Epsilon-Free Grammars and Lexicalized Grammars that Generate the Class of the Mildly Context-Sensitive Languages

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

This study focuses on the class of string languages generated by TAGs. It examines whether the class of string languages can be generated by some epsilon-free grammars and by some lexicalized grammars. Utilizing spine grammars, this problem is solved positively. This result for spine grammars can be translated into the result for TAGs.

1 Introduction

The class of grammars called mildly context-sensitive grammars has been investigated very actively. Since it was shown that tree-adjoining grammars (TAG) (Joshi et al., 1975; Joshi and Schabes, 1996; Abeillé and Rambow, 2000), combinatory categorial grammars (CCG), linear indexed grammars (LIG), and head grammars (HG) are weakly equivalent (Vijay-Shanker and Weir, 1994), the class of string languages generated by these mildly context-sensitive grammars has been thought to be very important in the theory of formal grammars and languages. This study is strongly motivated by the work of K. Vijay-Shanker and D. J. Wier (1994) and focuses on the class of string languages generated by TAGs.

In this paper, it is examined whether the class of string languages generated by TAGs can be generated by some epsilon-free grammars and, moreover, by some lexicalized grammars. An epsilon-free grammar is a grammar with a restriction that requires no use of epsilon-rules, that is, rules defined with the empty string. Because the definitions of the four formalisms presented in the paper (Vijay-Shanker and Weir, 1994) allow the use of epsilon-rules, and all of the examples use epsilon-rules, it is natural to consider the generation problem by epsilon-free grammars. Since the notion of lexicalization is very important in the study of TAGs (Joshi and Schabes, 1996), the generation problem by lexicalized grammars is also considered.

To solve these problems, spine grammars (Fujiyoshi and Kasai, 2000) are utilized, and it is shown that for every string language generated by a TAG, not only an epsilon-free spine grammar but also a lexicalized spine grammar that genarates it can be constructed. Spine grammars are a restricted version of context-free tree grammars (CFTGs), and it was shown that spine grammars are weakly equivalent to TAGs and equivalent to linear, non-deleting, monadic CFTGs. Because considerably simple normal forms of spine grammars are known, they are useful to study the formal properties of the class of string languages generated by TAGs.

Since both TAGs and spine grammars are tree generating formalisms, they are closely related. From any epsilon-free or lexicalized spine grammar constructed in this paper, a weakly equivalent TAG is effectively obtained without utilizing epsilon-rules or breaking lexicality, the results of this paper also hold for TAGs. Though TAGs and spine grammars are weakly equivalent, the tree languages generated by TAGs are properly included in those by spine grammars. This difference occurs due to the restriction on TAGs that requires the label of the foot node to be identical to the label of the root node in an auxiliary tree. In addition, restrictions on rules of spine grammars are more lenient in some ways. Because rules of spine grammars may be non-linear, non-orderpreserving, and deleting, during derivations the copies of subtrees may be made, the order of subtrees may be changed, and subtrees may be deleted. At this point, spine grammars are different from other characterizations of TAGs (Möennich, 1997; Möennich, 1998).

2 Preliminaries

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

Let $\mathcal{N}$ be the set of all natural numbers, and let $\mathcal{N}_+$ be

TAG+7: Seventh International Workshop on Tree Adjoining Grammar and Related Formalisms.
May 20-22, 2004, Vancouver, BC, CA.
Pages 16-23.

the set of all positive integers. The concatenation operator is denoted by ' $\cdot$ '. For an alphabet Σ, the set of strings over Σ is denoted by Σ^*, and the empty string is denoted by λ.

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 $a \in \Sigma$, the rank of a is denoted by $r(a)$. For $n \geq 0$, it is defined that $\Sigma_n = \{a \in \Sigma \mid r(a) = n\}$.

A set D is a *tree domain* if D is a nonempty finite subset of $(\mathcal{N}_+)^*$ satisfying the following conditions:

- For any $d \in D$, if $d', d'' \in (\mathcal{N}_+)^*$ and $d = d' \cdot d''$, then $d' \in D$.
- For any $d \in D$ and $i, j \in \mathcal{N}_+$, if $i \leq j$ and $d \cdot j \in D$, then $d \cdot i \in D$.

Let D be a tree domain and $d \in D$. Elements in D are called *nodes*. A node d' is a *child of* d if there exists $i \in \mathcal{N}_+$ such that $d' = d \cdot i$. A node is called a *leaf* if it has no child. The node λ is called the *root*. A node that is neither a leaf nor the root is called an *internal node*. The *path from the root to* d is the set of nodes $\{d' \in D \mid d' \text{ is a prefix of } d\}$.

Let Σ be a ranked alphabet. A *tree* over Σ is a function $\alpha : D \to \Sigma$ where D is a tree domain. The set of trees over Σ is denoted by T_Σ. The domain of a tree α is denoted by D_α. For $d \in D_\alpha$, $\alpha(d)$ is called the *label* of d. The *subtree* of α *at* d is $\alpha/d = \{(d', a) \in (\mathcal{N}_+)^* \times \Sigma \mid (d \cdot d', a) \in \alpha\}$.

