JoLLI manuscript No.
(will be inserted by the editor)

The Equivalence of Tree Adjoining Grammars and Monadic Linear Context-free Tree Grammars

Stephan Kepser · **Jim Rogers**

the date of receipt and acceptance should be inserted later

Abstract The equivalence of leaf languages of tree adjoining grammars and monadic linear context-free grammars was shown about a decade ago. This paper presents a proof of the strong equivalence of these grammar formalisms. Non-strict tree adjoining grammars and monadic linear context-free grammars define the same class of tree languages. We also present a logical characterisation of this tree language class showing that a tree language is a member of this class iff it is the two-dimensional yield of an MSO-definable three-dimensional tree language.

Keywords Tree adjoining grammar, monadic linear context-free tree grammar, monadic second-order logic, model theoretic syntax

1 Introduction

Tree Adjoining Grammars (Joshi et al 1975; Joshi and Schabes 1997) (TAGs) are a grammar formalism introduced by Joshi to extend the expressive power of context-free string grammars (alias local tree grammars) in a small and controlled way to render certain known mildly context-sensitive phenomena in natural language. The basic operation in these grammars, the adjunction operation, consists in replacing a node in a tree by a complete tree drawn from a finite collection.

Context-free Tree Grammars (CFTGs, see (Gécseg and Steinby 1997) for an overview) have been studied in informatics since the late 1960ies. They provide a very powerful mechanism of defining tree languages. Rules of a CFTG define how to replace non-terminal nodes by complete trees.

Stephan Kepser
Collaborative Research Center 441
University of Tübingen
Tübingen, Germany
E-mail: s.kepser@gmx.net

Jim Rogers
Computer Science Department
Earlham College
Richmond, IN, USA
E-mail: rogers@cs.earlham.edu

It has been observed quite early after the introduction of TAGs that the adjoining operation seems to be a special case of the more general deduction step in a CFTG-derivation. TAGs look like special cases of subclasses of CFTGs. This intuition was strengthened by showing that the yield languages definable by TAGs are equivalent to the yield languages definable by monadic linear non-deleting CFTGs, as was shown independently by Mönnich (1997) and Fujiyoshi and Kasai (2000). The question of the strong equivalence of the two formalisms remained unanswered.

Rogers (1998, 2003) introduced a variant of TAGs called non-strict TAGs. Non-strict TAGs generalise the definition of TAGs by releasing the conditions that the root node an foot node of an elementary tree must bear equal labels and that the label of the node to be replaced must be equal to the root node of the adjoined tree. The first proposal of such an extension of TAGs was made by Lang (1994). The new variant of TAGs looks even more like a subclass of CFTGs. And indeed, non-strict TAGs and monadic linear CFTGs are strongly equivalent. This is the main result of the present paper.

We would like to point out that there is a small technical issue connected with this result. Call a tree *ranked* iff for every node the number of its children is a function of its label. It is well known that CFTGs define ranked trees. TAGs on the other hand define unranked trees. A tree generated by a TAG may have a leaf node and an internal node labelled with the same label. Taking the definition of ranked trees strictly, this is not possible with CFTG generated trees. Our view on this issue is a practical one: a function is not just defined by its name, rather by its name and arity. Hence a three-place A can be distinguished from a constant A by their difference in arity, though the function – or label – name is the same. For every label, a TAG only introduces a finite set of arities. Hence we opt for extending the definition of a ranked alphabet to be a function form labels to finite sets of natural numbers.

The equivalence result and a previous result by Rogers (2003) provide a new characterisation of the class of tree languages definable by monadic linear CFTGs by means of logics. A tree language is definable by a monadic linear CFTG if and only if it is the two-dimensional yield of an MSO-definable three-dimensional tree language.

The paper is organised as follows. The next section introduces trees, context-free tree grammars, tree-adjoining grammars, and three-dimensional trees. Section 3 introduces a special type of CFTGs called footed CFTGs. These grammars can be seen as the CFTG-counterpart of TAGs. Section 3 contains the equivalence of monadic linear CFTGs and footed CFTGs. Section 4 shows that footed CFTGs are indeed the CFTG-counterpart of TAGs providing the equivalence of both grammar types. The fifth section states the aforementioned logical characterisation of the expressive power of monadic linear CFTGs.

2 Preliminaries

2.1 Trees

We consider labelled finite ordered ranked trees. A tree is ordered iff there is a linear order on the daughters of each node. A tree is ranked iff the label of a node implies the number of daughter nodes.

A tree domain is a finite subset of the set of strings over natural numbers that is closed under prefixes and left sisters. Formally, let $\mathbb{N}^*$ denote the set of all finite sequences of natural numbers including the empty sequence and $\mathbb{N}^+$ be the of all finite sequences of natural numbers excluding the empty sequence. A set $D \subset_{\text{fin}} \mathbb{N}^*$ is called a *tree domain* iff for all $u, v \in \mathbb{N}^* : uv \in D \Rightarrow u \in D$ (prefix closure) and for all $u \in \mathbb{N}^*, i \in \mathbb{N} : ui \in D \Rightarrow \forall j <$

$i : uj \in D$ (closure under left sisters). An element of a tree domain is an address of a node in the tree. It is called a *position*.

Let Σ be a set of labels. A *tree* is a pair (D, λ) where D is a tree domain and $\lambda : D \to \Sigma$ is a tree labelling function. The set of all trees labelled with symbols from Σ is denoted $\mathscr{T}_\Sigma$. A tree language $L \subseteq \mathscr{T}_\Sigma$ is just a subset of $\mathscr{T}_\Sigma$.

A set Σ of labels is *ranked* iff there is a function $\rho : \Sigma \to \wp_{\text{fin}}(\mathbb{N})$ assigning each symbol a finite set of arities. If $t = (D, \lambda)$ is a tree of a ranked alphabet Σ then for each position $p \in D : p(n-1) \in D, pm \notin D$ for every $m \geq n \Rightarrow n \in \rho(\lambda(p))$. For a natural number n the set Σ^n denotes the set of all labels of arity n, the set Σ^0 is the set of constants.

If X is a set (of symbols) disjoint from Σ, then $\mathscr{T}_\Sigma(X)$ denotes the set of trees $\mathscr{T}_{\Sigma \cup X}$ where all elements of X are taken as constants. The elements of X are understood to be "variables".

Let $X = \{x_1, x_2, x_3, \dots\}$ be a fixed denumerable set of *variables*. Let $X_0 = \emptyset$ and, for $k \geq 1$, $X_k = \{x_1, \dots, x_k\} \subset X$. For $k \geq 0, m \geq 0, t \in \mathscr{T}_\Sigma(X_k)$, and $t_1, \dots, t_k \in \mathscr{T}_\Sigma(X_m)$, we denote by $t[t_1, \dots, t_k]$ the result of *substituting* t_i for all occurrences of x_i in t. Note that $t[t_1, \dots, t_k]$ is in $\mathscr{T}_\Sigma(X_m)$. Note also that for $k = 0$, $t[t_1, \dots, t_k] = t$.

2.2 Context-Free Tree Grammars

We start with the definition of a context-free tree grammar quoting (Engelfriet and Schmidt 1977).

Definition 1 A *context-free tree grammar* is a quadruple $G = (\Sigma, \mathscr{F}, S, P)$ where

Σ	is a finite ranked alphabet of *terminals*,
$\mathscr{F}$	is a finite ranked alphabet of *nonterminals* or *function symbols*, disjoint with Σ,
$S \in \mathscr{F}^0$	is the *start symbol*, and
P	is a finite set of productions (or rules) of the form $F(x_1, \dots, x_k) \to \tau$, where $F \in \mathscr{F}^k$ and $\tau \in \mathscr{T}_{\Sigma \cup \mathscr{F}}(X_k)$.

We use the convention that for $k = 0$ an expression of the form $F(\tau_1, \dots, \tau_k)$ stands for F. In particular, for $F \in \mathscr{F}^0$, a rule is of the form $F \to \tau$ with $\tau \in \mathscr{T}_{\Sigma \cup \mathscr{F}}$. We sometimes use little superscripts to indicate the arity of a non-terminal, like in F^3.
The terminus *context-free tree grammar* is abbreviated by CFTG.

For a context-free tree grammar $G = (\Sigma, \mathscr{F}, S, P)$ we now define the direct derivation relation. Let $n \geq 0$ and let $\sigma_1, \sigma_2 \in \mathscr{T}_{\Sigma \cup \mathscr{F}}(X_n)$. We define $\sigma_1 \underset{G}{\Rightarrow} \sigma_2$ if and only if there is a production $F(x_1, \dots, x_k) \to \tau$, a tree $\eta \in \mathscr{T}_{\Sigma \cup \mathscr{F}}(X_{n+1})$ containing *exactly one* occurrence of x_{n+1}, and trees $\xi_1, \dots, \xi_k \in \mathscr{T}_{\Sigma \cup \mathscr{F}}(X_n)$ such that

$$\sigma_1 = \eta[x_1, \dots, x_n, F(\xi_1, \dots, \xi_k)]$$

and

$$\sigma_2 = \eta[x_1, \dots, x_n, \tau[\xi_1, \dots, \xi_k]].$$

In other words, σ_2 is obtained from σ_1 by replacing an occurrence of a subtree $F(\xi_1, \dots, \xi_k)$ by the tree $\tau[\xi_1, \dots, \xi_k]$.

As usual, $\overset{*}{\underset{G}{\Rightarrow}}$ stands for the reflexive-transitive closure of $\underset{G}{\Rightarrow}$. For a context-free tree grammar G, we define $L(G) = \{t \in T_\Sigma \mid S \overset{*}{\underset{G}{\Rightarrow}} t\}$. $L(G)$ is called the *tree language* generated

by G. Two grammars G and G' are *equivalent*, if they generate the same tree language, i.e., $L(G) = L(G')$.

