Multiple Context-Free Tree Grammars: Lexicalization and Characterization

Joost Engelfriet[a], Andreas Maletti[b], Sebastian Maneth[c]

[a] *LIACS, Leiden University, P.O. Box 9512, 2300 RA Leiden, The Netherlands*
[b] *Institute of Computer Science, Universität Leipzig, P.O. Box 100 920, 04009 Leipzig, Germany*
[c] *Department of Mathematics and Informatics, Universität Bremen, P.O. Box 330 440, 28334 Bremen, Germany*

Abstract

Multiple (simple) context-free tree grammars are investigated, where "simple" means "linear and non-deleting". Every multiple context-free tree grammar that is finitely ambiguous can be lexicalized; i.e., it can be transformed into an equivalent one (generating the same tree language) in which each rule of the grammar contains a lexical symbol. Due to this transformation, the rank of the nonterminals increases at most by 1, and the multiplicity (or fan-out) of the grammar increases at most by the maximal rank of the lexical symbols; in particular, the multiplicity does not increase when all lexical symbols have rank 0. Multiple context-free tree grammars have the same tree generating power as multi-component tree adjoining grammars (provided the latter can use a root-marker). Moreover, every multi-component tree adjoining grammar that is finitely ambiguous can be lexicalized. Multiple context-free tree grammars have the same string generating power as multiple context-free (string) grammars and polynomial time parsing algorithms. A tree language can be generated by a multiple context-free tree grammar if and only if it is the image of a regular tree language under a deterministic finite-copying macro tree transducer. Multiple context-free tree grammars can be used as a synchronous translation device.

Contents

1. Introduction

Multiple context-free (string) grammars (MCFG) were introduced in [87] and, independently, in [92] where they are called (string-based) linear context-free rewriting systems (LCFRS). They are of interest to computational linguists because they can model cross-serial dependencies, whereas they can still be parsed in polynomial time and generate semi-linear languages. Multiple context-free *tree* grammars were introduced in [57], in the sense that it is suggested in [57, Section 5] that they are the hyperedge-replacement context-free graph grammars in tree generating normal form, as defined in [27]. Such graph grammars generate the same string languages as MCFGs [21, 94]. It is shown in [57] that they generate the same tree languages as second-order abstract categorial grammars (2ACG), generalizing the fact that MCFGs generate the same string languages as 2ACGs [82]. It is also observed in [57] that the set-local multi-component tree adjoining grammar (MC-TAG, see [53, 93]), well-known to computational linguists, is roughly the monadic restriction of the multiple context-free tree grammar, just as the tree adjoining grammar (TAG, see [49, 51]) is roughly the monadic restriction of the (linear and nondeleting) context-free tree grammar, see [37, 61, 71]. We note that the multiple context-free tree grammar could also be called the *tree-based* LCFRS; such tree grammars were implicitly envisioned already in [92].

In this paper we define the multiple context-free tree grammars (MCFTG) in terms of familiar concepts from tree language theory (see, e.g., [41, 42]), and we base our proofs on elementary properties of trees and tree homomorphisms. Thus, we do not use other formalisms such as graph grammars, λ-calculus, or logic programs. Since the relationship between MCFTGs and the above type of graph grammars is quite straightforward, it follows from the results of [27] that the tree languages generated by MCFTGs can be characterized as the images of the regular tree languages under deterministic finite-copying macro tree transducers (see [26, 34, 39]). However, since no full version of [27] ever appeared in a journal, we present that characterization here (Theorem 76). It generalizes the well-known fact that the string languages generated by MCFGs can be characterized as the yields of the images of the regular tree languages under deterministic finite-copying top-down tree transducers, cf. [94]. These two characterizations imply (by a result from [26]) that the MCFTGs have the same string generating power as MCFGs, through the yields of their tree languages. We also give a direct proof of this fact (Corollary 70), and show how it leads to polynomial time parsing algorithms for MCFTGs (Theorem 72). All trees that have a given string as yield, can be viewed as "syntactic trees" of that string. A parsing algorithm computes, for a given string, one syntactic tree (or all syntactic trees) of that string in the tree language generated by the grammar. It should be noted that, due to its context-free nature, an MCFTG, like a TAG, also has derivation trees (or parse trees), which show the way in which a tree is generated by the rules of the grammar. A derivation tree can be viewed as a meta level tree and the derived syntactic tree as an object level tree, cf. [51]. In fact, the parsing algorithm computes a derivation tree (or all derivation trees) for the given string, and then computes the corresponding syntactic tree(s).