The expression of a tree over Σ is defined to be a string over elements of Σ, parentheses and commas. For $\alpha \in T_\Sigma$, if $\alpha(\lambda) = b$, $\max\{i \in \mathcal{N}_+ \mid i \in D_\alpha\} = n$ and for each $1 \leq i \leq n$, the expression of α/i is α_i, then the expression of α is $b(\alpha_1, \alpha_2, \ldots, \alpha_n)$. Note that n is the number of the children of the root. For $b \in \Sigma_0$, trees are written as b instead of $b()$.When the expression of α is $b(\alpha_1, \alpha_2, \ldots, \alpha_n)$, it is written that $\alpha = b(\alpha_1, \alpha_2, \ldots, \alpha_n)$, i.e., each tree is identified with its expression.

It is defined that ε is 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$(\alpha_2) \cdot \cdots \cdot$ yield(α_n).

Let Σ be a ranked alphabet, and let I be a set that is disjoint from Σ. $T_\Sigma(I)$ is defined to be $T_{\Sigma \cup I}$ where $\Sigma \cup I$ is the ranked alphabet obtained from Σ by adding all elements in I as symbols of rank 0.

Let $X = \{x_1, x_2, \ldots\}$ be the fixed countable set of variables. 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.

Let $\alpha, \beta \in T_\Sigma$ and $d \in D_\alpha$. It is defined that $\alpha\langle d \leftarrow \beta\rangle = \{(d', a) \mid (d', a) \in \alpha$ and d is not a prefix of $d'\} \cup \{(d \cdot d'', b) \mid (d'', b) \in \beta\}$, i.e., the tree $\alpha\langle d \leftarrow \beta\rangle$ is the result of replacing α/d by β.

Let $\alpha \in T_\Sigma(X_n)$, and let $\beta_1, \beta_2, \ldots, \beta_n \in T_\Sigma(X)$. The notion of substitution is defined. The result of substituting each β_i for nodes labeled by variable x_i in α, denoted by $\alpha[\beta_1, \beta_2, \ldots, \beta_n]$, is defined as follows.

- If $\alpha = a \in \Sigma_0$, then $a[\beta_1, \beta_2, \ldots, \beta_n] = a$.
- If $\alpha = x_i \in X_n$, then $x_i[\beta_1, \beta_2, \ldots, \beta_n] = \beta_i$.
- If $\alpha = b(\alpha_1, \alpha_2, \ldots, \alpha_k)$, $b \in \Sigma_k$ and $k \geq 1$, then $\alpha[\beta_1, \beta_2, \ldots, \beta_n] =$ $b(\alpha_1[\beta_1, \beta_2, \ldots, \beta_n], \ \ldots \ , \alpha_k[\beta_1, \beta_2, \ldots, \beta_n])$.

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).

Definition 2.1 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 $\alpha \in T_{N \cup \Sigma}$ and a node $d \in D_\alpha$, if $\alpha/d = A(\alpha_1, \alpha_2, \ldots, \alpha_n)$, $A \in N_n$, $\alpha_1, \alpha_2, \ldots, \alpha_n \in T_{N \cup \Sigma}$ and $A(x_1, x_2, \ldots, x_n) \to \beta$ is in P, then $\alpha \underset{\mathcal{G}}{\Rightarrow} \alpha\langle d \leftarrow \beta[\alpha_1, \alpha_2, \ldots, \alpha_n]\rangle$.

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 $L_S(\mathcal{G}) \subseteq (\Sigma_0 - \{\varepsilon\})^*$.

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 Spine Grammars

Spine grammars are CFTGs with a restriction called spinal-formed. To define this restriction, each nonterminal is additionally associated with a natural number.

Definition 2.2 A *head-pointing ranked alphabet* is a ranked alphabet in which each symbol is additionally associated with a natural number, called the *head* of a symbol, and the head of a symbol is satisfying the following conditions:

- If the rank of the symbol is 0, then the head of the symbol is also 0.
- If the rank of the symbol is greater than 0, then the head of the symbol is greater than 0 and less or equal to the rank of the symbol.

Let N be a head-pointing ranked alphabet. For $A \in N$, the head of A is denoted by $h(A)$.

Definition 2.3 Let $\mathcal{G} = (N, \Sigma, P, S)$ be a CFTG where N is a head-pointing ranked alphabet. For $n \geq 1$, a rule $A(x_1, x_2, \ldots, x_n) \rightarrow \alpha$ in P is *spinal-formed* if it satisfies the following conditions:

- There is exactly one leaf in α that is labeled by $x_{h(A)}$. The path from the root to the leaf is called the *spine* of α.
- For a node $d \in D_\alpha$, if d is on the spine and $\alpha(d) = B \in N$ with $r(B) \geq 1$, then $d \cdot h(B)$ is a node on the spine.
- Every node labeled by a variable in $X_n - \{x_{h(A)}\}$ is a child of a node on the spine.