We define three subtypes of context-free tree grammars. A production $F(x_1,\ldots,x_k) \to \tau$ is called *linear*, iff each variable $x_1,\ldots,x_k$ occurs at most once in τ. Linear productions do not allow the copying of subtrees. A tree grammar $G = (\Sigma, \mathscr{F}, S, P)$ is called a *linear* context-free tree grammar, if every rule in P is linear. All the CFTGs we consider in this paper are linear.

Secondly, a rule $F(x_1,\ldots,x_k) \to \tau$ is *non-deleting* iff each variable $x_1,\ldots,x_k$ occurs in τ. A CFTG is non-deleting if each rule is non-deleting.

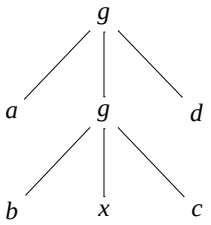
Thirdly, a CFTG $G = (\Sigma, \mathscr{F}, S, P)$ is *monadic* iff $\mathscr{F}^k = \emptyset$ for every $k > 1$. Non-terminals can only be constants or of rank 1. Monadic linear context-free tree grammars are abbreviated MLCFTGs.

We note that there are different types of derivation modes defined for CFTGs in general. These are (beyond the general one above) the inside-out and outside-in derivation modes. In inside-out derivation mode, a non-terminal node can only be expanded if the subtree below it does not contain any other non-terminal. In outside-in derivation mode, a non-terminal node can only be expanded if the path from the root to this nodes does not contain another non-terminal node. In general, these different derivation modes generate different tree languages. But for *linear* non-deleting CFTGs, all three different derivation modes generate the same tree language for a given grammar, as was shown by Kepser and Mönnich (2006). Since we only consider linear CFTGs in this paper, we just defined the general derivation mode.

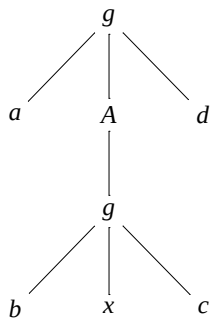
Example 1 Let $G_1 = (\{g^2, a, b, c, d, e\}, \{S, A^1\}, S, P)$ where P consists of the three rules

$S \to A(e)$

$A(x) \to$



$A(x) \to$



G_1 is monadic, linear, and non-deleting. The tree language generated by G_1 is depicted in Figure 1. As one can see, it is *not* a regular tree language. Its yield language is the non-context-free language $\{a^n b^n e c^n d^n \mid n \geq 1\}$.

2.3 Tree Adjoining Grammars

We consider so-called non-strict Tree Adjoining Grammars. Non-strict TAGs were introduced by Rogers (2003) as an extension of TAGs that reflects the fact that adjunction (or substitution) operations are fully controlled by obligatory and selective adjoining constraints. There is hence no need to additionally demand the equality of head and foot node labels or the equality of the labels of the replaced node with the head node of the adjoined tree.

Following Rogers (2003), a non-strict TAG is a pair (E, I) where E is a finite set of elementary trees in which each node is associated with

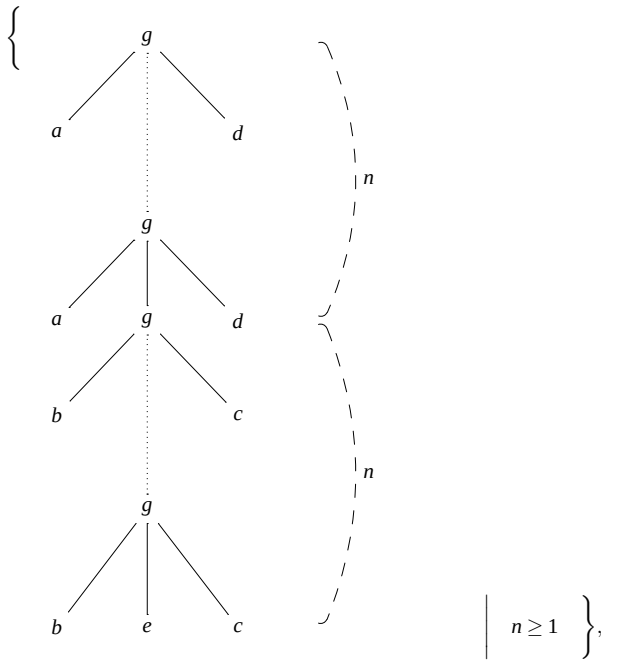


Fig. 1 The tree language generated by G_1.

- a label – drawn from some alphabet,
- a *selective adjunction* (SA) constraint – a subset of the set of names of the elementary trees, and
- an *obligatory adjunction* (OA) constraint – Boolean valued

and $I \subset E$ is a distinguished non-empty set of initial trees. Each elementary tree has a foot node.

Formally let Λ be a set of linguistic labels and Na be a finite set of labels disjoint from Λ (the set of names of trees). A tree is a pair (D, λ) where

- D is a tree domain,
- $\lambda : D \to \Lambda \times \wp(Na) \times \{\mathsf{true}, \mathsf{false}\}$ a labelling function.

Hence a node is labelled by a triple consisting of a linguistic label, an SA constraint, and an OA constraint. We denote $\mathscr{T}_{\Lambda,Na}$ the set of all trees. An *elementary* tree is a triple (D, λ, f) where (D, λ) is a tree and $f \in D$ is a leaf node, the *foot node*.

Definition 2 A non-strict TAG is a quintuple $G = (\Lambda, Na, E, I, name)$ where

- Λ is a set of labels,
- Na is a finite set of tree names,
- E is a finite set of elementary trees,
- $I \subseteq E$ is a finite set of initial trees, and
- $name : E \to Na$ is a bijection, the tree naming function.

An adjunction is the operation of replacing a node n with a non-empty SA constraint by an elementary tree t listed in the SA constraint. The daughters of n become daughters of the

foot node of t. A substitution is like an adjunction except that n is a leaf and hence there are no daughters to be moved to the foot node of t.

Formally, let t, t' be two trees and G a non-strict TAG. Then $t' = (D_{t'}, \lambda_{t'})$ is derived from $t = (D_t, \lambda_t)$ in a single step (written $t \underset{G}{\Rightarrow} t'$) iff there is a position $p \in D_t$ and an elementary tree $s \in E$ with foot node f_s such that

- $\lambda_t(p) = (L, SA, OA)$ with $L \in \Lambda$, $SA \subseteq Na, OA \in \{\mathsf{true}, \mathsf{false}\}$,
- $D_{t'} = \{q \in D_t \mid \nexists v \in \mathbb{N}^+ : q = pv\} \cup \{pv \mid v \in D_s\} \cup \{p f_s v \mid v \in \mathbb{N}^+, pv \in D_t\}$,
- $\lambda_{t'}(q) = \begin{cases} \lambda_t(q) & \text{if } q \in D_t \text{ and } \nexists v \in \mathbb{N}^* : q = pv, \\ \lambda_s(v) & \text{if } v \in D_s \text{ and } q = pv, \\ \lambda_t(pv) & \text{if } v \in \mathbb{N}^+, pv \in D_t, q = p f_s v. \end{cases}$

We write $t' = adj(t, p, s)$ if t' is the result of adjoining s in t at position p. As usual, $\overset{*}{\underset{G}{\Rightarrow}}$ is the reflexive-transitive closure of $\underset{G}{\Rightarrow}$. Note that this definition also subsumes substitution. A substitution is just an adjunction at a leaf node.

A tree is in the language of a given grammar, iff every OA constraint on the way is fulfilled, i.e., no node of the tree is labelled with true as OA constraint.

SA and OA constraints only play a role in derivations, they should not appear as labels of trees of the tree language generated by a TAG. Let π_1 be the first projection on a triple. It can be extended in a natural way to apply to trees by setting

- $D_{\pi_1(t)} = D_t$, and for each $p \in D_t$,
- $\lambda_{\pi_1(t)}(p) = L$ if $\lambda_t(p) = (L, SA, OA)$ for some $SA \subseteq Na, OA \in \{\mathsf{true}, \mathsf{false}\}$.

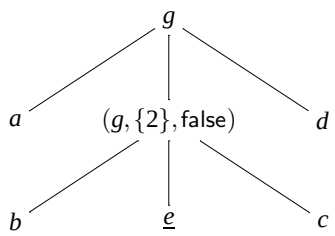
Now

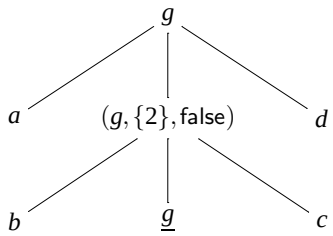
$$L(G) = \left\{ \pi_1(t) \;\middle|\; \begin{array}{l} \exists s \in I \text{ such that } s \overset{*}{\underset{G}{\Rightarrow}} t, \\ \nexists p \in D_t \text{ with } \lambda_t(p) = (L, SA, \mathsf{true}) \text{ for some } L \in \Lambda, SA \subseteq Na \end{array} \right\}.$$

One of the differences between TAGs and CFTGs is that there is no such concept of a non-terminal symbol or node in TAGs. The thing that comes closest is a node labelled with an OA constraint set to true. Such a node must be further expanded. The opposite is a node with an empty SA constraint. Such a node is a terminal node, because it must not be expanded. Nodes labelled with an OA constraint set to false but a non-empty SA constraint may or may not be expanded. They can neither be regarded as terminal nor as non-terminal nodes.

