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

Stephan Kepser and Jim Rogers
CRC 441 Computer Science Department
University of Tübingen Earlham College
Tübingen, Germany Richmond, IN, USA

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.

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 (1992). 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.

The paper is organised as follows. The next section introduces context-free tree grammars and tree adjoining grammars. Section 3 provides the main result of this paper by describing how to code monadic linear CFTGs (MLCFTG henceforth) and their derivations by non-strict TAGs and non-strict TAGS by MLCFTGs. The next section presents a logical characterisation of the expressive power of MLCFTGs, which is a consequence of the main result of the paper. Due to lack of space all formal definitions and proofs are devised to a technical appendix.

2 CFTGs and TAGs

We assume familiarity with trees and tree domains (see Appendix) and 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 $G = (\Sigma, \mathcal{F}, S, P)$ where

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

As usual, we set $\sigma_1 \underset{G}{\Rightarrow} \sigma_2$ if and only if there is a production $F(x_1,\ldots,x_k) \to \tau$ such that σ_2 is obtained from σ_1 by replacing an occurrence of a subtree $F(\xi_1,\ldots,\xi_k)$ by the tree $\tau[\xi_1,\ldots,\xi_k]$ (details in the appendix). 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.

We define three subtypes of context-free tree grammars. A production $F(x_1,\ldots,x_k) \to \tau$ is called *linear*, if 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 $G = (\Sigma, \mathcal{F}, S, P)$ 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* if each variable $x_1,\ldots,x_k$ occurs in τ. A CFTG is non-deleting if each rule is non-deleting.

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

The second class of grammar formalisms we consider in this paper are Tree Adjoining Grammars. We are particularly interested in so-called non-strict Tree Adjoining Grammars. Non-strict TAGs were introduced by Rogers (1998, 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 label of the replaced node with the head node of the adjoined tree.

Definition 2 (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

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

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. A tree is in the language of a given grammar, if 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. Hence the tree language of a TAG is the set of tree generable by this TAG with all SA and OA constraints stripped off.

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.

3 The Equivalence of MLCFTGs and TAGs

In this section, we show how MLCFTGs and their derivations can be encoded in non-strict TAGs and vice verse. We start with the encoding of MLCFTGs. In general, CFTGs may have deleting rules. The application of such a rule leads to the deletion of complete subtrees generated in previous steps. This deletion cannot be rendered by TAGs. But for MLCFTGs there is a solution to this problem, as the following result shows.

Proposition 3 (Fujiyoshi, 2005) *For every monadic linear context-free tree grammar there exists an equivalent* non-deleting *monadic linear context-free tree grammar.*

We therefore assume our MLCFTGs to be non-deleting. Under this assumption an equivalent non-strict TAG can be constructed. The general construction idea is that each right-hand side of a rule is in some sense an elementary tree with the foot node being the mother of the variable node.

Proposition 4 *For every non-deleting MLCFTG there exists an equivalent non-strict TAG.*

The full proof is to be found in the appendix. The main idea is the following. The right hand side of each rule is taken as an elementary tree. The non-terminals in such a rhs have to be constrained by an obligatory adjunction constraint. The associated selective adjunction constraint contains those right hand sides that the non-terminal may be expanded into according to P. If the rhs contains the variable x, the mother of x is the foot node. If it does not contain x, then it is the rhs of a non-terminal of arity 0. Hence the rhs will only be used for substitution and not adjunction.

We will now show that for every TAG there exists an equivalent MLCFTG. To do this we define a special type of CFTGs, CFTGs that are very similar to non-strict TAGs.

Definition 5 Let $G = (\Sigma, \mathcal{F}, S, P)$ be a linear CFTG. A rule $F(x_1, \ldots, x_k) \to t$ is *footed* if there exists a position p in the domain of t such that p has exactly k daughters, for $1 \leq i \leq k : \lambda(pi) = x_i$, and no position different from $\{p1, \ldots, pk\}$ 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* if every rule of G is footed.

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

The full construction and proof can be found in the appendix, as well as the proof of the next result.

Proposition 7 *A tree language is definable by a footed CFTG if and only if it is definable by a monadic linear CFTG.*

We are now in the position to present the main result of our paper.

Theorem 8 *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.*

4 A Logical Characterisation of MLCFTGs

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 monadic linear CFTGs.