We define the MCFTG as a straightforward generalization of the MCFG, based on tree substitution rather than string substitution, where a (second-order) tree substitution is a tree homomorphism. However, our formal syntactic definition of the MCFTG is closer to the one of the context-free tree grammar (CFTG) as in, e.g., [31, 37, 42, 58, 61, 81, 90]. Just as for the MCFG, the semantics of the MCFTG is a least fixed point semantics, which can easily be viewed as a semantics based on parse trees (Theorem 9). Moreover, we provide a rewriting semantics for MCFTGs (similar to the one for CFTGs and similar to the one in [78] for MCFGs) leading to a usual notion of derivation, for which the derivation trees then equal the parse trees (Theorem 19). Intuitively, an MCFTG G is a *simple* (i.e., linear and nondeleting) context-free tree grammar (spCFTG) in which several nonterminals are rewritten in one derivation step. Thus every rule of G is a sequence of rules of an spCFTG, and the left-hand side nonterminals of these rules are rewritten simultaneously. However, a sequence of nonterminals can only be rewritten if (earlier in the derivation) they were introduced explicitly as such by the application of a rule of G. Therefore, each rule of G must also specify the sequences of (occurrences of) nonterminals in its right-hand side that may later be rewritten. This restriction is called "locality" in [53, 78, 93].

Apart from the above-mentioned results (and some related results), our main result is that MCFTGs can be lexicalized (Theorem 44). Let us consider an MCFTG G that generates a tree language $L(G)$ over the ranked alphabet Σ, and let $\Delta \subseteq \Sigma$ be a given set of *lexical items*. We say that G is *lexicalized* (with respect to Δ) if every rule of G contains at least one lexical item (or anchor). Lexicalized grammars are of importance for several reasons. First, a lexicalized grammar is often more understandable, because the rules of the grammar can be grouped around the lexical items. Each rule can then be viewed as lexical information on its anchor, demonstrating a syntactical construction in which the anchor can

occur. Second, a lexicalized grammar defines a so-called dependency structure on the lexical items of each generated object, allowing to investigate certain aspects of the grammatical structure of that object, see [64]. Third, certain parsing methods can take significant advantage of the fact that the grammar is lexicalized, see, e.g., [86]. In the case where each lexical item is a symbol of the string alphabet (i.e., has rank 0), each rule of a lexicalized grammar produces at least one symbol of the generated string. Consequently, the number of rule applications (i.e., derivation steps) is clearly bounded by the length of the input string. In addition, the lexical items in the rules guide the rule selection in a derivation, which works especially well in scenarios with large alphabets (cf. the detailed account in [10]).

We say that G is *finitely ambiguous* (with respect to Δ) if, for every $n \geq 0$, $L(G)$ contains only finitely many trees with n occurrences of lexical items. For simplicity, let us also assume here that every tree in $L(G)$ contains at least one lexical item. Obviously, if G is lexicalized, then it is finitely ambiguous. Our main result is that for a given MCFTG G it is decidable whether or not G is finitely ambiguous, and if so, a lexicalized MCFTG G' can be constructed that is (strongly) equivalent to G, i.e., $L(G') = L(G)$. Moreover, we show that G' is grammatically similar to G, in the sense that their derivation trees are closely related: every derivation tree of G' can be translated by a finite-state tree transducer into a derivation tree of G for the same syntactic tree, and vice versa. To be more precise, this can be done by a linear deterministic top-down tree transducer with regular look-ahead ($\mathrm{LDT^R}$-transducer). We say that G and G' are $\mathrm{LDT^R}$-equivalent. Since the class of $\mathrm{LDT^R}$-transductions is closed under composition, this is indeed an equivalence relation for MCFTGs. Note that, due to the $\mathrm{LDT^R}$-equivalence of G' and G, any parsing algorithm for G' can be turned into a parsing algorithm for G by translating the derivation trees of G' in linear time into derivation trees of G, using the $\mathrm{LDT^R}$-transducer. Thus, the notion of $\mathrm{LDT^R}$-equivalence is similar to the well-known notion of cover for context-free grammars (see, e.g., [46, 74]). For context-free grammars, no $\mathrm{LDT^R}$-transducer can handle the derivation tree translation that corresponds to the transformation into Greibach Normal Form. In fact, our lexicalization of MCFTGs generalizes the transformation of a context-free grammar into Operator Normal Form as presented in [46], which is much simpler than the transformation into Greibach Normal Form.