Example 2 Let $G_2 = (\{g, a, b, c, d, e\}, \{1, 2\}, E, \{name^{-1}(1)\}, name)$ where E and $name$ are as follows:

1 :



2 : 

To enhance readability we simplified node labels for all nodes that have an empty SA constraint. For these nodes we only present the linguistic label, the information $(\emptyset, \mathsf{false})$ is omitted. The foot nodes are underlined. Note that this grammar is even a strict TA grammar. Note also that it generates the same tree language as the MLCFTG from Example 1, i.e., $L(G_2) = L(G_1)$.

2.4 Three-Dimensional Trees

We introduce the concept of *three-dimensional* trees to provide a logical characterisation of the tree languages generable by a monadic linear CFTG. Multi-dimensional trees, their logics, grammars and automata are thoroughly discussed by Rogers (2003). Here, we just quote those technical definitions to provide our results. The reader who wishes to gain a better understanding of the concepts and formalisms connected with multi-dimensional trees is kindly referred to (Rogers 2003).

Formally, a three-dimensional tree domain $T3 \subset_{\text{fin}} (\mathbb{N}^*)^*$ is a finite set of sequences where each element of a sequence is itself a sequence of natural numbers such that for all $u, v \in (\mathbb{N}^*)^*$ if $uv \in T3$ then $u \in T3$ (prefix closure) and for each $u \in (\mathbb{N}^*)^*$ the set $\{v \mid v \in \mathbb{N}^*, uv \in T3\}$ is a tree domain in the sense of Subsection 2.1.

Let Σ be a set of labels. A *three-dimensional tree* is a pair $(T3, \lambda)$ where $T3$ is a three-dimensional tree domain and $\lambda : T3 \to \Sigma$ is a (node) labelling function.

For a node $x \in T3$ we define its immediate successors in three dimensions as follows. $x \lhd_3 y$ iff $y = x \cdot m$ for some $m \in \mathbb{N}^*$, i.e., x is the longest proper prefix of y. $x \lhd_2 y$ iff $x = u \cdot m$ and $y = u \cdot mj$ for some $u \in T3, m \in \mathbb{N}^*, j \in \mathbb{N}$, i.e. x and y are at the same 3rd dimensional level, but x is the mother of y in a tree at that level. Finally, $x \lhd_1 y$ iff $x = u \cdot mj$ and $y = u \cdot m(j+1)$ for some $u \in T3, m \in \mathbb{N}^*, j \in \mathbb{N}$, i.e. x and y are at the same 3rd dimensional level and x is the immediate left sister of y in a tree at that level.

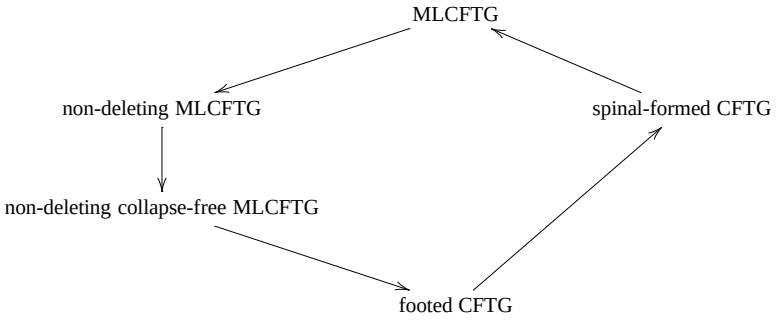
We consider the weak monadic second-order logic over the relations $\lhd_3, \lhd_2, \lhd_1$. Explanations about this logic and its relationship to T3 grammars and automata can be found in (Rogers 2003).

3 The Equivalence between MLCFTGs and Footed CFTGs

The equivalence between MLCFTGs and TAGs is proven by showing that both grammar formalisms are equivalent to a third formalism, so-called footed CFTGs.

Definition 3 Let $G = (\Sigma, \mathscr{F}, S, P)$ be a linear CFTG. For $k > 0$ a rule $F(x_1, \ldots, x_k) \to t$ is *footed* iff there exists a position $p \in D_t$ such that p has exactly k daughters, for $0 \leq i \leq k-1$: $\lambda(pi) = x_{i+1}$, and no position different from $\{p0, \ldots, p(k-1)\}$ is labelled with a variable. The node p is called the foot node and the path from the root of t to p is called the *spine* of t. A CFTG G is *footed* iff every rule of G expanding a symbol from $\mathscr{F}^k$ with $k > 0$ is footed.

Footed CFTGs are apparently the counterpart of non-strict TAGs in the world of context-free grammars. Before we show this, we present in this section that footed CFTGs are actually equivalent to MLCFTGs. This is done in several intermediate steps, which are sketched in the following diagram.



Each arrow indicates that a grammar of one type can be equivalently recasted as a grammar of the target type. We first show how to convert a MLCFTG into an equivalent footed CFTG in several steps. The start is the observation by Fujiyoshi that deletion rules do not contribute to the expressive power of MLCFTGs.

Proposition 1 (Fujiyoshi 2005) *For every monadic linear context-free tree grammar there exists an equivalent* non-deleting *monadic linear context-free tree grammar.*

But even in a non-deleting MLCFTG there may still be rules that delete nodes in a derivation step. Let $G = (\Sigma, \mathscr{F}, S, P)$ be a non-deleting MLCFTG. A rule $A(x) \to x$ in P is called a *collapsing* rule. A collapsing rule actually deletes the non-terminal node A in a tree. Iff a CFTG does not contain a collapsing rule, the grammar is called *collapse-free*. Note that by definition footed CFTGs are collapse-free, because the tree on the righthand side of a collapsing rule provides no position having daughters (cp. Def. 3). Note also that the example MLCFTG in Ex. 1 is non-deleting and collapse-free. The next proposition shows that collapsing rules can be eliminated from MLCFTGs.

Proposition 2 *For every non-deleting MLCFTG there exists an equivalent non-deleting* collapse-free *MLCFTG.*

The propostion can be proven by stepwise applying all collapsing rules to the right hand sides (or rhs, for short) of the non-collapsing rules.

MLCFTGs are not necessarily footed CFTGs, even when they are non-deleting and collapse-free. The reason is the following. Every right-hand side of every rule of a non-deleting MLCFTG has exactly one occurrence of the variable x. But this variable may have sisters. I.e. there may be subtrees in the rhs which have the same mother as x. Such a rule is apparently not footed. And its rhs can hardly be used as a base for an elementary tree in a TAG. Fortunately, though, a non-deleting collapse-free MLCFTG can be transformed into an equivalent footed CFTG. The resulting footed CFTG is usually not monadic any more. But this does not constitute any problem when translating the footed CFTG into a TAG.

The main idea in the transformation is the following. Let $B(x) \to t$ be a non-footed grammar rule containing the subtree $g(t_1, x, t_2)$. The undesirable sister subtrees t_1 and t_2 are replaced by variables yielding a new rule $B(x_1, x_2, x_3) \to t'$ where t' is the result of replacing

the subtree $g(t_1,x,t_2)$ by $g(x_1,x_2,x_3)$. The new rule is footed, but now ternary, not monadic. So is the non-terminal B. The original sister subtrees t_1 and t_2 still have to be dealt with. Suppose there is a grammar rule $D(x) \to \tau$ such that τ contains a subtree $B(\theta)$. In this rhs we replace $B(\theta)$ by $B(t_1,\theta,t_2)$. Now the non-terminal B is also ternary in the rhs, and the modified grammar rule can be applied to it. And if we apply the modified grammar rule, the trees t_1 and t_2 are moved back to being sisters of θ and daughters of the node g below which they were originally found.

The technical proof is illustrated by an example following it.

Proposition 3 *For every non-deleting collapse-free MLCFTG there exists an equivalent footed CFTG.*

Proof Let CFTG $G = (\Sigma, \mathscr{F}, S, P)$ be a non-deleting collapse-free MLCFTG. The transformation proceeds in two major steps. First step:
Let $(A(x) \to \tau) \in P$ and $f(t_1,\dots t_k)$ be a subtree of τ such that $k > 1$, $f \in \Sigma^k$, $t_j = x$ for some $1 \le j \le k$ and $t_i \in \mathscr{T}_{\Sigma\cup\mathscr{F}}$ for $i \ne j$. For each $1 \le i \le k, i \ne j$ we introduce a new non-terminal $T_i \notin \mathscr{F}^0$ of rank 0 and a new rule $T_i \to t_i$. Rule $A(x) \to \tau$ is replaced by $A(x) \to \tau'$ where τ' is the result of replacing the subtree $f(t_1,\dots t_k)$ by $f(T_1,\dots,T_{j-1},x,T_{j+1},\dots,T_k)$.
This step does not change the generated tree language nor the type of the grammar. It is just helpful in the formulation of the next step.