Theorem 9 *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.*

5 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.

References

Engelfriet, Joost and Erik Meineche Schmidt (1977). IO and OI. I. *Journal of Computer and System Sciences*, **15**(3):328–353.

Fujiyoshi, Akio (2005). Linearity and nondeletion on monadic context-free tree grammars. *Information Processing Letters*, **93**(3):103–107.

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

Gécseg, Ferenc and Magnus Steinby (1997). Tree languages. In Grzegorz Rozenberg and Arto Salomaa, eds., *Handbook of Formal Languages, Vol 3: Beyond Words*, pp. 1–68. Springer-Verlag.

Joshi, Aravind, L.S. Levy, and M. Takahashi (1975). Tree adjunct grammar. *Journal of Computer and System Sciences*, **10**(1):136–163.

Joshi, Aravind and Yves Schabes (1997). Tree adjoining grammars. In G. Rozenberg and A. Salomaa, eds., *Handbook of Formal Languages*, volume 3: Beyond Words of *Handbook of Formal Languages*, pp. 69–123. Springer, Berlin.

Kepser, Stephan and Uwe Mönnich (2006). Closure properties of linear context-free tree languages with an application to optimality theory. *Theoretical Computer Science*, **354**(1):82–97.

Lang, Bernard (1992). Recognition can be harder than parsing. In *Proc. of the 2nd Int. Workshop on Tree Adjoining Grammars*.

Miller, Philip H. (1999). *Strong Generative Capacity: The Semantics of Linguistic Formalism*. CSLI Publications.

Mönnich, Uwe (1997). Adjunction as substitution. In Geert-Jan Kruijff, Glyn Morill, and Richard Oehrle, eds., *Formal Grammar '97*, pp. 169–178.

Rogers, James (1998). *A Descriptive Approach to Language-Theoretic Complexity*. CSLI Publications.

Rogers, James (2003). wMSO theories as grammar formalisms. *Theoretical Computer Science*, **293**(2):291–320.

Technical Appendices

A Trees and Tree Grammars

A.1 Two-Dimensional Trees

We consider labelled finite ordered ranked trees. A tree is ordered if there is a linear order on the daughters of each node. A tree is ranked if 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, a set $D \subset_{\text{fin}} \mathbb{N}^*$ is called a *tree domain* iff for all $u,v \in \mathbb{N}^* : uv \in D \Rightarrow u \in D$ and for all $u \in \mathbb{N}^*, i \in \mathbb{N} : ui \in D \Rightarrow \forall j < i : uj \in D$. 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 $\mathcal{T}_\Sigma$. A tree language $L \subseteq \mathcal{T}_\Sigma$ is just a subset of $\mathcal{T}_\Sigma$.

A set Σ of labels is *ranked* if there is a function $\rho : \Sigma \to \mathbb{N}$ assigning each symbol an arity. If $t = (D,\lambda)$ is a tree of a ranked alphabet Σ then for each position $p \in D : \rho(\lambda(p)) = n \Rightarrow pn \in D, pm \notin D$ for every $m > n$.

If X is a set (of symbols) disjoint from Σ, then $\mathcal{T}_\Sigma(X)$ denotes the set of trees $\mathcal{T}_{\Sigma \cup X}$ 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 $k \geq 0, m \geq 0, t \in \mathcal{T}_\Sigma(X_k)$, and $t_1, \dots, t_k \in \mathcal{T}_\Sigma(X_m)$, we denote by $t[t_1, \dots, t_k]$ the result of *substituting* t_i for x_i in t. Note that $t[t_1, \dots, t_k]$ is in $\mathcal{T}_\Sigma(X_m)$. Note also that for $k = 0$, $t[t_1, \dots, t_k] = t$.

A.2 Context-Free Tree Grammars

For a context-free tree grammar $G = (\Sigma, \mathcal{F}, S, P)$ we define the direct derivation relation. Let $n \geq 0$ and let $\sigma_1, \sigma_2 \in \mathcal{T}_{\Sigma \cup \mathcal{F}}(X_n)$. 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 $\eta \in \mathcal{T}_{\Sigma \cup \mathcal{F}}(X_{n+1})$ containing *exactly one* occurrence of x_{n+1}, and trees $\xi_1, \dots, \xi_k \in \mathcal{T}_{\Sigma \cup \mathcal{F}}(X_n)$ 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 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* 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.