The intuition of this restriction is given as follows. Let α be the right-hand side of a spinal-formed rule, and let d be a node on the spine of α. Suppose that $\alpha/d = B(\alpha_1, \alpha_2, \ldots, \alpha_n)$ and $B \in N_n$. Suppose also that the rule $B(x_1, x_2, \ldots, x_n) \rightarrow \beta$ is applied to d. Then, the tree $\alpha\langle d \leftarrow \beta[\alpha_1, \alpha_2, \ldots, \alpha_n]\rangle$ also satisfies the conditions of the right-hand side of a spinal-formed rule, i.e., the spines of α and β are combined into the new well-formed spine. Note that every node labeled by a variable in $X_n - \{x_{h(A)}\}$ is still a child of a node on the new spine.

A CFTG $\mathcal{G} = (N, \Sigma, P, S)$ is *spinal-formed* if every rule $A(x_1, x_2, \ldots, x_n) \rightarrow \alpha$ in P with $n \geq 1$ is spinal-formed. To shorten our terminology, it is said '*spine grammars*' instead of 'spinal-formed CFTGs'.

Example 2.4 Examples of spinal-formed and non-spinal-formed rules are shown. Let $\Sigma = \{a, b, c\}$ where the ranks of a, b, c are $0, 1, 3$, respectively. Let $N = \{A, B, C, D, E\}$ where the ranks of A, B, C, D, E are $4, 2, 5, 1, 0$, respectively, and the head of A, B, C are $3, 1, 5$, respectively. Note that $r(D) = 1$ and $r(E) = 0$ imply that $h(D) = 1$ and $h(E) = 0$. See Figure 1. The rule $A(x_1, x_2, x_3, x_4) \rightarrow \alpha_1$ is spinal-formed though x_2 occurs twice and x_4 does not occur in α_1. The rule $A(x_1, x_2, x_3, x_4) \rightarrow \alpha_2$ is not spinal-formed because the third child of the node labeled by A is not on the spine despite $h(A) = 3$. The rule $A(x_1, x_2, x_3, x_4) \rightarrow \alpha_3$ is not spinal-formed because there are two nodes labeled by $x_{h(A)}$. The rule $A(x_1, x_2, x_3, x_4) \rightarrow \alpha_4$ is not spinal-formed because the node labeled by x_4 is not a child of a node on the spine.

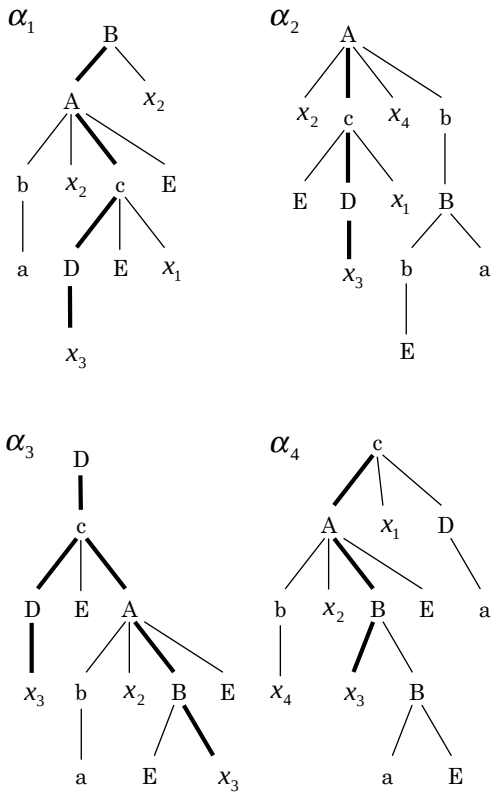

Figure 1: Right-hand sides of rules

For spine grammars, the following results are known.

Theorem 2.5 (Fujiyoshi and Kasai, 2000) *The class of string languages generated by spine grammars coincides with the class of string languages generated by TAGs.*

Theorem 2.6 (Fujiyoshi and Kasai, 2000) *For any spine grammar, there exists a equivalent spine grammar $\mathcal{G} = (N, \Sigma, P, S)$ that satisfies the following conditions:*

- *For all $A \in N$, the rank of A is either 0 or 1.*
- *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 2.*
- *For each $A \in N_1$, if $A(x) \rightarrow \alpha$ is in P, then either $\alpha = B_1(B_2(\cdots(B_m(x))\cdots))$ with $m \geq 0$ and*