Second step:
Let CFTG $G = (\Sigma, \mathscr{F}, S, P)$ be a non-deleting collapse-free MLCFTG after Step 1.
Let $(A(x) \to \tau) \in P$ and $f(T_1,\dots T_{j-1},x,T_{j+1},\dots,T_k)$ be a subtree of τ for some k and $f \in \Sigma^k$.
Consider the rule set $R = \{A(x) \to t\} \subseteq P$ of rules with $A(x)$ as their lhs. It can be divided into two disjoint parts. Let $R_1 =$

$$\{A(x) \to t \mid t \text{ contains a subtree } g(T_1,\dots T_{j-1},x,T_{j+1},\dots,T_n) \text{ for some } j,n \text{ and } g \in \Sigma^n\}$$

and $R_2 = R \setminus R_1$. None of the rules in R_1 are footed, while all rules in R_2 are. Define R_1' by

$$\left\{ A'(x_1,\dots,x_n) \to t' \;\middle|\; \begin{array}{l} (A(x) \to t) \in R_1, \\ t \text{ contains a subtree } g(T_1,\dots T_{j-1},x,T_{j+1},\dots,T_n) \\ \qquad \text{for some } j,n \text{ and } g \in \Sigma^n \\ A' \notin \mathscr{F} \text{ is a new nonterminal of rank } n \\ t' \text{ is the result of replacing } g(T_1,\dots T_{j-1},x,T_{j+1},\dots,T_n) \\ \qquad \text{in } t \text{ by } g(x_1,\dots,x_n) \end{array} \right\}.$$

Note that R_1' consists of footed rules only. We also define a set W of rewrite rules as follows.
$W_1 =$

$$\left\{ A(x) \to A'(T_1,\dots T_{j-1},x,T_{j+1},\dots,T_n) \;\middle|\; \begin{array}{l} (A(x) \to t) \in R_1, t \text{ contains a subtree} \\ g(T_1,\dots T_{j-1},x,T_{j+1},\dots,T_n) \\ \text{for some } j,n \text{ and } g \in \Sigma^n \end{array} \right\}.$$

If $R_2 = \emptyset$ set $W_2 = \emptyset$. If $R_2 \ne \emptyset$ set $W_2 = \{A(x) \to A(x)\}$. Now $W = W_1 \cup W_2$.
Let $P_1 = (P \setminus R_1) \cup R_1'$ be a set of grammar rules. We apply the rewrite rules in W to the grammar rules in P_1 to get the desired new grammar P'. Define $app(W, B(x_1,\dots,x_k) \to t) =$

$$\left\{ B(x_1,\dots,x_k) \to t' \;\middle|\; \begin{array}{l} t' \text{ is the result of rewriting every occurrence of } A \\ \text{in } t \text{ by some rule from } W \end{array} \right\}.$$

Different occurrences of A in t can be rewritten by different rules from W. Now,

$$P' = \bigcup_{(B(x_1,\ldots,x_k)\to t)\in P_1} app(W, B(x_1,\ldots,x_k)\to t).$$

The new grammar is $G' = (\Sigma, \mathscr{F}', S, P')$ where $\mathscr{F}' \supseteq \mathscr{F}$ contains all new nonterminals introduced in the definition of R_1'. Rule set P' contains only footed rules for the lhs $A(x)$ and for all new lhs derived from $A(x)$ in the definition of R_1'.

Claim 1: If $S \overset{*}{\underset{G}{\Rightarrow}} t$ for some $t \in \mathscr{T}_{\Sigma\cup\mathscr{F}}$ then there is a $t' \in \mathscr{T}_{\Sigma\cup\mathscr{F}'}$ such that $S \overset{*}{\underset{G'}{\Rightarrow}} t'$ and t can be rewritten to t' via rules from W.

Claim 1 is proven by induction on the length of the derivation of t.

For $S \overset{*}{\underset{G}{\Rightarrow}} S$ this is trivially true.

Let $S \overset{*}{\underset{G}{\Rightarrow}} t$. Then there is a $s \in \mathscr{T}_{\Sigma\cup\mathscr{F}}$ with $S \overset{*}{\underset{G}{\Rightarrow}} s \underset{G}{\Rightarrow} t$. Thus there is a $\sigma \in \mathscr{T}_{\Sigma\cup\mathscr{F}\cup X_1}$ and a tree $\sigma_1 \in \mathscr{T}_{\Sigma\cup\mathscr{F}}$ and a rule $(B(x)\to\xi) \in P$ such that $s = \sigma[B[\sigma_1]]$ and $t = \sigma[\xi[\sigma_1]]$.

We may assume that $(B(x)\to\xi) \in R_1$, the other case being simpler.

We assume ξ contains a subtree $g(T_1,\ldots,T_{j-1},x,T_{j+1},\ldots T_k)$ for some $j,k \in \mathbb{N}$ and $g \in \Sigma^k$.

Hence R_1' contains a rule $B'(x_1,\ldots,x_k)\to\xi'$ where ξ' is the result of replacing the subtree $g(T_1,\ldots,T_{j-1},x,T_{j+1},\ldots T_k)$ in ξ by $g(x_1,\ldots,x_k)$. And W contains a rewrite rule $B(x) \to B'(T_1,\ldots,T_{j-1},x,T_{j+1},\ldots T_k)$.

By Ind.H., there is a tree $s' \in \mathscr{T}_{\Sigma\cup\mathscr{F}'}$ with $S \overset{*}{\underset{G'}{\Rightarrow}} s'$ and s can be rewritten to s' via rules from W.

Thus $s' = \sigma'[B'[T_1,\ldots,T_{j-1},\sigma_1',T_{j+1},\ldots T_k]]$ where σ (resp. σ_1) can be rewritten to σ' (resp. σ_1') via rules from W.

Thus grammar rule $B'(x_1,\ldots,x_k)\to\xi'$ can be applied to s' yielding $\sigma'[\xi'[T_1,\ldots,T_{j-1},\sigma_1',T_{j+1},\ldots T_k]] = t'$ and t can be rewritten to t' via rules from W.

Note that this implies that $L(G) \subseteq L(G')$.

Claim 2: If $S \overset{*}{\underset{G'}{\Rightarrow}} t'$ for some $t' \in \mathscr{T}_{\Sigma\cup\mathscr{F}'}$ then there is a $t \in \mathscr{T}_{\Sigma\cup\mathscr{F}}$ such that $S \overset{*}{\underset{G}{\Rightarrow}} t$ and t can be rewritten to t' via rules from W.

Claim 2 is proven by induction on the length of the derivation of t'.

For $S \overset{*}{\underset{G'}{\Rightarrow}} S$ this is trivially true.

Let $S \overset{*}{\underset{G'}{\Rightarrow}} t'$. Then there is an $s' \in \mathscr{T}_{\Sigma\cup\mathscr{F}'}$ with $S \overset{*}{\underset{G'}{\Rightarrow}} s' \underset{G'}{\Rightarrow} t'$. Thus there is a $\sigma' \in \mathscr{T}_{\Sigma\cup\mathscr{F}'\cup X_k}$ and trees $\sigma_1',\ldots,\sigma_k' \in \mathscr{T}_{\Sigma\cup\mathscr{F}'}$ and a rule $(B'(x_1,\ldots,x_k)\to\xi) \in P'$ such that $s' = \sigma'[B'[\sigma_1',\ldots,\sigma_k']]$ and $t' = \sigma'[\xi[\sigma_1',\ldots,\sigma_k']]$.

We assume the grammar rule $B'(x_1,\ldots,x_k)\to\xi$ to be the result of transforming a unary non-footed rule. Thus there exists a $j \in \mathbb{N}$ and a $g \in \Sigma^k$ and a grammar rule $(B(x)\to\xi') \in P$ such that ξ' contains a subtree $g(T_1,\ldots,T_{j-1},x,T_{j+1},\ldots T_k)$ (for suitable non-terminals $T_1,\ldots,T_{j-1},T_{j+1},\ldots T_k$ of arity 0) and ξ is the result of first replacing this subtree by $g(x_1,\ldots,x_k)$ and then applying some rewrite rules from W. And W contains a rewrite rule $B(x)\to B'(T_1,\ldots,T_{j-1},x,T_{j+1},\ldots T_k)$.

This implies that $\sigma_l' = T_l$ for $1 \le l \le k, l \ne j$.

By Ind.H., there is a tree $s \in \mathscr{T}_{\Sigma\cup\mathscr{F}}$ with $S \overset{*}{\underset{G}{\Rightarrow}} s$ and s can be rewritten to s' by rules from W.

Thus $s = \sigma[B[\sigma_j]$ where σ (resp. σ_j) can be rewritten to σ' (resp. σ_j') via rules from W. And grammar rule $(B(x)\to\xi') \in P$ can be applied to s yielding $\sigma[\xi'[\sigma_j]] = t$. That t can be rewritten to t' follows from $\sigma_l' = T_l$ and the construction of ξ'.

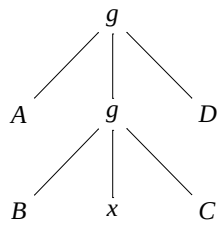
Note that this implies that $L(G') \subseteq L(G)$.

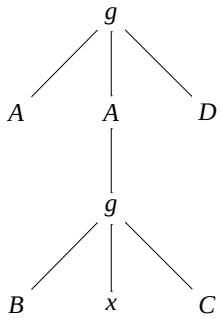
Claims 1 and 2 show that $L(G) = L(G')$.
By repetition of Step 2, all non-footed rules can be replaced in P. □

Note that the finally resulting footed CFTG may have a lot more rules than the original MLCFTG. We'd like to illuminate the construction by means of an example.

Example 3 We convert the non-deleting collapse-free MLCFTG G_1 from Example 1 into a footed CFTG. In Step 1, we replace subtrees which are sisters of variables by new non-terminals. This step is not really useful in G_1, but we show it anyway. The resulting grammar rules are

$S \rightarrow A(e)$

$A(x) \rightarrow$ 

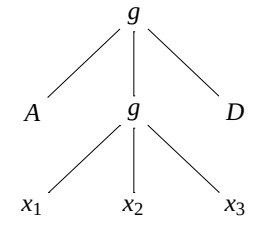
$A(x) \rightarrow$ 

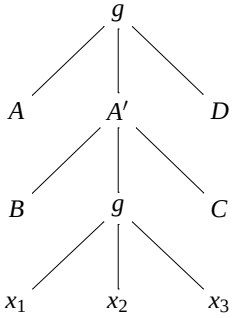
$A \rightarrow a$
$B \rightarrow b$
$C \rightarrow c$
$D \rightarrow d$

This grammar contains non-footed rules, the two rules that expand $A(x)$. Hence we have to apply Step 2.
Rule set $R_1 = \{A(x) \rightarrow g(A,g(B,x,C),D), A(x) \rightarrow g(A,A(g(B,x,C)),D)\}$.
Rule set $R_2 = \emptyset$ is empty.
Now $R'_1 =$
$\{A'(x_1,x_2,x_3) \rightarrow g(A,g(x_1,x_2,x_3),D), A'(x_1,x_2,x_3) \rightarrow g(A,A(g(x_1,x_2,x_3)),D)\}$.
And $W = W_1 = \{A(x) \rightarrow A'(B,x,C)\}$.