A.3 Tree Adjoining Grammars

We present now the formal definitions of non-strict TAGs and the tree languages they generate. 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 $\mathcal{T}_{\Lambda,Na}$ the set of all trees. An *elementary* tree is a triple $((D,\lambda,f)$ where (D,λ) is a tree and $f \in D$ is a leaf node, the *foot node*.

Definition 10 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' is derived from 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, if 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

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

A.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 A.1.

Let Σ be a set of labels. A *tree-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 $= u \cdot mj$ 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).

We next define the two-dimensional yield of a three-dimensional tree. Let $(T3, \lambda)$ be a tree-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 such that the immediate successors of p in the second dimension become the immediate successors of f in the second dimension.

Formally, let $t = (T3, \lambda)$ be a tree-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) = $ fold-in$(T3, \lambda)$ and fold-in$^{k+1}(T3, \lambda) = $ fold-in(fold-in$^k(T3, \lambda)$). For every tree t there is a $k \in \mathbb{N}$ such that fold-in$^k(t) = $ 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 tree-dimensional tree t is the (two-dimensional) tree of the immediate successors of the root of fold-in$^\omega(t)$, i.e,

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

B From MLCFTGs to TAGs

In this section, we provide the technical definitions and proofs of how MLCFTGs and their derivations can be encoded in non-strict TAGs.

Before we can show how to code MLCFTGs with non-strict TAGs we have to provide another restriction on the way the MLCFTGs look and show that this is not really a restriction. Let $G = (\Sigma, \mathcal{F}, S, P)$

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. Such a step cannot be performed in a TAG, which *never* deletes any structure built in a derivation. This problem can be overcome, because collapsing rules can be eliminated from non-deleting MLCFTGs. If a CFTG does not contain a collapsing rule, the grammar is called *collapse-free*.

Proposition 11 *For every non-deleting MLCFTG there exists an equivalent non-deleting collapse-free MLCFTG.*

The idea of the proof is to apply the collapsing rule to all right-hand sides. Thus it is no longer needed. Some care has to be taken, if there is another way to expand the non-terminal that can collapse.

PROOF. Let CFTG $G = (\Sigma, \mathcal{F}, S, P)$ be a non-deleting MLCFTG. Let $A \in \mathcal{F}^{\,1}$ such that $A(x) \to x$ in P. We distinguish two cases.
Case 1: There is no rule $A(x) \to t$ in P with $t \neq x$.
In this case, we apply the rule $A(x) \to x$ to all occurrences of A in all other right-hand sides of P obtaining P'. Then rule $A(x) \to x$ can be safely removed from P', because it does not contain any occurrence of A any more. The tree language generated by $(\Sigma, \mathcal{F}, S, P')$ is obviously the same as that of G.

Case 2: There is a rule $A(x) \to t$ in P with $t \neq x$.
Let $B(x) \to t'$ be a rule in P that contains k occurrences of A in t'. It will be replaced by the set

$$\mathcal{B} = \{B(x) \to t'' \mid t'' \text{ is the result of applying } A(x) \to x \text{ to some occurrences of } A \text{ in } t'\}.$$

The cardinality of this set is 2^k. This step is performed for every rule that has some occurrence of A in its rhs. The new rule set P' is P union all sets $\mathcal{B}$. Then rule $A(x) \to x$ can be safely removed from P'. The tree language generated by $(\Sigma, \mathcal{F}, S, P')$ is obviously the same as that of G. □

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.

Proposition 12 *For every non-deleting collapse-free MLCFTG there exists an equivalent footed CFTG.*