The *multiplicity* (or *fan-out*) of an MCFTG is the maximal number of nonterminals that can be rewritten simultaneously in one derivation step. The lexicalization of MCFTGs, as discussed above, increases the multiplicity of the grammar by at most the maximal rank of the lexical symbols in Δ. When viewing an MCFTG as generating a string language, consisting of the yields of the generated trees, it is natural that all lexical items are symbols of rank 0, which means that they belong to the alphabet of that string language. The lexicalization process is then called strong lexicalization, because it preserves the generated tree language (whereas weak lexicalization just requires preservation of the generated string language). Thus, strong lexicalization of MCFTGs does not increase the multiplicity. In particular spCFTGs, which are MCFTGs of multiplicity 1, can be strongly lexicalized as already shown in [70]. Note that all TAG tree languages can be generated by spCFTGs [61]. Although TAGs can be weakly lexicalized (see [36]), they cannot be strongly lexicalized, which was unexpectedly shown in [65]. Thus, from the lexicalization point of view, spCFTGs have a significant advantage over TAGs. The strong lexicalization of MCFTGs (with lexical symbols of rank 0) is presented without proof (and without the notion of $\mathrm{LDT^R}$-equivalence) in [25].

The *width* of an MCFTG is the maximal rank of its nonterminals. The lexicalization of MCFTGs increases the width of the grammar by at most 1.

In addition to the above results we compare the MCFTGs with the MC-TAGs and prove that they have ("almost") the same tree generating power, as also presented in [25]. It is shown in [61] that "non-strict" TAGs, which are a slight generalization of TAGs, generate the same tree languages as monadic spCFTGs, where 'monadic' means width at most 1; i.e., all nonterminals have rank 1 or 0. We confirm and strengthen the above-mentioned observation in [57] by showing that both MCFTGs and monadic MCFTGs have the same tree generating power as non-strict MC-TAGs (Theorems 49 and 61), with a polynomial increase of multiplicity. Since the constructions preserve lexicalized grammars, we obtain that non-strict MC-TAGs can be (strongly) lexicalized. Note that by a straightforward generalization of [65] it can be shown that non-strict TAGs cannot be strongly lexicalized. Then we show that even (strict) MC-TAGs have the same tree generating power as MCFTGs (Theorem 58). To be precise, if L is a tree language generated by an MCFTG, then the tree language $\#(L) = \{\#(t) \mid t \in L\}$ can be generated by an MC-TAG, where $\#$ is a "root-marker" of rank 1. This result settles a problem stated in [93, Section 4.5].[1] It also implies

[1] In the first paragraph of that section, Weir states that "it would be interesting to investigate whether there exist LCFRS's with object level tree sets that cannot be produced by any MCTAG."

that, as opposed to TAGs, MC-TAGs can be (strongly) lexicalized (Theorem 60).

It is shown in [60, 95] that 2ACGs, and in particular tree generating 2ACGs, can be lexicalized (for $\Delta = \Sigma$). Although 2ACGs and MCFTGs generate the same tree languages, this does not imply that MCFTGs can be lexicalized. It is shown in [83] that multi-dimensional TAGs can be strongly lexicalized. Although it seems that for every multi-dimensional TAG there is an MCFTG generating the same tree language (see the Conclusion of [58]), nothing else seems to be known about the relationship between multi-dimensional TAGs and MC-TAGs or MCFTGs.