Applying W to $(P \setminus R_1) \cup R'_1$ yields the new grammar

$S \rightarrow A'(B,e,C)$

$A'(x_1,x_2,x_3) \rightarrow$ 

$A'(x_1,x_2,x_3) \rightarrow$ 

$A \rightarrow a$
$B \rightarrow b$
$C \rightarrow c$
$D \rightarrow d$

We call the resulting footed grammar G_2.

Having shown by now that there is an equivalent footed CFTG for every MLCFTG we will now turn to the inverse direction. This is also done via intermediate steps.

The following definitions are quoted from (Fujiyoshi and Kasai 2000, p. 62). A ranked alphabet is *head-pointing*, iff it is a triple (Σ, ρ, h) such that (Σ, ρ) is a ranked alphabet and h is a function from Σ to $\mathbb{N}$ such that, for each $A \in \Sigma$, if $\rho(A) \geq 1$ then $0 \leq h(A) < \rho(A)$, otherwise $h(A) = 0$. The integer $h(A)$ is called the head of A.

Definition 4 Let $G = (\Sigma, \mathscr{F}, S, P)$ be a CFTG such that $\mathscr{F}$ is a head-pointing ranked alphabet. For $n \geq 1$, a production $A(x_1, \ldots, x_n) \to t$ in P is *spinal-formed* iff it satisfies the following conditions:

- There is exactly one leaf in t that is labelled by $x_{h(A)}$. The path from the root to that leaf is called the spine of t, or the spine when t is obvious.
- For a node $d \in D_t$, if d is on the spine and $\lambda(d) = B \in \mathscr{F}$ with $\rho(B) \geq 1$, then $d \cdot h(B)$ is a node on the spine.
- Every node labelled by a variable in $X_n \setminus \{x_{h(A)}\}$ is a child of a node on the spine.

A CFTG $G = (\Sigma, \mathscr{F}, S, P)$ is *spinal-formed* iff every production $A(x_1, \ldots, x_n) \to t$ in P with $n \geq 1$ is spinal-formed.

The intuition behind this definition as well as illustrating examples can be found in (Fujiyoshi and Kasai 2000, p. 63). We will not quote them here, because spinal-formed CFTGs are just an equivalent form of CFTGs on the way to showing that footed CFTGs can be rendered by MLCFTGs.

Proposition 4 *For every footed CFTG there exists an equivalent spinal-formed CFTG.*

Note that a rule in footed CFTG fulfills the first and the third condition of a spinal-formed rule already. What has to be shown is that the set $\mathscr{F}$ of non-terminals can be made to be head-pointing. Since the rhs of a footed CFTG rule has a spine, the non-terminals on the spine can be made head-pointing by following the spine. For all other non-terminals we arbitrarily choose the first daughter to be the head daughter.

Proof Let $G = (\Sigma, \mathscr{F}, S, P)$ be a footed CFTG.
Define CFTG $G' = (\Sigma, \mathscr{F}', S, P')$ as follows.
Set $\mathscr{F}_1 = \{(A, 0) \mid A \in \mathscr{F}^{>0}\}$,
$\mathscr{F}_2 = \{(A, k) \mid A \in \mathscr{F}^{>0}, \exists t \in rhs(P), p \in D_t : \lambda_t(p) = A, pk \in spine(t)\}$, and
$\mathscr{F}' = \mathscr{F}^0 \cup \mathscr{F}_1 \cup \mathscr{F}_2$.

For every $(A, k) \in \mathscr{F}_1 \cup \mathscr{F}_2$ set $h(A, k) = k$ (the head of (A, k)).

Define relab : $rhs(P) \to \mathscr{T}_{\mathscr{F}' \cup \Sigma \cup X}$ as follows.

$$
\begin{aligned}
D &= D_t, \\
\text{for each } p \in D &: \\
\lambda_{\mathrm{relab}(t)}(p) &= \begin{cases} (A,k) & \text{if } \lambda_t(p) = A \in \mathscr{F}^{>0}, pk \in spine(t), \\ (A,0) & \text{if } \lambda_t(p) = A \in \mathscr{F}^{>0}, p \notin spine(t), \\ A & \text{if } \lambda_t(p) = A \in \mathscr{F}^0, \\ f & \text{if } \lambda_t(p) = f \in \Sigma \cup X \end{cases} \\
\mathrm{relab}(t) &= (D, \lambda_{\mathrm{relab}(t)})
\end{aligned}
$$

For trees $t \in \mathscr{T}_{\mathscr{F}' \cup \Sigma \cup X}$ the inverse of relab can be defined by

$$D = D_t,$$
$$\text{for each } p \in D:$$
$$\lambda_{\text{relab}^{-1}(t)}(p) = \begin{cases} A \text{ if } \lambda_t(p) = (A,k) \in \mathscr{F}_1 \cup \mathscr{F}_2, \\ A \text{ if } \lambda_t(p) = A \in \mathscr{F}^0, \\ f \text{ if } \lambda_t(p) = f \in \Sigma \cup X \end{cases}$$
$$\text{relab}^{-1}(t) = (D, \lambda_{\text{relab}^{-1}(t)})$$

Set

$$P' = \{(A,k)(x_1,\ldots,x_n) \to \text{relab}(t) \mid \exists (A(x_1,\ldots,x_n) \to t) \in P, (A,k) \in \mathscr{F}'\} \cup \{A \to relab(t) \mid \exists (A \to t) \in P, A \in \mathscr{F}^0\}.$$

Grammar G' is spinal-formed, as a simple check reveals.

Claim 1: For every tree $t \in \mathscr{T}_{\Sigma \cup \mathscr{F}}$: if $S \overset{*}{\underset{G}{\Rightarrow}} t$ then there exists a tree $t' \in \mathscr{T}_{\Sigma \cup \mathscr{F}'}$ with $S \overset{*}{\underset{G'}{\Rightarrow}} t'$ and $t = \text{relab}^{-1}(t')$.

Proven by an induction on the length of the derivation of t.

For $S \overset{*}{\underset{G}{\Rightarrow}} S$ this is trivially true.

Let $S \overset{*}{\underset{G}{\Rightarrow}} t$. Then there is a $s \in \mathscr{T}_{\Sigma \cup \mathscr{F}}$ with $S \overset{*}{\underset{G}{\Rightarrow}} s \underset{G}{\Rightarrow} t$. Thus there is a $\sigma \in \mathscr{T}_{\Sigma \cup \mathscr{F} \cup X_k}$ and trees $\sigma_1, \ldots \sigma_k \in \mathscr{T}_{\Sigma \cup \mathscr{F}}$ and a rule $(B(x_1, \ldots x_k) \to \xi) \in P$ such that $s = \sigma[B[\sigma_1, \ldots, \sigma_k]]$ and $t = \sigma[\xi[\sigma_1, \ldots, \sigma_k]]$.

By Ind.H., there is a tree $s' \in \mathscr{T}_{\Sigma \cup \mathscr{F}'}$ with $S \overset{*}{\underset{G'}{\Rightarrow}} s'$ and $s = \text{relab}^{-1}(s')$.

By definition of P' there is a rule $((B,l)(x_1, \ldots x_k) \to \text{relab}(\xi)) \in P'$. And there is a $\sigma' \in \mathscr{T}_{\Sigma \cup \mathscr{F}' \cup X_k}$ with $\sigma = \text{relab}^{-1}(\sigma')$ and trees $\sigma'_1, \ldots \sigma'_k \in \mathscr{T}_{\Sigma \cup \mathscr{F}'}$ with $\sigma_j = \text{relab}^{-1}(\sigma'_j)$ such that $s' = \sigma[(B,l)[\sigma'_1, \ldots, \sigma'_k]]$.

Therefore $s' \underset{G'}{\Rightarrow} t' = \sigma'[\text{relab}(\xi)[\sigma'_1, \ldots, \sigma'_k]]$ and $t = \text{relab}^{-1}(t)$.

The argument for $B \in \mathscr{F}^0$ is even simpler.

Claim 2: For evert tree $t \in \mathscr{T}_{\Sigma \cup \mathscr{F}'}$: if $S \overset{*}{\underset{G'}{\Rightarrow}} t$ then $S \overset{*}{\underset{G}{\Rightarrow}} \text{relab}^{-1}(t)$.

Claim 2 can be proven by a simple induction on the length of the derivation of t. Claims 1 and 2 together show that $L(G) = L(G')$. □

Proposition 5 (Fujiyoshi and Kasai 2000) *For every spinal-formed CFTG there exists an equivalent MLCFTG.*

This is a corollary of Theorem 1 (p. 65) of (Fujiyoshi and Kasai 2000). The authors see this fact themselves. They state on p. 65 immediately above Theorem 1:

> "It follows from Theorem 1 that the class of tree languages generated by spine grammars is the same as the class of tree languages generated by linear nondeleting monadic CFTGs, that is, CFTGs with nonterminals of rank 1 and 0 only, and with exactly one occurrence of x in every right-hand side of a production for a nonterminal of rank 1."