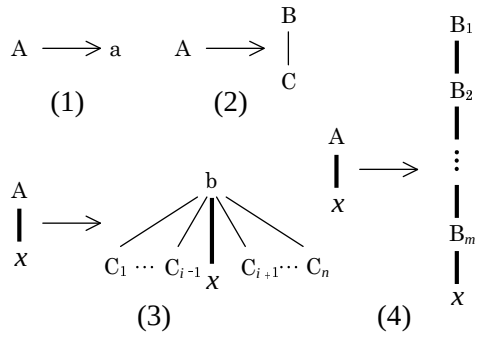

Figure 2: Rules in normal form

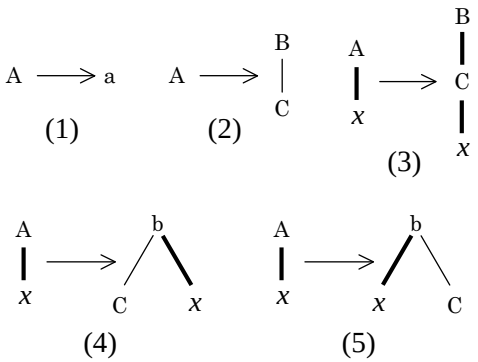

Figure 3: Rules in strong normal form

$B_1, B_2, \ldots, B_m \in N_1$ *or* $\alpha = b(C_1, C_2, \ldots, C_n)$ *with* $n \geq 1$, $b \in \Sigma_n$, *and* $C_1, C_2, \ldots, C_n$ *such that all are in* N_0 *but* $C_i = x$ *for exactly one* $i \in \{1, \ldots, n\}$. *See (3) and (4) in Figure 2.*

A spine grammar satisfies the condition of Theorem 2.6 is said to be in *normal form*. Note that for a spine grammar in normal form, the heads assigned for each nonterminal are not essential anymore because $h(A) = r(A)$ for all $A \in N$.

Theorem 2.7 (Fujiyoshi and Kasai, 2000) *For any spine grammar, there exists a weakly equivalent spine grammar* $\mathcal{G} = (N, \Sigma, P, S)$ *that satisfies the following conditions:*

- *For all* $A \in N$, *the rank of* A *is either 0 or 1.*
- *For all* $a \in \Sigma$, *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.*
- *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.

A spine grammar satisfies the condition of Theorem 2.7 is said to be in *strong normal form*.

3 The Construction of Epsilon-Free Spine Grammars

According to our definition of spine grammars, they are allowed to generate trees with leaves labeled by the special symbol ε, which is treated as the empty string while taking the yields of trees. In this section, it is shown that for any spine grammar, there exists a weakly equivalent epsilon-free spine grammar. Because any epsilon-free spine grammar cannot generate a tree with its leaves labeled by ε, it is clear that for a spine grammar with epsilon-rules, there generally doesn't exist an equivalent epsilon-free spine grammar.

Definition 3.1 A spine grammar $\mathcal{G} = (N, \Sigma, P, S)$ is *epsilon-free* if for any rule $A(x_1, x_2, \ldots, x_n) \to \alpha$ in P, α has no node labeled by the symbol ε.

Theorem 3.2 *For any spine grammar* $\mathcal{G} = (N, \Sigma, P, S)$, *if* $\lambda \notin L_S(\mathcal{G})$, *then we can construct a weakly equivalent epsilon-free spine grammar* $\mathcal{G}'$. *If* $\lambda \in L_S(\mathcal{G})$, *then we can construct a weakly equivalent spine grammar* $\mathcal{G}'$ *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 yield$(\alpha) =$ 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 yield$(\alpha) =$ yield(α').

(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 yield$(\alpha[\varepsilon]) =$ 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 $C \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \gamma$ and yield$(\gamma) = \lambda$. Therefore, $A(x) \underset{\mathcal{G}}{\Rightarrow} b(C, x) \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} b(\gamma, x)$ or $A(x) \underset{\mathcal{G}}{\Rightarrow} b(x, C) \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} b(x, \gamma)$, 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 $A \overset{k}{\underset{\mathcal{G}'}{\Rightarrow}} \alpha'$, 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), $A \underset{\mathcal{G}'}{\Rightarrow} B(C) \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \beta'[\gamma'] = \alpha'$ for some $\beta' \in T_\Sigma(X_1)$ and $\gamma' \in T_\Sigma$ such that $B(x) \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \beta'$ and $C \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \gamma'$. 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$(\beta) =$ yield(β'). By the induction hypothesis of (i), there exists $\gamma \in T_\Sigma$ such that $C \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \gamma$, and yield$(\gamma) =$ yield(γ'). By the definition of P', $A \to B(C)$ is in P. Therefore, $A \underset{\mathcal{G}}{\Rightarrow} B(C) \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \beta[\gamma]$, and yield$(\beta[\gamma]) =$ yield$(\beta'[\gamma'])$. In the case (2), $A \underset{\mathcal{G}'}{\Rightarrow} \overline{B} \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \alpha'$. 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 $C \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \gamma$ and yield$(\gamma) = \lambda$. By the induction hypothesis of (iii), there exists $\beta \in T_\Sigma(X_1)$ such that $B(x) \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \beta$ and yield$(\beta[\varepsilon]) =$ yield(α'). Therefore, $A \underset{\mathcal{G}}{\Rightarrow} B(C) \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \beta[\gamma]$, and yield$(\beta[\gamma]) =$ yield(α').

