T_\Sigma$ such that $B(x) \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} \beta'$ and $\overline{C} \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} \gamma'$. By the induction hypothesis of (ii), there exists $\beta \in T_\Sigma(X_1)$ such that $B(x) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \beta$ and yield$(\beta) =$ yield(β'). By the induction hypothesis of (iii), there exists $\gamma \in T_\Sigma(X_1)$ such that $C(x) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \gamma$ and yield$(\gamma[\varepsilon]) =$ yield(γ'). By the definition of P', $A(x) \to B(C(x))$ is in P. Therefore, $A(x) \underset{\mathcal{G}}{\Rightarrow} B(C(x)) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \beta[\gamma]$ and yield$(\beta[\gamma[\varepsilon]]) =$ yield$(\beta'[\gamma'])$. In the case (2), $\overline{A} \underset{\mathcal{G}'}{\Rightarrow} c(C) \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} c(\gamma') = \alpha'$ for some $\gamma' \in T_\Sigma$ such that $C \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} \gamma'$. By the induction hypothesis of (i), there exists $\gamma \in T_\Sigma$ such that $C \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \gamma$ and yield$(\gamma) =$ yield(γ'). By the definition of P', $A(x) \to b(C, x)$ or $A(x) \to b(x, C)$ is in P. Without loss of generality, we may assume that $A(x) \to b(C, x)$ is in P. Therefore, $A(x) \underset{\mathcal{G}}{\Rightarrow} b(C, x) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} b(\gamma, x)$ and yield$(b(\gamma, x)[\varepsilon]) =$ yield$(c(\gamma'))$.

The "if" part is similarly proved as follows. For 0-step derivations, (i), (ii) and (iii) clearly hold since there doesn't exists $\alpha \in T_\Sigma$ nor $\alpha \in T_\Sigma(X_1)$ for each statement.

The cases for 1-step derivations are proved.

[Proof of (i)] If $A \underset{\mathcal{G}}{\Rightarrow} \alpha$ and $\alpha \in T_\Sigma$, then $\alpha = a$ for some $a \in \Sigma_0$ and the rule $A \to a$ in P has been used. Therefore, $A \to a$ is in P' and $A \underset{\mathcal{G}'}{\Rightarrow} a$.

[Proof of (ii) and (iii)] There doesn't exists $\alpha \in T_\Sigma$ such that $A \underset{\mathcal{G}}{\Rightarrow} \alpha$.

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

[Proof of (i)] If $A \underset{\mathcal{G}}{\overset{k}{\Rightarrow}} \alpha$, then the rule used at the first step must be of the form $A \to B(C)$. Thus, $A \underset{\mathcal{G}}{\Rightarrow} B(C) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \beta[\gamma] = \alpha$ for some $\beta \in T_\Sigma(X_1)$ and $\gamma \in T_\Sigma$ such that $B(x) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \beta$ and $C \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \gamma$. Here, we have to think of the two cases: (1) yield$(\gamma) \neq \lambda$ and (2) yield$(\gamma) = \lambda$. In the case (1), by the induction hypothesis of (ii), there exists $\beta' \in T_\Sigma(X_1)$ such that $B(x) \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} \beta'$ and yield$(\beta') =$ yield(β), and by the induction hypothesis of (i), there exists $\gamma' \in T_\Sigma$ such that $C \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} \gamma'$ and yield$(\gamma') =$ yield(γ). By the definition of P', $A \to B(C)$ is in P. Therefore, $A \underset{\mathcal{G}'}{\Rightarrow} B(C) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \beta'[\gamma']$ and yield$(\beta'[\gamma']) =$ yield$(\beta[\gamma])$. In the case (2), $C \in E_0$. Thus, $A \to \overline{B}$ is in P'. By the induction hypothesis of (iii), there exists $\beta' \in T_\Sigma(X_1)$ such that $\overline{B} \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \beta'$ and yield$(\beta') =$ yield$(\beta[\varepsilon])$. Therefore, $A \underset{\mathcal{G}'}{\Rightarrow} \overline{B} \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} \beta'$ and yield$(\beta') =$ yield$(\beta[\gamma])$.