We are now done showing that MLCFTGs are equivalent to footed CFTGs.

Theorem 1 *A tree language is definable by a monadic linear CFTG if and only if it is definable by a footed CFTG.*

4 The Equivalence between Footed CFTGs and TAGs

The aim of this section is to show that footed CFTGs are indeed the counterpart of non-strict TAGs. The following proofs are technical, but the constructions are not really difficult. We first translate footed CFTGs into non-strict TAGs. The basic idea is that every right-hand side of every rule from the CFTG is an elementary tree. The new foot node is the node that is the mother of the variables in the rhs of a rule. Of course, the variables and the nodes bearing them have to be removed from the elementary tree. To construct the TAG, every rhs of the CFTG gets a name. Every non-terminal in a rhs receives an obligatory adjunction constraint. The selection adjunction constraint it receives is the set of names of those rhs that are the rhs of the rules that expand this non-terminal. The initial trees are the rhs of the rules that expand the start symbol of the CFTG.

Proposition 6 *For every footed CFTG there exists an equivalent non-strict TAG.*

Proof Let CFTG $G = (\Sigma, \mathscr{F}, S, P)$ be a footed CFTG. Let Na be a set of labels such that $|Na| = |P|$. Define a bijection $name : Na \to rhs(P)$ mapping names in Na to right hand side of rules in P in some arbitrary way.

For a non-terminal $A \in \mathscr{F}^k$ we define the set

$$Rhs_A = \{name(r) \mid (A(x_1, \ldots, x_k) \to r) \in P\}.$$

We define a function el-tree : $rhs(P) \to \mathscr{T}_{\Sigma \cup \mathscr{F}, Na}$ by considering two cases. For $(A(x_1, \ldots, x_k) \to t) \in P$ such that $f \in D_t$ with $\lambda_t(fi) = x_{i+1}$ set

$$\begin{aligned} D &= D_t \setminus \{fi \mid 0 \leq i \leq k-1\} \\ &\text{for each } p \in D: \\ \lambda(p) &= \begin{cases} (\lambda_t(p), \emptyset, \mathsf{false}) & \text{if } \lambda_t(p) \in \Sigma, \\ (B, Rhs_B, \mathsf{true}) & \text{if } \lambda_t(p) = B \in \mathscr{F} \end{cases} \\ \text{el-tree}(t) &= (D, \lambda, f) \end{aligned}$$

For $(A \to t) \in P$ set

$$\begin{aligned} D &= D_t \\ &\text{for each } p \in D: \\ \lambda(p) &= \begin{cases} (\lambda_t(p), \emptyset, \mathsf{false}) & \text{if } \lambda_t(p) \in \Sigma, \\ (B, Rhs_B, \mathsf{true}) & \text{if } \lambda_t(p) = B \in \mathscr{F} \end{cases} \\ f &= 0^k \text{ for } k \in \mathbb{N}, 0^k \in D, 0^{k+1} \notin D \\ \text{el-tree}(t) &= (D, \lambda, f) \end{aligned}$$

We let $G' = (\Sigma, Na, \{\text{el-tree}(r) \mid r \in rhs(P)\}, \{\text{el-tree}(S)\}, name)$ be the non-strict TAG derived from G. An example of the construction is given in Example 4, directly below this proof.

For a given footed CFTG G and derived TAG G' we can define a function tag-tree : $\mathscr{T}_{\Sigma \cup \mathscr{F}} \to \mathscr{T}_{\Sigma \cup \mathscr{F}, Na}$ from footed CFTG generated trees to TAG generated trees similar to the function el-tree as follows. For $t = (D_t, \lambda_t) \in \mathscr{T}_{\Sigma \cup \mathscr{F}}$ we set

$$\begin{aligned} D &= D_t \\ &\text{for each } p \in D: \\ \lambda(p) &= \begin{cases} (\lambda_t(p), \emptyset, \mathsf{false}) & \text{if } \lambda_t(p) \in \Sigma, \\ (B, Rhs_B, \mathsf{true}) & \text{if } \lambda_t(p) = B \in \mathscr{F} \end{cases} \\ \text{tag-tree}(t) &= (D, \lambda) \end{aligned}$$

The function tag-tree is partially the inverse of π_1 (the projection onto the first element of a tuple, defined in Def. 2 on p. 5), i.e., $\pi_1(\text{tag-tree}(t)) = t$ for every $t \in \mathscr{T}_{\Sigma\cup\mathscr{F}}$.

Claim 1: For every tree $t \in \mathscr{T}_{\Sigma\cup\mathscr{F}}$: if $S \overset{*}{\underset{G}{\Rightarrow}} t$ then el-tree$(S) \overset{*}{\underset{G'}{\Rightarrow}}$ tag-tree(t).

Proven by induction on the length of the derivation of t.
For $S \overset{*}{\underset{G}{\Rightarrow}} S$ the claim is true by definition of el-tree(S).

Let $t \in \mathscr{T}_{\Sigma\cup\mathscr{F}}$ and $S \overset{*}{\underset{G}{\Rightarrow}} t$. By definition of $\overset{*}{\underset{G}{\Rightarrow}}$ there is a tree s such that $S \overset{*}{\underset{G}{\Rightarrow}} s \underset{G}{\Rightarrow} t$ and a position $p \in D_s$. We distinguish two cases.
Case 1: p is not a leaf node.
Then there is a $B \in \mathscr{F}^k$, $\sigma \in \mathscr{T}_{\Sigma\cup\mathscr{F}}(X_1), \sigma_1, \ldots, \sigma_k \in \mathscr{T}_{\Sigma\cup\mathscr{F}}$, a rule $(B(x_1,\ldots,x_k) \to \xi) \in P$ such that $s = \sigma[B[\sigma_1,\ldots,\sigma_k]]$, $B[\sigma_1,\ldots,\sigma_k]$ is the subtree at position p, and $t = \sigma[\xi[\sigma_1,\ldots,\sigma_k]]$.
By Ind.H., el-tree$(S) \overset{*}{\underset{G'}{\Rightarrow}}$ tag-tree(s) and $\lambda_{\text{tag-tree}(s)}(p) = (B, Rhs_B, true)$.
By definition of G' there is an elementary tree el-tree(ξ) with $name(\text{el-tree}(\xi)) \in Rhs_B$. Therefore we can adjoint el-tree(ξ) at position p of tag-tree(s). By definition of adjoin, the result of this adjoin operation is just tag-tree$(\sigma[\xi[\sigma_1,\ldots,\sigma_k]])$ = tag-tree(t), and hence el-tree$(S) \overset{*}{\underset{G'}{\Rightarrow}}$ tag-tree(t).

Case 2: p is a leaf node.
Then there is a $B \in \mathscr{F}^0$, $\sigma \in \mathscr{T}_{\Sigma\cup\mathscr{F}}$, a rule $(B \to \xi) \in P$ such that $s = \sigma[B]$, B is the subtree at position p, and $t = \sigma[\xi]$.
By Ind.H., el-tree$(S) \overset{*}{\underset{G'}{\Rightarrow}}$ tag-tree(s) and $\lambda_{\text{tag-tree}(s)}(p) = (B, Rhs_B, true)$.
By definition of G' there is an elementary tree el-tree(ξ) with $name(\text{el-tree}(\xi)) \in Rhs_B$. Therefore we can substitute $(B, Rhs_B, true)$ with el-tree(ξ) at position p of tag-tree(s). By definition of substitution as a special case of adjoin, the result of this substitution operation is just tag-tree$(\sigma[\xi])$ = tag-tree(t), and hence el-tree$(S) \overset{*}{\underset{G'}{\Rightarrow}}$ tag-tree(t).

Claim 2: For every tree $t \in \mathscr{T}_{\Sigma\cup\mathscr{F},Na}$: if el-tree$(S) \overset{*}{\underset{G'}{\Rightarrow}} t$ then $S \overset{*}{\underset{G}{\Rightarrow}} \pi_1(t)$.

Proven by induction on the length of the derivation of t.
For el-tree$(S) \overset{*}{\underset{G'}{\Rightarrow}}$ el-tree(S) the claim is true by definition of el-tree(S).

Let $t \in \mathscr{T}_{\Sigma\cup\mathscr{F},Na}$ such that el-tree$(S) \overset{*}{\underset{G'}{\Rightarrow}} t$. By definition of $\overset{*}{\underset{G'}{\Rightarrow}}$ there exists a tree $s \in \mathscr{T}_{\Sigma\cup\mathscr{F},Na}$ such that el-tree$(S) \overset{*}{\underset{G'}{\Rightarrow}} s \underset{G'}{\Rightarrow} t$. We distinguish two cases.

Case 1: Step $s \underset{G'}{\Rightarrow} t$ is an adjunction step.
There is a position $p \in D_s$ and an elementary tree $e \in E$ with foot node f_s. By definition of G' $\lambda_s(p) = (B, Rhs_B, \mathsf{true})$ and $name(e) \in Rhs_B$.
Hence there is a rule $(B(x_1,\ldots,x_k) \to e') \in P$ with $e =$ el-tree(e').
Hence $\pi_1(s) \underset{G}{\Rightarrow} \pi_1(t)$ and there is a $\sigma \in \mathscr{T}_{\Sigma\cup\mathscr{F}}(X_1), \sigma_1, \ldots, \sigma_k \in \mathscr{T}_{\Sigma\cup\mathscr{F}}$ with $\pi_1(s) = \sigma[B[\sigma_1,\ldots,\sigma_k]]$ and $\pi_1(t) = \sigma[e'[\sigma_1,\ldots,\sigma_k]]$.
By Ind.H., $S \overset{*}{\underset{G}{\Rightarrow}} \pi_1(s)$. Therefore $S \overset{*}{\underset{G}{\Rightarrow}} \pi_1(t)$.