PROOF. Let CFTG $G = (\Sigma, \mathcal{F}, S, P)$ be a non-deleting collapse-free MLCFTG. The transformation proceeds in two major steps. First step:
Let $(A(x) \to \tau) \in P$ and $f(t_1, \ldots t_k)$ be a subtree of τ such that $k > 1$, $f \in \Sigma^k$, $t_j = x$ for some $1 \leq j \leq k$ and $t_i \in \mathcal{T}_{\Sigma \cup \mathcal{F}}$ for $i \neq j$. For each $1 \leq i \leq k, i \neq j$ we introduce a new non-terminal $T_i \notin \mathcal{F}^{\,0}$ 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 $f(t_1, \ldots t_k)$ by $f(T_1, \ldots, T_{j-1}, x, T_{j+1}, \ldots, T_k)$. 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 $G = (\Sigma, \mathcal{F}, S, P)$ be a non-deleting collapse-free MLCFTG after Step 1.
Let $(A(x) \to \tau) \in P$ and $f(T_1, \ldots T_{j-1}, x, T_{j+1}, \ldots, T_k)$ be a subtree of τ.
We distinguish two cases.
Case 1: There is no other rule in P with $A(x)$ as its lhs.
Firstly, rule $A(x) \to \tau$ is replaced by $A(x_1, \ldots, x_k) \to \tau'$ where τ' is the result of replacing the subtree $f(T_1, \ldots T_{j-1}, x, T_{j+1}, \ldots, T_k)$ by $f(x_1, \ldots, x_k)$.
Secondly, consider the term rewrite rule $A(x) \to A(T_1, \ldots T_{j-1}, x, T_{j+1}, \ldots, T_k)$. This term rewrite rule is applied to all occurrences of a subtree headed by A of all rhs of P (including τ').

Case 2: There is another rule in P with $A(x)$ as its lhs.
This case is more complicated, because we cannot just rewrite all occurrences of subtrees headed by A. Again, rule $A(x) \to \tau$ is replaced by $A(x_1, \ldots, x_k) \to \tau'$ where τ' is the result of replacing the subtree $f(T_1, \ldots T_{j-1}, x, T_{j+1}, \ldots, T_k)$ by $f(x_1, \ldots, x_k)$ yielding new rule set P''.
Secondly, consider the term rewrite rule $r = (A(x) \to A(T_1, \ldots T_{j-1}, x, T_{j+1}, \ldots, T_k))$. For a grammar rule $g = (B(x_1, \ldots, x_n) \to t)$ in P'' define

$$app(r, g) = \{B(x_1, \ldots, x_n) \to t' \mid t' \text{ is the result of applying } r \text{ to some occurrences of } A \text{ in } t\}.$$

This set also contains the original grammar rule $B(x_1, \ldots, x_n) \to t$ for no occurrence chosen to apply r to. The new grammar rule set P' is defined as

$$\bigcup_{(B(x_1,\ldots,x_n)\to t)\in P''} app(r, B(x_1, \ldots, x_n) \to t).$$

The new grammar $G' = (\Sigma, \mathcal{F} \cup \{A^k\}, S, P')$ generates the same tree language as G, as can be shown by an induction on the length of the derivation. And P' contains one non-footed rule less than P. 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. □

Proposition 13 *For every footed CFTG there exists an equivalent non-strict TAG.*

Before providing the proof we would like to point out that in a footed CFTG there cannot be a collapsing rule $A(x) \to x$, because there is no position p having daughters (cp. Def. 5). This fact is implicitely used in the proof below.

PROOF. Let CFTG $G = (\Sigma, \mathcal{F}, S, P)$ 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 $A \in \mathcal{F}^k$ 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 : $rhs(P) \to \mathcal{T}_{\Sigma \cup \mathcal{F}, Na}$ by considering two cases. For $(A(x_1, \ldots, x_k) \to t) \in P$ such that $f \in D_t$ with $\lambda_t(fi) = x_i$ set

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

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

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

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.

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

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

The function tag-tree is partially the inverse of π_1, i.e., $\pi_1(\text{tag-tree}(t)) = t$ for every $t \in \mathcal{T}_{\Sigma \cup \mathcal{F}}$.

Claim 1: For every tree $t \in \mathcal{T}_{\Sigma \cup \mathcal{F}}$: 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 $t \in \mathcal{T}_{\Sigma \cup \mathcal{F}}$ 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 $B \in \mathcal{F}^k$, $\sigma \in \mathcal{T}_{\Sigma \cup \mathcal{F}}(X_1), \sigma_1, \ldots, \sigma_k \in \mathcal{T}_{\Sigma \cup \mathcal{F}}$, 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 $B \in \mathcal{F}^0$, $\sigma_1 \in \mathcal{T}_{\Sigma \cup \mathcal{F}}$, a rule $(B \to \xi) \in P$ such that $s = \sigma_1[B]$, B is the subtree at position p, and $t = \sigma_1[\xi]$.
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$(\sigma_1[\xi]) =$ tag-tree(t), and hence el-tree$(S) \overset{*}{\underset{G'}{\Rightarrow}}$ tag-tree(t).