The structure of this paper is as follows. Section 2 consists of preliminaries, mostly on trees and tree homomorphisms. Since a sequence of nonterminals of an MCFTG generates a sequence of trees, we also consider sequences of trees, called forests. The substitution of a forest for a sequence of symbols in a forest is realized by a tree homomorphism. In Section 3 we define the MCFTG, its least fixed point semantics (in terms of forest substitution), its derivation trees, and its derivations. Every derivation tree yields a tree, called its value, and the tree language generated by the grammar equals the set of values of its derivation trees. The set of derivation trees is itself a regular tree language. We recall the notion of an $\mathrm{LDT}^{\mathrm{R}}$-transducer, and we define two MCFTGs to be $\mathrm{LDT}^{\mathrm{R}}$-equivalent if there is a value-preserving $\mathrm{LDT}^{\mathrm{R}}$-transducer from the derivation trees of one grammar to the other, and vice versa. Section 4 contains a number of normal forms. For every MCFTG we construct an $\mathrm{LDT}^{\mathrm{R}}$-equivalent MCFTG in such a normal form. In Section 4.1 we discuss some basic normal forms, such as permutation-freeness which means that application of a rule cannot permute subtrees. In Section 4.2 we prove that every MCFTG can be transformed into Growing Normal Form (generalizing the result of [89, 90] for spCFTGs). This means that every derivation step increases the sum of the number of terminal symbols and the number of "big nonterminals" (which are the sequences of nonterminals that form the left-hand sides of the rules of the MCFTG). It even holds for finitely ambiguous MCFTGs, with 'terminal' replaced by 'lexical' (Theorem 37). Thus, this result is already part of our lexicalization procedure. Moreover, we prove that finite ambiguity is decidable. Section 5 is devoted to the remaining, main part of the lexicalization procedure. It shows that every MCFTG in (lexical) Growing Normal Form can be transformed into an $\mathrm{LDT}^{\mathrm{R}}$-equivalent lexicalized MCFTG. The intuitive idea is to transport certain lexical items from positions in the derivation tree that contain more than one lexical item (more precisely, that are labeled with a rule of the grammar that contains more than one lexical item), up to positions that do not contain any lexical item. In Section 6.1 we prove that MCFTGs have the same tree generating power as non-strict MC-TAGs. We define non-strict MC-TAGs as a special type of MCFTGs, namely "footed" ones, which (as in [61]) are permutation-free MCFTGs such that in every rule the arguments of each left-hand side nonterminal are all passed to one node in the right-hand side of the rule. Then we prove in Section 6.2 that (strict) MC-TAGs have the same tree generating power as MCFTGs, as explained above, and we show that MC-TAGs can be strongly lexicalized. In Section 6.3 we observe that every MC-TAG (and hence every MCFTG) can be transformed into an equivalent MCFTG of width at most 1, which is in contrast to the fact that spCFTGs (and arbitrary context-free tree grammars) give rise to a strict hierarchy with respect to width, as shown in [30, Theorem 6.5] (see also [67, Lemma 24]). In all the results of Section 6 the constructed grammar is $\mathrm{LDT}^{\mathrm{R}}$-equivalent to the given one. In Section 7.1 we define the multiple context-free (string) grammar (MCFG) as the "monadic case" of the MCFTG, which means that all terminal and nonterminal symbols have rank 1, except for a special terminal symbol and the initial nonterminal symbol that have rank 0. We prove (using permutation-freeness) that every tree language $L(G)$ that is generated by an MCFTG G can also be generated by an MCFG, provided that we view every tree as a string in the usual way (Theorem 67). Using this we show that $\mathrm{yd}(L(G))$, which is the set of yields of the trees in $L(G)$, can also be generated by an MCFG G' and, in fact, every MCFG string language is of that form. Since, moreover, the derivation trees of G and G' are related by $\mathrm{LDT}^{\mathrm{R}}$-transducers (in a way similar to $\mathrm{LDT}^{\mathrm{R}}$-equivalence), this result can be used to transform any polynomial time parsing algorithm for MCFGs into a polynomial time parsing algorithm for MCFTGs, as discussed in Section 7.2. In Section 8 we recall the notion of macro tree transducer, and show that the tree translation that computes the value of a derivation tree of an MCFTG G can be realized by a deterministic finite-copying macro tree transducer ($\mathrm{DMT}_{\mathrm{fc}}$-transducer). This implies that $L(G)$ is the image of a regular tree language (viz. the set of derivation trees of G) under a $\mathrm{DMT}_{\mathrm{fc}}$-transduction. Vice versa, every such image can be generated by an MCFTG that can be obtained by a straightforward product construction. From this characterization of the MCFTG tree languages we obtain a number of other characterizations (including those for the MCFG string languages), known from the literature. Thus, they are the tree/string languages generated by context-free graph grammars, they are the tree/string languages generated by 2ACGs, and they are the tree/string languages obtained as images of the regular tree languages under deterministic MSO-definable