[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) $A(x) \to B(C(x))$, (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 $\overline{A} \overset{k}{\underset{\mathcal{G}'}{\Rightarrow}} \alpha'$, 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), $\overline{A} \underset{\mathcal{G}'}{\Rightarrow} B(\overline{C}) \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \beta'[\gamma'] = \alpha'$ for some $\beta' \in T_\Sigma(X_1)$ and $\gamma' \in T_\Sigma$ such that $B(x) \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \beta'$ and $\overline{C} \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \gamma'$. 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$(\beta) =$ yield(β'). By the induction hypothesis of (iii), there exists $\gamma \in T_\Sigma(X_1)$ such that $C(x) \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \gamma$ and yield$(\gamma[\varepsilon]) =$ yield(γ'). By the definition of P', $A(x) \to B(C(x))$ is in P. Therefore, $A(x) \underset{\mathcal{G}}{\Rightarrow} B(C(x)) \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \beta[\gamma]$, and yield$(\beta[\gamma[\varepsilon]]) =$ yield$(\beta'[\gamma'])$. In the case (2), $\overline{A} \underset{\mathcal{G}'}{\Rightarrow} c(C) \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} c(\gamma') = \alpha'$ for some $\gamma' \in T_\Sigma$ such that $C \overset{*}{\underset{\mathcal{G}'}{\Rightarrow}} \gamma'$. By the induction hypothesis of (i), there exists $\gamma \in T_\Sigma$ such that $C \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} \gamma$ 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, $A(x) \underset{\mathcal{G}}{\Rightarrow} b(C, x) \overset{*}{\underset{\mathcal{G}}{\Rightarrow}} b(\gamma, x)$, 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 $A(x) \underset{\mathcal{G}}{\overset{k}{\Rightarrow}} \alpha$, 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) $A(x) \to b(x, C)$. The proof of the case (1) is direct from the induction hypothesis. In the case (2), $A(x) \underset{\mathcal{G}}{\Rightarrow} b(C, x) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} b(\gamma, x) = \alpha$ for some $\gamma \in T_\Sigma$ such that $C \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \gamma$. 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 $C \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} \gamma'$, and yield$(\gamma') =$ yield(γ). By the definition of P', $A(x) \to b(C, x)$ is in P'. Therefore, $A(x) \underset{\mathcal{G}'}{\Rightarrow} b(C, x) \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} b(\gamma', x)$, and yield$(b(\gamma', x)) =$ yield$(b(\gamma, x))$. (b) If yield$(\gamma) = \lambda$, then $C \in E_0$, and $A(x) \to c(x)$ 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 $A(x) \underset{\mathcal{G}}{\overset{k}{\Rightarrow}} \alpha$, 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) $A(x) \to b(x, C)$. In the case (1), $A(x) \underset{\mathcal{G}}{\Rightarrow} B(C(x)) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \beta[\gamma] = \alpha$ for some $\beta, \gamma \in T_\Sigma(X_1)$ such that $B(x) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \beta$ and $C(x) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \gamma$. By the definition of P', $\overline{A} \to B(\overline{C})$ is in P'. 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 $\gamma' \in T_\Sigma$ such that $\overline{C} \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} \gamma'$ and yield$(\gamma') =$ yield$(\gamma[\varepsilon])$. Therefore, $\overline{A} \underset{\mathcal{G}'}{\Rightarrow} B(\overline{C}) \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} \beta'[\gamma']$ and yield$(\beta'[\gamma']) =$ yield$(\beta[\gamma[\varepsilon]])$. In the case (2), $A(x) \underset{\mathcal{G}}{\Rightarrow} b(C, x) \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} b(\gamma, x) = \alpha$ for some $\gamma \in T_\Sigma$ such that $C \underset{\mathcal{G}}{\overset{*}{\Rightarrow}} \gamma$ and yield$(\gamma) \neq \lambda$. By the definition of P', $\overline{A} \to c(C)$ is in P'. 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). Therefore, $\overline{A} \underset{\mathcal{G}'}{\Rightarrow} c(C) \underset{\mathcal{G}'}{\overset{*}{\Rightarrow}} c(\gamma')$, 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 $L_S(\mathcal{G}') = L_S(\mathcal{G})$. □

4 Lexicalization of Spine Grammars

In this section, lexicalization of spine grammars is discussed. First, it is seen that there exists a tree language generated by a spine grammar that no lexicalized spine grammar can generate. Next, it is shown that for any spine grammar, there exists a weakly equivalent lexicalized spine grammar. In the construction of a lexicalized spine grammar, the famous technique to construct a context-free grammar (CFG) in Greibach normal form (Hopcroft and Ullman, 1979) is employed. The technique can be adapted to spine grammars because paths of derivation trees of spine grammars can be similarly treated as derivation strings of CFGs.

Definition 4.1 A spine grammar $\mathcal{G} = (N, \Sigma, P, S)$ is *lexicalized* if it is epsilon-free and for any rule $A(x_1, x_2, \ldots, x_n) \to \alpha$ in P, α has exactly one leaf labeled by a terminal and the other leaves are labeled by a nonterminal or a variable.