[Proof of (ii)] If $A(x) \overset{k}{\underset{\mathcal{G}}{\Rightarrow}} \alpha$, then the rule used at the first step is one of the follwoing 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 direct from the induction hypothesis. In the case (2), $A(x) \underset{\mathcal{G}}{\Rightarrow} b(C, x) \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} b(\gamma, x) = \alpha$ for some $\gamma \in T_\Sigma$ such that $C \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \gamma$. Here, we have to think of the two cases: (a) yield$(\gamma) \neq \lambda$ and (b) yield$(\gamma) = \lambda$. (a) If yield$(\gamma) \neq \lambda$, then by the induction hypothesis of (i), there exists $\gamma' \in T_\Sigma$ such that $C \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \gamma'$ and yield(γ') = yield(γ). By the definition of P', $A(x) \rightarrow b(C, x)$ is in P'. Therefore, $A(x) \underset{\mathcal{G}'}{\Rightarrow} b(C, x) \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} b(\gamma', x)$ and yield$(b(\gamma', x))$ = yield$(b(\gamma, x))$. (b) If yield$(\gamma) = \lambda$, then $C \in E_0$ and $A(x) \rightarrow c(x)$ is in P'. Therefore, $A(x) \underset{\mathcal{G}'}{\Rightarrow} c(x)$ and yield$(c(x))$ = yield$(b(\gamma, x))$. The proof of the case (3) is similar to that of the case (2).

[Proof of (iii)] If $A(x) \overset{k}{\underset{\mathcal{G}}{\Rightarrow}} \alpha$, then the rule used at the first step is one of the follwoing 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), $A(x) \underset{\mathcal{G}}{\Rightarrow} B(C(x)) \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \beta[\gamma] = \alpha$ for some $\beta, \gamma \in T_\Sigma(X_1)$ such that $B(x) \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \beta$ and $C(x) \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \gamma$. By the definition of P', $\overline{A} \rightarrow B(\overline{C})$ is in P'. By the induction hypothesis of (ii), there exists $\beta' \in T_\Sigma(X_1)$ such that $B(x) \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \beta'$ and yield(β') = yield(β). By the induction hypothesis of (iii), there exists $\gamma' \in T_\Sigma$ such that $\overline{C} \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \gamma'$ and yield(γ') = yield$(\gamma[\varepsilon])$. Therefore, $\overline{A} \underset{\mathcal{G}'}{\Rightarrow} B(\overline{C}) \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \beta'[\gamma']$ and yield$(\beta'[\gamma'])$ = yield$(\beta[\gamma[\varepsilon]])$. In the case (2), $A(x) \underset{\mathcal{G}}{\Rightarrow} b(C, x) \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} b(\gamma, x) = \alpha$ for some $\gamma \in T_\Sigma$ such that $C \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \gamma$ and yield$(\gamma) \neq \lambda$. By the definition of P', $\overline{A} \rightarrow c(C)$ is in P'. By the induction hypothesis of (i), there exists $\gamma' \in T_\Sigma$ such that $C \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \gamma'$ and yield(γ') = yield(γ). Therefore, $\overline{A} \underset{\mathcal{G}'}{\Rightarrow} c(C) \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} c(\gamma')$ and yield$(c(\gamma'))$ = yield$(b(\gamma, x)[\varepsilon])$. The proof of the case (3) is similar to that of the case (2).

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

5 Conclusions

In this paper, the desirable features of linear monadic CFTGs have been discovered: the restriction of nondeletion doesn't affect their generative power of tree languages, and the restriction of epsilon-freeness can be assumed when their generation of string languages is considered. The key to the proofs of this paper was the simplicity of the definition 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, linguistic 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 International Workshop on Tree Adjoining Grammar and Related Formalisms: Proceedings of the Workshop*, Vancouver, pages 16–23.

Aravind K. Joshi and Yves Schabes, 1996. *Handbook of Formal Languages*, volume 3, chapter Tree-adjoining grammars, pages 69–124. Springer, Berlin.

Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. 1975. Tree adjunct grammars. *J. Computer & 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 Grammars 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. Comput.*, 25(4):862–873.

William C. Rounds. 1970. Mapping and grammars 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.