tree/tree-to-string transductions (where MSO stands for Monadic Second-Order logic). Section 9 is based on the natural idea that, since every "big nonterminal" of an MCFTG generates a forest, i.e., a sequence of trees, we can also use an MCFTG to generate a set of pairs of trees (i.e., a tree translation) and hence, taking yields, to realize a string translation. We study the resulting translation device in Section 9 and call it an MCFT-transducer. It generalizes the (binary) rational tree translation of [79] (called synchronous forest substitution grammar in [69]) and the synchronous context-free tree grammar of [73]. We prove two results similar to those in [73]. The first result characterizes the MCFT-transductions in terms of macro tree transducers, generalizing the characterization of the MCFTG tree languages of Section 8. We show that the MCFT-transductions are the bimorphisms determined by the $\mathrm{DMT}_{\mathrm{fc}}$-transductions as morphisms (Theorem 81). The second result generalizes the parsing result for MCFTGs in Section 7. It shows that any polynomial time parsing algorithm for MCFGs can be transformed into a polynomial time parsing algorithm for MCFT-transducers (Theorem 82). For an MCFT-transducer M, the algorithm parses a given input string w and translates it into a corresponding output string; more precisely, the algorithm computes all pairs (t_1, t_2) in the transduction of M such that the yield of t_1 is w. Finally, in Section 10, we consider two generalizations of the MCFTG for which the basic semantic definitions are essentially still valid. In both cases the generalized MCFTG is able to generate an unbounded number of copies of a subtree, by allowing several occurrences of the same nonterminal (in the first case) or the same variable (in the second case) to appear in the right-hand side of a rule. Consequently, the resulting tree languages need not be semi-linear anymore. The first generalization is the *parallel* MCFTG (or PMCFTG), which is the obvious generalization of the well-known parallel MCFG of [87]. Roughly speaking, in a parallel MCFTG (or parallel MCFG), whenever two occurrences of the same nonterminal are introduced in a derivation step, these occurrences must be rewritten in exactly the same way in the remainder of the derivation. We did not study the lexicalization of PMCFTGs, but for all the other results on MCFTGs there are analogous results for PMCFTGs with almost the same proofs. The second generalization, which we briefly consider, is the *general* (P)MCFTG, for which we drop the restriction that the rules must be linear (in the variables). Thus a general (P)MCFTG can copy subtrees during one derivation step. General MCFTGs are discussed in [8]. The general MCFTGs of multiplicity 1 are the classical IO context-free tree grammars. The synchronized-context-free tree languages of [7] (which are defined by logic programs) lie between the MCFTG tree languages and the general PMCFTG tree languages. The general PMCFTG tree languages can be characterized as the images of the regular tree languages under arbitrary deterministic macro tree transductions, but otherwise we have no results for general (P)MCFTGs.

As observed above, part of the results in this contribution were first presented in [27], [70], and [25].

2. Preliminaries

We denote the set $\{1, 2, 3, \dots\}$ of positive integers by $\mathbb{N}$ and the set of nonnegative integers by $\mathbb{N}_0 = \mathbb{N} \cup \{0\}$. For every $n \in \mathbb{N}_0$, we let $[n] = \{i \in \mathbb{N} \mid i \leq n\}$. For a set A, we denote its cardinality by $|A|$. A partition of A is a set Π of subsets of A such that each element of A is contained in exactly one element of Π; we allow the empty set $\emptyset$ to be an element of Π. For two functions $f\colon A \to B$ and $g\colon B \to C$ (where A, B, and C are sets), the composition $g \circ f\colon A \to C$ of f and g is defined as usual by $(g \circ f)(a) = g(f(a))$ for every $a \in A$.

2.1. Sequences and strings

Let A be a (not necessarily finite) set. When we view A as a set of basic (i.e., indecomposable) elements, we call A an alphabet and each of its elements a symbol. Note that we do not require alphabets to be finite; finiteness will be explicitly mentioned.[2] For every $n \in \mathbb{N}_0$, we denote by A^n the n-fold Cartesian product of A containing sequences over A; i.e., $A^n = \{(a_1, \dots, a_n) \mid a_1, \dots, a_n \in A\}$ and $A^0 = \{(\,)\}$ contains only the empty sequence $(\,)$, which we also denote by ε. Moreover, we let $A^+ = \bigcup_{n \in \mathbb{N}} A^n$ and $A^* = \bigcup_{n \in \mathbb{N}_0} A^n$. When A is viewed as an alphabet, the sequences in A^* are also called strings. Let $w = (a_1, \dots, a_n)$ be a sequence (or string). Its length n is denoted by $|w|$. For $i \in [n]$, the i-th element of w is a_i. The elements of w are said to occur in w. The set $\{a_1, \dots, a_n\}$ of elements of w will

[2] Infinite alphabets are sometimes convenient. For instance, it is natural to view the infinite set $\{x_1, x_2, \dots\}$ of variables occurring in trees as an alphabet, see Section 2.3. We will use grammars with infinite alphabets as a technical tool in Section 3.3 to define the derivations of usual grammars, which of course have finite alphabets.