The following example is a spine grammar that no lexicalized spine grammar is equivalent.

Example 4.2 Let us consider the spine grammar $\mathcal{G} = (N, \Sigma, P, S)$ such that $\Sigma = \{a, b\}$ with $r(a) = 0$ and $r(b) = 1$, $N = \{S\}$ with $r(S) = 0$, and P consists of $S \to a$ and $S \to b(S)$. The tree language generated by $\mathcal{G}$ is $L(\mathcal{G}) = \{a, b(a), b(b(a)), b(b(b(a))), \ldots\}$. Suppose that $L(\mathcal{G})$ is generated by a lexicalized spine grammar $\mathcal{G}'$. Because $L_S(\mathcal{G}) = \{a\}$, all trees in $L(\mathcal{G})$ have to be derived in one step, and the set of rules of $\mathcal{G}'$ has to be $\{S \to a, S \to b(a), S \to b(b(a)), S \to b(b(b(a))), \ldots\}$. However, the number of rules of $\mathcal{G}'$ has to be finite. Therefore, $L(\mathcal{G})$ can not be generated by any lexicalized spine grammar.

To prove the main theorem, the following lemmas are needed.

Lemma 4.3 *For any spine grammar $\mathcal{G}$, we can construct a weakly equivalent spine grammar in normal form $\mathcal{G}' = (N, \Sigma, P, S)$ that doesn't have a rule of the form $A(x) \to b(x)$ with $A \in N_1$ and $b \in \Sigma_1$.*

Proof. Without loss of generality, we may assume that $\mathcal{G}$ is in normal form. For each rule of the form $A(x) \to b(x)$, delete it and add the rule $A(x) \to x$. Then, a weakly equivalent grammar is constructed. □

Lemma 4.4 *For any spine grammar $\mathcal{G}$, we can construct an equivalent spine grammar in normal form $\mathcal{G}' =$*

(N, Σ, P, S) *that doesn't have a rule of the form* $A(x) \to x$ *with* $A \in N_1$.

Proof. Omitted. □

The following lemmas guarantees that the technique to construct a CFG in Greibach normal form (Hopcroft and Ullman, 1979) can be adapted to spine grammars.

Lemma 4.5 *Define an A-rule to be a rule with a nonterminal A on the left-hand side. Let* $\mathcal{G} = (N, \Sigma, P, S)$ *be a spine grammar such that* $r(A) \leq 1$ *for all* $A \in N$.

For $A \in N_0$, *let* $A \to \alpha$ *be a rule in* P *such that* $\alpha(d) = B$ *for some* $d \in D_\alpha$ *and* $B \in N_0$. *Let* $\{B \to \beta_1, B \to \beta_2, \ldots, B \to \beta_r\}$ *be the set of all B-rules. Let* $\mathcal{G}' = (N, \Sigma, P', S)$ *be obtaind from* $\mathcal{G}$ *by deleting the rule* $A \to \alpha$ *from* P *and adding the rules* $A \to \alpha\langle d \leftarrow \beta_i\rangle$ *for all* $1 \leq i \leq r$. *Then* $L(\mathcal{G}') = L(\mathcal{G})$.

For $A \in N_1$, *let* $A(x) \to \alpha$ *be a rule in* P *such that* $\alpha(d) = B$ *for some* $d \in D_\alpha$ *and* $B \in N_1$. *Let* $\{B(x) \to \beta_1, B(x) \to \beta_2, \ldots, B(x) \to \beta_r\}$ *be the set of all B-rules. Let* $\mathcal{G}'' = (N, \Sigma, P'', S)$ *be obtaind from* $\mathcal{G}$ *by deleting the rule* $A(x) \to \alpha$ *from* P *and adding the rules* $A(x) \to \alpha\langle d \leftarrow \beta_i[\alpha/d \cdot 1]\rangle$ *for all* $1 \leq i \leq r$. *Then* $L(\mathcal{G}'') = L(\mathcal{G})$.

Proof. Omitted. □

Lemma 4.6 *Let* $\mathcal{G} = (N, \Sigma, P, S)$ *be spine grammar such that* $r(A) \leq 1$ *for all* $A \in N$.

For $A \in N_0$, *let* $A \to \alpha_1, A \to \alpha_2, \ldots, A \to \alpha_r$ *be the set of A-rules such that for all* $1 \leq i \leq r$, *there exists a leaf node* $d_i \in D\alpha_i$ *labeled by A. Let* $A \to \beta_1, A \to \beta_2, \ldots, A \to \beta_s$ *be the remaining A-rules. Let* $\mathcal{G}' = (N \cup \{Z\}, \Sigma, P', S)$ *be the spine grammar formed by adding a new nonterminal* Z *to* N_1 *and replacing all the A-rules by the rules:*

$$1) \quad \left.\begin{array}{l} A \to \beta_i \\ A \to Z(\beta_i) \end{array}\right\} 1 \leq i \leq s$$

$$2) \quad \left.\begin{array}{l} Z(x) \to \alpha_i\langle d_i \leftarrow x\rangle \\ Z(x) \to Z(\alpha_i\langle d_i \leftarrow x\rangle) \end{array}\right\} 1 \leq i \leq r$$

Then $L(\mathcal{G}') = L(\mathcal{G})$.