Case 2: Step $s \underset{G'}{\Rightarrow} t$ is a substitution step.

This case is similar to the adjunction step but simpler. Claims 1 and 2 together show that $L(G) = L(G')$. □

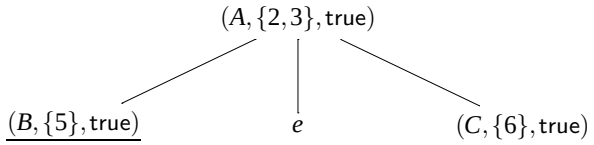
Example 4 To explain the construction in the proof above we transform the grammar G_3 at the end of Example 3. The names are $Na = \{1,2,3,4,5,6,7\}$ with *name* defined as

Na	rhs
1	$A(B,e,C)$
2	$g(A,g(x_1,x_2,x_3),D)$
3	$g(A,A(B,g(x_1,x_2,x_3),C),D)$

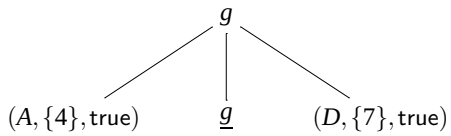
Na	rhs
4	a
5	b
6	c
7	d

We obtain the following elementary trees. (Again we simplify node labels of type $(L,\emptyset,\mathsf{false})$ to just L. Foot nodes are underlined.)

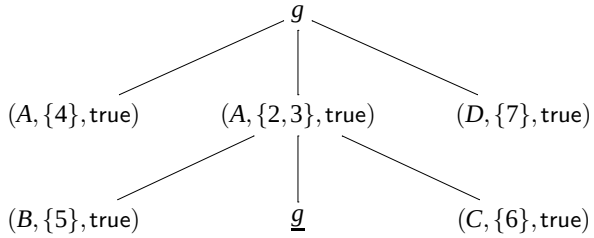
1 :



2 :



3 :



4: a 6: c
5: b 7: d

Tree 1 is the only initial tree.

If we substitute the substitution nodes 4 – 7 into the other elementary trees, the grammar bears a remarkable similarity the the TA grammar G_2 of Example 2. Tree 2 of G_2 corresponds to tree 3, and tree 1 of G_2 to the result of adjoining tree 2 into 1.

We now show the inverse direction. The idea of the construction is to take the elementary trees as right-hand sides of rules in a footed CFTG to be constructed. The non-terminals that are expanded – and hence the left-hand sides of rules – are those nodes that have an SA constraint that contains the name of the elementary tree under consideration. The arity of the non-terminal is just the number of daughters of such a node.

Proposition 7 *For every non-strict TAG there exists an equivalent footed CFTG.*

Proof Let $G = (\Sigma, Na, E, I, name)$ be a non-strict TAG.
Let $S \notin \Sigma$ be a new symbol (the new start symbol). Set

$$\mathscr{F}^k = \{(L,SA,v) \mid \exists t \in E \exists p \in D_t :\ \lambda_t(p) = (L,SA,v), v \in \{\mathsf{true},\mathsf{false}\}, SA \neq \emptyset,\\ p(k-1) \in D_t, pk \notin D_t\}.$$

Set $\mathscr{F} = \{S\} \cup \bigcup_{k\geq 0} \mathscr{F}^k$ the set of non-terminals. For an elementary tree $t = (D_t, \lambda_t, f) \in E$ we define $rhs(t,k)$ by

$$D = D_t \cup \{f\,j \mid 0 \leq j \leq k-1\}$$

for each $p \in D$:

$$\lambda(p) = \begin{cases} L & \text{if } \lambda_t(p) = (L, \emptyset, \mathsf{false}), L \in \Sigma, \\ (L, SA, v) & \text{if } \lambda_t(p) = (L, SA, v), L \in \Sigma, SA \neq \emptyset, \\ & v \in \{\mathsf{true}, \mathsf{false}\}, \\ x_{j+1} & \text{if } p = f\,j, 0 \leq j \leq k-1 \end{cases}$$

$$rhs(t,k) = (D, \lambda)$$

Note that for $k = 0$ the tree domain $D = D_t$. Define P_1 as

$$\{(L, SA, v)(x_1, \ldots, x_k) \to rhs(t,k) \mid (L, SA, v) \in \mathscr{F}^k, t \in E : name(t) \in SA\}$$
$$\cup\ \{S \to rhs(i,0) \mid i \in I\}$$

and P_2 as

$$\{(L, SA, \mathsf{false})(x_1, \ldots, x_k) \to L(x_1, \ldots, x_k) \mid$$
$$\exists t \in E\ \exists p \in t : \lambda_t(p) = (L, SA, \mathsf{false}), p(k-1) \in D_t, pk \notin D_t\}.$$

The set P of productions is $P_1 \cup P_2$. Let $G' = (\mathscr{F}, \Sigma, S, P)$ be a CFTG.
A simple check of the definition of the productions shows that G' is footed. Note that the rules in P_1 are used for the derivation proper while those in P_2 serve the purpose of stripping off the undesirable SA and OA constraint information coded in the non-terminals. The above construction is illustrated by Example 5 following immediately after the technical part of the proof.

Claim 1: For every tree $t \in \mathscr{T}_{\Sigma,Na}$: if $i \overset{*}{\underset{G}{\Rightarrow}} t$ with $i \in I$ then $S \overset{*}{\underset{G'}{\Rightarrow}} rhs(t,0)$.
Proven by an induction on the length of the derivation of t.
For $i \in I$, if $i \overset{*}{\underset{G}{\Rightarrow}} i$ then there is a rule $(S \to rhs(i,0)) \in P$ by definition of P. And hence $S \overset{*}{\underset{G'}{\Rightarrow}} rhs(i,0)$.
If $i \overset{*}{\underset{G}{\Rightarrow}} t$ with $i \in I$ then there is an $s \in \mathscr{T}_{\Sigma,Na}$, an $e \in E$ and a position $p \in D_s$ such that $i \overset{*}{\underset{G}{\Rightarrow}} s \underset{G}{\Rightarrow} t$ and $t = adj(s,p,e)$, $\lambda_s(p) = (L, SA, v)$ with $L \in \Sigma, name(e) \in SA, v \in \{\mathsf{true}, \mathsf{false}\}$.
Let $k = \max\{j \mid pj \in D_s\} + 1$.
By Ind,H,., $S \overset{*}{\underset{G'}{\Rightarrow}} rhs(s,0)$.
Furthermore $(L, SA, v) \in \mathscr{F}^k$, $\lambda_{rhs(s,0)}(p) = (L, SA, v)$, and $((L, SA, v)(x_1, \ldots, x_k) \to rhs(e,k)) \in P$ by definition of P. Hence $rhs(s,0) \underset{G'}{\Rightarrow} rhs(t,0)$.

Claim 2: $L(G) \subseteq L(G')$.
Let $t \in L(G)$. Then there is a $t' \in \mathscr{T}_{\Sigma,Na}$ and an $i \in I$ such that $i \overset{*}{\underset{G}{\Rightarrow}} t'$ and $t = \pi_1(t')$ and there is no position $p \in D_{t'}$ where $\lambda_{t'}(p) = (L, SA, \mathsf{true})$ for some $L \in \Sigma, SA \subseteq Na$. Now, $rhs(t',0) \overset{*}{\underset{G'}{\Rightarrow}} t$ using only rules from P_2 by definition of P_2 and t'. And $S \overset{*}{\underset{G'}{\Rightarrow}} rhs(t',0)$ by Claim 1. Hence $S \overset{*}{\underset{G'}{\Rightarrow}} t$ and $t \in L(G')$.

Claim 3: For every tree $t \in \mathscr{T}_{\Sigma \cup \mathscr{F}}$: if $S \overset{*}{\underset{G'}{\Rightarrow}} t$ using only productions from P_1 then there is a $i \in I$ and a $t' \in \mathscr{T}_{\Sigma,Na}$ such that $i \overset{*}{\underset{G}{\Rightarrow}} t'$ and $t = rhs(t',0)$.
Claim 3 can also be proven by an induction on the length of the derivation of t.

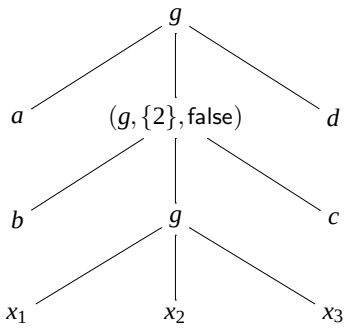
Claim 4: $L(G') \subseteq L(G)$.
Let $t \in \mathscr{T}_\Sigma$ such that $t \in L(G')$. Then $S \overset{*}{\underset{G'}{\Rightarrow}} t$. It is simple to see that there is a tree $s \in \mathscr{T}_{\Sigma \cup \mathscr{F}}$ such that there is a derivation sequence $S \overset{*}{\underset{G'}{\Rightarrow}} s \overset{*}{\underset{G'}{\Rightarrow}} t$ and every rule in $S \overset{*}{\underset{G'}{\Rightarrow}} s$ is in P_1 while every rule in $s \overset{*}{\underset{G'}{\Rightarrow}} t$ is in P_2. By Claim 3, there is an $i \in I$ and an $s' \in \mathscr{T}_{\Sigma,Na}$ such that $i \overset{*}{\underset{G}{\Rightarrow}} s'$ and $s = rhs(s',0)$. Since every rule in $s \overset{*}{\underset{G'}{\Rightarrow}} t$ is in P_2, there is no position p in s' such that $\lambda_{s'}(p) = (L, SA, \mathsf{true})$ for some $L \in \Sigma, SA \subseteq Na$. Hence $\pi_1(s') \in L(G)$ by definition of $L(G)$. But since every rule in $s \overset{*}{\underset{G'}{\Rightarrow}} t$ is in P_2, it follows that $\pi_1(s') = t$ by definition of P_2.