Claim 2: For every tree $t \in \mathcal{T}_{\Sigma \cup \mathcal{F}, Na}$: 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 $t \in \mathcal{T}_{\Sigma \cup \mathcal{F}, Na}$ such that el-tree$(S) \overset{*}{\underset{G'}{\Rightarrow}} t$. By definition of $\overset{*}{\underset{G'}{\Rightarrow}}$ there exists a tree $s \in \mathcal{T}_{\Sigma \cup \mathcal{F}, Na}$ 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 $\sigma \in \mathcal{T}_{\Sigma \cup \mathcal{F}}(X_1), \sigma_1, \ldots, \sigma_k \in \mathcal{T}_{\Sigma \cup \mathcal{F}}$ 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.
There is a leaf node $p \in D_s$ 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 $(B \to e') \in P$ with $e =$ el-tree(e').
Hence $\pi_1(s) \underset{G}{\Rightarrow} \pi_1(t)$ and there is a $\sigma_1 \in \mathcal{T}_{\Sigma \cup \mathcal{F}}$ with $\pi_1(s) = \sigma_1[B]$ and $\pi_1(t) = \sigma_1[e']$.
By Ind.H., $S \overset{*}{\underset{G}{\Rightarrow}} \pi_1(s)$. Therefore $S \overset{*}{\underset{G}{\Rightarrow}} \pi_1(t)$.

Claims 1 and 2 together show that $L(G) = L(G')$. □

C From TAGs to MLCFTGs

In this section, we present the proof that for every TAG there exists an equivalent MLCFTG. This happens in several steps. First, we show that TAGs can be rendered as footed CFTGs. Then we introduce spinal formed CFTGs. This type of CFTGs was defined by Fujiyoshi and Kasai (2000). We show that for every footed CFTG there exists an equivalent spinal-formed CFTG. We proceed quoting a result by Fujiyoshi and Kasai (2000) that states that for every spinal-formed CFTG there exists an equivalent MLCFTG.

Proposition 14 *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. Set

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

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

$$\begin{array}{rcl} D & = & D_t \cup \{f\,j \mid 1 \leq j \leq k\} \\ & & \text{for each } p \in D: \\ \lambda(p) & = & \begin{cases} L & \text{if } \lambda_t(p) = (L, \emptyset, \mathsf{false}), L \in \Sigma, \\ (L, SA, v) & \text{if } \lambda_t(p) = (L, \emptyset, v), L \in \Sigma, SA \neq \emptyset, v \in \{\mathsf{true}, \mathsf{false}\}, \\ x_j & \text{if } p = f\,j, 1 \leq j \leq k \end{cases} \\ rhs(t,k) & = & (D, \lambda) \end{array}$$

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

$$\begin{array}{rl} & \{(L, SA, v)(x_1, \ldots, x_k) \to rhs(t,k) \mid (L, SA, v) \in N^k, t \in E : name(t) = SA\} \\ \cup & \{S \to rhs(i,0) \mid i \in I\} \end{array}$$

and P_2 as

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

The set P of productions is $P_1 \cup P_2$. Let $G' = (N, \Sigma, S, P)$ be a CFTG.
A simple check of the definition of the productions shows that G' is footed.