For $A \in N_1$, *let* $A(x) \to A(\alpha_1), A(x) \to A(\alpha_2), \ldots, A(x) \to A(\alpha_r)$ *be the set of A-rules such that A is the label of the root node of the right-hand side. Let* $A(x) \to \beta_1$, $A(x) \to \beta_2, \ldots, A(x) \to \beta_s$ *be the remaining A-rules. Let* $\mathcal{G}'' = (N \cup \{Z\}, \Sigma, P'', S)$ *be the spine grammar formed by adding a new nonterminal* Z *to* N_1 *and replacing all the A-rules by the rules:*

$$1) \quad \left.\begin{array}{l} A(x) \to \beta_i \\ A(x) \to \beta_i[Z(x)] \end{array}\right\} 1 \leq i \leq s$$

$$2) \quad \left.\begin{array}{l} Z(x) \to \alpha_i \\ Z(x) \to \alpha_i[Z(x)] \end{array}\right\} 1 \leq i \leq r$$

Then $L(\mathcal{G}'') = L(\mathcal{G})$.

Proof. Omitted. □

Theorem 4.7 *For any spine grammar* $\mathcal{G} = (N, \Sigma, P, S)$, *we can construct a weakly equivalent lexicalized spine grammar* $\mathcal{G}'$.

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 epsilon-free and $\mathcal{G}$ is in normal form without a rule of the form $A(x) \to b(x)$ with $A \in N_1$ and $b \in \Sigma_1$. We may also assume that $\mathcal{G}$ is in normal form without a rule of the form $A(x) \to x$ with $A \in N_1$.

Each rule in P is one of the following form:

Type 1 $A \to a$ with $A \in N_0$ and $a \in \Sigma_0$,

Type 2 $A \to B(C)$ with $A \in N_0$, $B \in N_1$, and $C \in N_0$,

Type 3 $A(x) \to b(C_1, C_2, \ldots, C_n)$ with $A \in N_1$, $n \geq 2$ $b \in \Sigma_n$, and $C_1, C_2, \ldots, C_n$ such that all are in N_0 but $C_i = x$ for exactly one $i \in \{1, \ldots, n\}$, or

Type 4 $A(x) \to B_1(B_2(\cdots(B_m(x))\cdots))$ with $A \in N_1$, $m \geq 1$, and $B_1, B_2, \ldots, B_m \in N_1$.

Because of the assumption above, $m \geq 1$ and $n \geq 2$.

First, by the technique to construct a CFG in Greibach normal form, we replace all type 1 and type 2 rules with rules of the form $A \to B_1(B_2(\cdots(B_m(a))\cdots))$ with $m \geq 0$, $a \in \Sigma_0$, and $B_1, \ldots, B_m \in N_1$ and new type 4 rules with a new nonterminal on the left-hand side. See (1) in Figure 4.

Secondly, we consider type 4 rules of the form $A(x) \to B(x)$. By the standard technique of formal language theory, those rules can be replaced by other type 4 rules with at least 2 nonterminals on the right-hand side.

Thirdly, by the technique to construct a CFG in Greibach normal form, we replace all type 1 and type 2 rules with $A(x) \to b(\gamma_1, \gamma_2, \ldots, \gamma_n)$ with $\gamma_1, \gamma_2, \ldots, \gamma_n$ such that all are in N_0 but $\gamma_i \in T_{N_1}(X_1)$ for exactly one $i \in \{1, \ldots, n\}$. See (2) in Figure 4.

Lastly, the remaining non-lexicalized rules are only of the form $A(x) \to b(\gamma_1, \gamma_2, \ldots, \gamma_n)$. Because $n \geq 2$, the right-hand side has a node labeled by a nonterminal in N_0. This node can be replaced by the right-hand side of the rules of the form $C_i \to D_1(D_2(\cdots(D_m(a))\cdots))$. See (3) in Figure 4.

A weakly equivalent lexicalized spine grammar $\mathcal{G}'$ is constructed. □

If $\mathcal{G}$ is epsilon-free and dosen't have a rule of the form $A(x) \to b(x)$ with $A \in N_1$ and $b \in \Sigma_1$, then the equivalence of the construted grammar is preserved. The following corollary is immediate.

Corollary 4.8 *For any epsilon-free spine grammar* $\mathcal{G} = (N, \Sigma, P, S)$ *such that* $\Sigma_1 = \emptyset$, *we can construct an equivalent lexicalized spine grammar* $\mathcal{G}'$.