Claims 2 and 4 together show that $L(G) = L(G')$. □

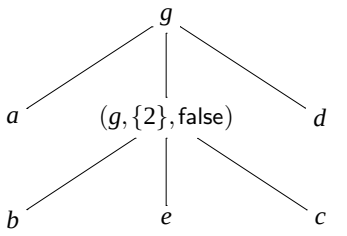
This completes the proof of the equivalence between footed CFTGs and non-strict TAGS.

Example 5 To illustrate the construction of the proof we provide an example of transforming a non-strict TAG into an equivalent footed CFTG. The input TAG is G_2 from Example 2.
The set of non-terminals is $\mathscr{F} = \{S, (g, \{2\}, \mathsf{false})^3\}$.
Rule set P_1 consists of two rules:

$(g, \{2\}, \mathsf{false})(x_1, x_2, x_3) \rightarrow$



$S \rightarrow$



Rule set P_2 consists of a single rule:

$$(g, \{2\}, \mathsf{false})(x_1, x_2, x_3) \rightarrow g(x_1, x_2, x_3)$$

The footed grammar is given by $(\{S, (g, \{2\}, \mathsf{false})\}, \{g, a, b, c, d, e\}, S, P_1 \cup P_2)$.

The above results are accumulated in the following two theorems.

Theorem 2 *A tree language is definable by a footed CFTG if and only if it is definable by a non-strict TAG.*

We can now present the main result of this paper. It is an immediate consequence of the theorem above and Theorem 1.

Theorem 3 *The class of tree languages definable by non-strict Tree Adjoining Grammars is exactly the class of tree languages definable by monadic linear context-free tree grammars.*

5 A Logical Characterisation

The aim of this section is to show that the theorem above and results by Rogers (2003) on TAGs can be combined to yield a logical characterisation of tree languages definable by monadic linear CFTGs. A tree language is generable by a MLCFTG iff it is the two-dimensional yield of an MSO-definable three-dimensional tree language.

We start by defining the two-dimensional yield of a three-dimensional tree. Let $(T3,\lambda)$ be a three-dimensional tree. A node $p \in T3$ is an internal node, iff $p \neq \varepsilon$ (p is not the root) and there is a p' with $p \lhd_3 p'$ (p has an immediate successor in the 3rd dimension). For an internal node we define a fold-in operation that replaces the node by the subtree it roots. Consider the set S of immediate successors of p. By definition it is a two-dimensional tree domain. We demand it to have a foot node, i.e., a distinguished node $f \in S$ that has no immediate successors in the second dimension. The operation replaces p by S.

Formally, let $t = (T3,\lambda)$ be a three-dimensional tree. We say t is *footed in the second dimension* iff for every node p the two-dimensional tree domain $\{p' \mid p \lhd_3 p'\}$ has a foot node.

Let $p \in T3$ be an internal node. Hence $p = p'm$ for some $p' \in T3$ (the immediate predecessor of p) and $m \in \mathbb{N}^*$. Let $p'mf \in T3$ be the foot node of the immediate successors of p where $f \in \mathbb{N}^*$. Define

$$\begin{aligned} T3' = {} & \{r \in T3 \mid \nexists r' \in (\mathbb{N}^*)^* : r = pr'\} \cup \\ & \{p'(m\cdot n)r \mid p'mnr \in T3, n \in \mathbb{N}^*, r \in (\mathbb{N}^*)^*\} \cup \\ & \{p'(m\cdot f\cdot n) \mid p'(m\cdot n) \in T3, n \in \mathbb{N}^*\}. \end{aligned}$$

The set in the first line is the set of node of which p is not a prefix. It is unchanged. The set in the second line is the set of successors of p in the 3rd dimension. They are folded in at the place of p. The set in the third line is the set of successor of p in the second dimension. They are appended to the folded-in foot node. The labelling of the nodes in $T3'$ is derived from the labelling of the originating nodes in $T3$.

$$\lambda'(s) = \begin{cases} \lambda(s) & \text{if } s \in T3 \mid \nexists r' \in (\mathbb{N}^*)^* : s = pr' \\ \lambda(p'mnr) & \text{if } s = p'(m\cdot n)r, n \in \mathbb{N}^*, r \in (\mathbb{N}^*)^*\} \\ \lambda(p'(m\cdot n)) & \text{if } s = p'(m\cdot f\cdot n), n \in \mathbb{N}^*. \end{cases}$$

Now we define fold-in$((T3,\lambda),p) = (T3',\lambda')$.

The operation is similar to an adjunction operation in a TAG derivation. It can be iterated until there is no internal node left. If choose$(T3)$ is a choice function that chooses an arbitrary internal node from a (three-dimensional) tree domain $T3$, then

$$\text{fold-in}(T3,\lambda) = \begin{cases} \text{fold-in}((T3,\lambda),\text{choose}(T3)) & \text{if there is an internal node in } T3 \\ (T3,\lambda) & \text{otherwise} \end{cases}$$

Now define recursively fold-in$^1(T3,\lambda) = \text{fold-in}(T3,\lambda)$ and fold-in$^{k+1}(T3,\lambda) = \text{fold-in}($ fold-in$^k(T3,\lambda))$. For every tree t there is a $k \in \mathbb{N}$ such that fold-in$^k(t) = \text{fold-in}^{k+1}(t)$ because there are no internal nodes left. Hence we can safely write fold-in$^\omega(t)$, because for every tree t the fixed point is reached after finitely many steps.

Now consider a tree t that has no internal nodes. It consists of the root and its immediate successors in the 3rd dimension. These form a two-dimensional tree. The two-dimensional yield of a three-dimensional tree t is the (two-dimensional) tree of the immediate successors of the root of fold-in$^\omega(t)$, i.e,

$$\text{yield}(t) = \{(p,\lambda(p)) \mid p \in \text{fold-in}^\omega(t), \varepsilon \lhd_3 p\}.$$

After this longish definition of a two-dimensional yield of a three-dimensional tree we can now state the main theorem of this section. It provides a logical characterisation of the tree languages definable by MLCFTGs.

Theorem 4 *A tree language is generable by a monadic linear context-free tree grammar iff it is the two-dimensional yield of an MSO-definable three-dimensional tree language.*

Proof Rogers (2003) showed in Theorems 5 and 13 that a tree language is generable by a non-strict TAG iff it is the two-dimensional yield of an MSO-definable three-dimensional tree language. The theorem is an immediate consequence of Rogers' result and our Theorem 3. □

6 Conclusion

We showed that non-strict TAGs and monadic linear CFTGs are strongly equivalent. We thereby rendered an old intuition about TAGs to be true (at least for non-strict ones). The strong equivalence result yields a new logical characterisation of the expressive power of monadic linear CFTGs. A tree language is definable by a MLCFTG iff it is the two-dimensional yield of an MSO-definable three-dimensional tree language.

It is known that there is a whole family of mildly context-sensitive grammar formalisms that all turned out to be weakly equivalent. It would be interesting to compare their relative expressive powers in terms of tree languages, because, finally, linguists are interested in linguistic analyses, i.e., tree languages, and not so much in unanalysed utterances. For string based formalisms, the notion of strong generative capacity has to be extended along the lines proposed by Miller (1999). The current paper is one step in a program of comparing the strong generative capacity of mildly context-sensitive grammar formalisms.

This research was in part funded by a grant of the German Research Council (DFG SFB-441). We would like to thank three anonymous referees for their comments and suggestions, which helped improving this paper.

References

Engelfriet J, Schmidt EM (1977) IO and OI. I. Journal of Computer and System Sciences 15(3):328–353
Fujiyoshi A (2005) Linearity and nondeletion on monadic context-free tree grammars. Information Processing Letters 93(3):103–107, DOI doi:10.1016/j.ipl.2004.10.008
Fujiyoshi A, Kasai T (2000) Spinal-formed context-free tree grammars. Theory of Computing Systems 33(1):59–83
Gécseg F, Steinby M (1997) Tree languages. In: Rozenberg G, Salomaa A (eds) Handbook of Formal Languages, Vol 3: Beyond Words, Springer-Verlag, pp 1–68
Joshi A, Schabes Y (1997) Tree adjoining grammars. In: Rozenberg G, Salomaa A (eds) Handbook of Formal Languages, Handbook of Formal Languages, vol 3: Beyond Words, Springer, Berlin, pp 69–123
Joshi A, Levy LS, Takahashi M (1975) Tree adjunct grammar. Journal of Computer and System Sciences 10(1):136–163
Kepser S, Mönnich U (2006) Closure properties of linear context-free tree languages with an application to optimality theory. Theoretical Computer Science 354(1):82–97
Lang B (1994) Recognition can be harder than parsing. Computational Intelligence 10:486–494
Miller PH (1999) Strong Generative Capacity: The Semantics of Linguistic Formalism. CSLI Publications
Mönnich U (1997) Adjunction as substitution. In: Kruijff GJ, Morill G, Oehrle R (eds) Formal Grammar '97, pp 169–178
Rogers J (1998) A Descriptive Approach to Language-Theoretic Complexity. CSLI Publications
Rogers J (2003) wMSO theories as grammar formalisms. Theoretical Computer Science 293(2):291–320