Claim 1: For every tree $t \in \mathcal{T}_{\Sigma,Na}$: 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 $s \in \mathcal{T}_{\Sigma,Na}$, 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 $k = \max\{j \mid pj \in D_s\}$.
By Ind,H,., $S \overset{*}{\underset{G'}{\Rightarrow}} rhs(s,0)$.
Furthermore $(L, SA, v) \in N^k$, $\lambda_{rhs(s,0)}(p) = (L, SA, v)$, 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 $t' \in \mathcal{T}_{\Sigma,Na}$ 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 $t \in \mathcal{T}_{\Sigma \cup N}$: if $S \overset{*}{\underset{G'}{\Rightarrow}} t$ using only productions from P_1 then there is a $i \in I$ and a $t' \in \mathcal{T}_{\Sigma,Na}$ such that $i \overset{*}{\underset{G}{\Rightarrow}} t'$ and $t = rhs(t',0)$.
Proven by an induction on the length of the derivation of t.
If $S \underset{G'}{\Rightarrow} t$ then there is an $i \in I$ such that $t = rhs(i,0)$ by definition of P_1. And hence $i \overset{*}{\underset{G}{\Rightarrow}} i$.
If $S \overset{*}{\underset{G'}{\Rightarrow}} t$ using only productions from P_1, then there is an $s \in \mathcal{T}_{\Sigma \cup N}$ such that $S \overset{*}{\underset{G'}{\Rightarrow}} s \underset{G'}{\Rightarrow} t$ using only productions from P_1. Thus there is a position $p \in D_s$ a label $(L, SA, v) \in N^k$ with

$L \in \Sigma, v \in \{\mathsf{true}, \mathsf{false}\}, \lambda_s(p) = (L, SA, v)$. There is a rule $((L, SA, v) \to \xi) \in P_1$ and trees $\sigma \in \mathcal{T}_{\Sigma \cup N}(X_1), \sigma_1, \ldots, \sigma_k \in \mathcal{T}_{\Sigma \cup N}$ such that $s = \sigma[(L, SA, v)[\sigma_1, \ldots, \sigma_k]]$ and $t = \sigma[\xi[\sigma_1, \ldots, \sigma_k]]$.
By Ind.H., there is an $i \in I$ and $s' \in \mathcal{T}_{\Sigma, Na}$ such that $i \overset{*}{\underset{G}{\Rightarrow}} s'$ and $s = rhs(s', 0)$, $\lambda_{s'}(p) = (L, SA, v)$.
By definition of P_1 there is an $e \in E$ such that $\xi = rhs(e, k)$ and $name(e) \in SA$. Now, $s \underset{G}{\Rightarrow} adj(s', p, e)$ and $t = rhs(adj(s', p, e), 0)$. Hence $i \overset{*}{\underset{G}{\Rightarrow}} adj(s', p, e)$.

Claim 4: $L(G') \subseteq L(G)$.
Let $t \in \mathcal{T}_\Sigma$ such that $t \in L(G')$. Then $S \overset{*}{\underset{G'}{\Rightarrow}} t$. It is simple to see that there is a tree $s \in \mathcal{T}_{\Sigma \cup N}$ 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 $s' \in \mathcal{T}_{\Sigma, Na}$ 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 $L(G) = L(G')$. □

The following definitions are quoted from (Fujiyoshi and Kasai, 2000, p. 62). A ranked alphabet is *head-pointing,* if it is a tripe (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 $1 \leq h(A) \leq \rho(A)$, otherwise $h(A) = 0$. The integer $h(A)$ is called the head of A.

Definition 15 Let $G = (N, \Sigma, S, P)$ be a CFTG such that N 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* if 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 $\lambda(d) = B \in N$ 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 $G = (N, \Sigma, S, P)$ is *spinal-formed* if 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 TAGs can be rendered by MLCFTGs.

Proposition 16 *For every footed CFTG there exists an equivalent spinal-formed CFTG.*

PROOF. Let $G = (N, \Sigma, S, P)$ be a footed CFTG.
Define CFTG $G' = (N', \Sigma, S, P')$ as follows.
Set $N_1 = \{(A, 1) \mid A \in N^{>0}\}$,
$N_2 = \{(A, k) \mid A \in N^{>0}, \exists t \in rhs(P), p \in D_t : \lambda_t(p) = A, pk \in spine(t)\}$, and
$N' = N^0 \cup N_1 \cup N_2$.

For every $(A,k) \in N_1 \cup N_2$ set $h(A,k) = k$ (the head of (A,k)).