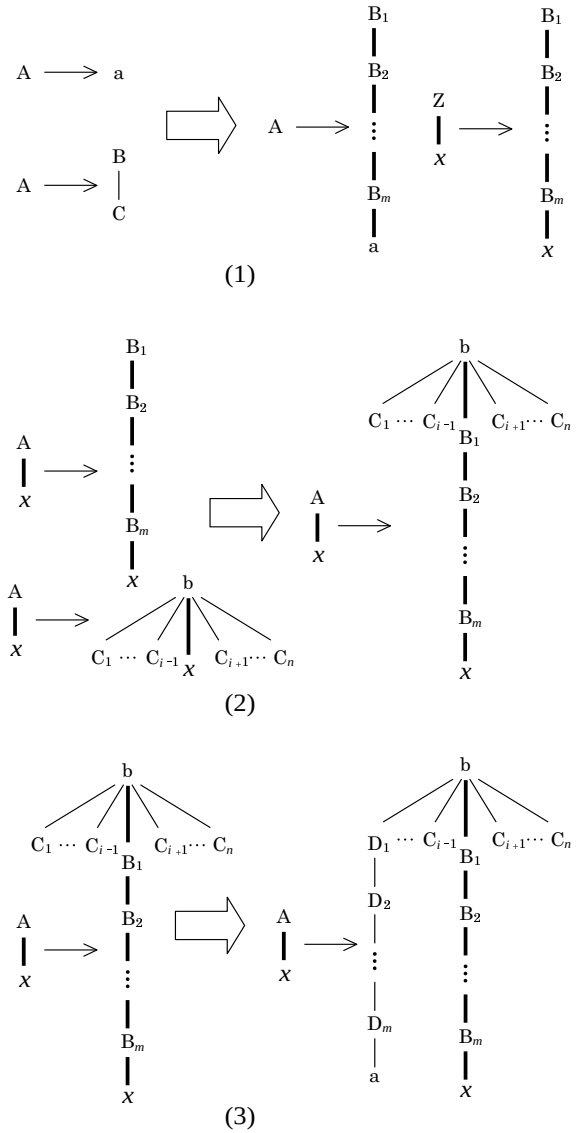

Figure 4: The explanation for the proof

5 The Results for TAGs

From a spine grammar, a weakly equivalent TAG is obtained easily. Recall the definition of TAGs in the paper (Joshi and Schabes, 1996). Let $\mathcal{G} = (N, \Sigma, P, S)$ be a spine grammar in strong normal form. A weakly equivalent TAG $\mathcal{G}' = (\Sigma_0, N \cup (\Sigma - \Sigma_0), I, A, S)$ can be constructed as follows. The set of initial trees is the smallest set satisfying following conditions:

- The tree $S\downarrow$ is in I.
- If $A \rightarrow a$ with $A \in N_0$ and $a \in \Sigma_0$ is in P, then $A_{NA}(a)$ is in I.
- If $A \rightarrow B(C)$ with $A \in N_0$, $B \in N_1$, and $C \in N_0$ is in P, then $A_{NA}(B_{OA}(C\downarrow))$ is in I.

The set of auxiliary trees is the smallest set satisfying following conditions:

- If $A(x) \rightarrow B(C(x))$ with $A \in N_1$ and $B, C \in N_1$ is in P, then $A_{NA}(B_{OA}(C_{OA}(A_{NA}*)))$ is in A.
- If $A(x) \rightarrow b(C, x)$ with $A \in N_1$, $b \in \Sigma_2$ and $C \in N_0$ is in P, then $A_{NA}(b_{NA}(C\downarrow, A_{NA}*))$ is in A.
- If $A(x) \rightarrow b(x, C)$ with $A \in N_1$, $b \in \Sigma_2$ and $C \in N_0$ is in P, then $A_{NA}(b_{NA}(A_{NA}*, C\downarrow))$ is in A.

The way to construct a weakly equivalent TAG from a spine grammar in strong normal form was shown. By a similar way, a weakly equivalent TAG is effectively obtained from any epsilon-free or lexicalized spine grammar constructed in this paper without utilizing epsilon-rules or breaking lexicality. Therefore, the results for spine grammars also hold for TAGs.

Corollary 5.1 *For any TAG, we can construct a weakly equivalent epsilon-free TAG.*

Corollary 5.2 *For any TAG, we can construct a weakly equivalent lexicalized TAG.*

References

Anne Abeillé and Owen Rambow, editors. 2000. *Tree adjoining grammars: formalisms, linguistic analysis and processing*. CSLI Publications, Stanford, California.

Akio Fujiyoshi and Takumi Kasai. 2000. Spinal-formed context-free tree grammars. *Theory of Computing Systems*, 33(1):59–83.

John E. Hopcroft and Jeffrey D. Ullman. 1979. *Introduction to Automata Theory, Languages and Computation*. Addison Wesley, Reading, Massachusetts.

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.

Uwe Möennich. 1998. TAGs M-constructed. In *TAG+ 4th Workshop*, Philadelphia.

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.