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

$$\begin{aligned} D &= D_t, \\ &\quad \text{for each } p \in D: \\ \lambda_{\text{relab}(t)}(p) &= \begin{cases} (A,k) & \text{if } \lambda_t(p) = A \in N^{>0}, pk \in spine(t), \\ (A,1) & \text{if } \lambda_t(p) = A \in N^{>0}, p \notin spine(t), \\ A & \text{if } \lambda_t(p) = A \in N^0, \\ f & \text{if } \lambda_t(p) = f \in \Sigma \cup X \end{cases} \\ \text{relab}(t) &= (D, \lambda_{\text{relab}(t)}) \end{aligned}$$

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

$$\begin{aligned} D &= D_t, \\ &\quad \text{for each } p \in D: \\ \lambda_{\text{relab}^{-1}(t)}(p) &= \begin{cases} A & \text{if } \lambda_t(p) = (A,k) \in N_1 \cup N_2, \\ A & \text{if } \lambda_t(p) = A \in N^0, \\ f & \text{if } \lambda_t(p) = f \in \Sigma \cup X \end{cases} \\ \text{relab}^{-1}(t) &= (D, \lambda_{\text{relab}(t)}) \end{aligned}$$

Set

$$\begin{aligned} P' = &\{(A,k)(x_{1,\ldots,x_n}) \to \text{relab}(t) \mid \exists (A(x_1,\ldots,x_n) \to t) \in P, (A,k) \in N'\} \cup \\ &\{A \to relab(t) \mid \exists (A \to t) \in P, A \in A^0\}. \end{aligned}$$

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

Claim 1: For every tree $t \in \mathcal{T}_{\Sigma \cup N}$: if $S \overset{*}{\underset{G}{\Rightarrow}} t$ then there exists a tree $t' \in \mathcal{T}_{\Sigma \cup N'}$ 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 $s \in \mathcal{T}_{\Sigma \cup N}$ with $S \overset{*}{\underset{G}{\Rightarrow}} s \underset{G}{\Rightarrow} t$. Thus there is a $\sigma \in \mathcal{T}_{\Sigma \cup N \cup X_k}$ and trees $\sigma_1, \ldots \sigma_k \in \mathcal{T}_{\Sigma \cup N}$ 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 $s' \in \mathcal{T}_{\Sigma \cup N'}$ 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 $\sigma' \in \mathcal{T}_{\Sigma \cup N' \cup X_k}$ with $\sigma = \text{relab}^{-1}(\sigma')$ and trees $\sigma'_1, \ldots \sigma'_k \in \mathcal{T}_{\Sigma \cup N'}$ 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 $B \in N^0$ is even simpler.

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

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 $s \in \mathcal{T}_{\Sigma \cup N'}$ with $S \overset{*}{\underset{G'}{\Rightarrow}} s \underset{G'}{\Rightarrow} t$. Thus there is a $\sigma \in \mathcal{T}_{\Sigma \cup N' \cup X_k}$ and trees $\sigma_1, \ldots \sigma_k \in$

$\mathcal{T}_{\Sigma\cup N'}$ and a rule $((B,l)(x_1,\ldots x_k)\to\xi)\in P'$ such that $s=\sigma[B[\sigma_1,\ldots,\sigma_k]]$ and $t=\sigma_1[\xi[\sigma_1,\ldots,\sigma_k]]$.
By Ind.H., $S\overset{*}{\underset{G}{\Rightarrow}}\text{relab}^{-1}(s)$.

By definition of P' there is a rule $(B(x_1,\ldots x_k)\to\text{relab}^{-1}(\xi))\in P$. And $\text{relab}^{-1}(s)=\text{relab}^{-1}(\sigma)[B[\text{relab}^{-1}(\sigma_1),\ldots,\text{relab}^{-1}(\sigma_k)]]$.
Therefore $\text{relab}^{-1}(s)\underset{G'}{\Rightarrow}\text{relab}^{-1}(\sigma)[\text{relab}^{-1}(\xi)[\text{relab}^{-1}(\sigma_1),\ldots,\text{relab}^{-1}(\sigma_k)]]=\text{relab}^{-1}(t)$.

The argument for $B\in N^0$ is even simpler.

Claims 1 and 2 together show that $L(G)=L(G')$. □

Proposition 17 (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). To see this, it suffices to inspect the normal form of Theorem 1 and see that it defines indeed a monadic linear CFTG.

We have now also proven Proposition 7. It was shown in the above Propositons that for every footed CFTG there exists an equivalent MLCFTG. The inverse direction was shown in Section B in Propositions 3, 11, and 12.

D A Logical Characterisation

Theorem 18 *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 8. □