be denoted by $\mathrm{occ}(w)$. The sequence w is *repetition-free* if no element of A occurs more than once in w; i.e., $|\mathrm{occ}(w)| = n$. A *permutation* of w is a sequence $(a_{i_1}, \ldots, a_{i_n})$ of the same length such that $\{i_1, \ldots, i_n\} = [n]$. Given another sequence $v = (a'_1, \ldots, a'_m)$ the concatenation $w \cdot v$, also written just wv, is simply $(a_1, \ldots, a_n, a'_1, \ldots, a'_m)$. Moreover, for every $n \in \mathbb{N}_0$, the n-fold concatenation of w with itself is denoted by w^n, in particular $w^0 = \varepsilon$. As usual, we identify the sequence (a) of length 1 with the element $a \in A$ it contains, so $A = A^1 \subseteq A^+$. Consequently, we often write the sequence $(a_1, \ldots, a_n)$ as $a_1 \cdots a_n$. However, if the $a_1, \ldots, a_n$ are themselves sequences, then $a_1 \cdots a_n$ will always denote their concatenation and never the sequence $(a_1, \ldots, a_n)$ of sequences.

Notation. In the following we will often denote sequences over a set A by the same letters as the elements of A. For instance, we will write $a = (a_1, \ldots, a_n)$ with $a \in A^+$ and $a_i \in A$ for all $i \in [n]$. It should hopefully always be clear whether a sequence over A or an element of A is meant. We will consider sequences over several different types of sets, and it would be awkward to use different letters, fonts, or decorations (like $\overline{a}$ and $\vec{a}$) for all of them.

Homomorphisms. Let A and B be sets. A (string) homomorphism from A to B is a mapping $h\colon A \to B^*$. It determines a mapping $h^*\colon A^* \to B^*$ which is also called a (string) homomorphism and which is defined inductively as follows for $w \in A^*$:

$$h^*(w) = \begin{cases} \varepsilon & \text{if } w = \varepsilon \\ h(a) \cdot h^*(v) & \text{if } w = av \text{ with } a \in A \text{ and } v \in A^*. \end{cases}$$

We note that h^* and h coincide on A and that $h^*(wv) = h^*(w) \cdot h^*(v)$ for all $w, v \in A^*$. In certain particular cases, which will be explicitly mentioned, we will denote h^* simply by h, for readability.[3] A homomorphism *over* A is a homomorphism from A to itself. We will often use the following homomorphism from A to B, in the special case where $B \subseteq A$. For a string w over A, the *yield* of w with respect to B, denoted $\mathrm{yd}_B(w)$, is the string over B that is obtained from w by erasing all symbols not in B. Formally, yd_B is the homomorphism from A to B such that $\mathrm{yd}_B(a) = a$ if $a \in B$ and $\mathrm{yd}_B(a) = \varepsilon$ otherwise, and we define $\mathrm{yd}_B(w) = \mathrm{yd}_B^*(w)$. Thus,

$$\mathrm{yd}_B(w) = \begin{cases} \varepsilon & \text{if } w = \varepsilon \\ a\,\mathrm{yd}_B(v) & \text{if } w = av \text{ with } a \in B \text{ and } v \in A^* \\ \mathrm{yd}_B(v) & \text{if } w = av \text{ with } a \in A \setminus B \text{ and } v \in A^*. \end{cases}$$

Note that yd_A is the identity on A^*.

Context-free grammars. We assume that the reader is familiar with context-free grammars [3], which are presented here as systems $G = (N, \Sigma, S, R)$ containing a finite alphabet N of nonterminals, a finite alphabet Σ of terminals that is disjoint to N, an initial nonterminal $S \in N$, and a finite set R of rules of the form $A \to w$ with a nonterminal $A \in N$ and a string $w \in (N \cup \Sigma)^*$. Each nonterminal A generates a language $L(G, A)$, which is given by $L(G, A) = \{w \in \Sigma^* \mid A \Rightarrow_G^* w\}$ using the reflexive, transitive closure $\Rightarrow_G^*$ of the usual rewriting relation $\Rightarrow_G = \{(uAv, uwv) \mid u, v \in (N \cup \Sigma)^*,\ A \to w \in R\}$ of the context-free grammar G. The language generated by G is $L(G) = L(G, S)$. The nonterminals $A, A' \in N$ are *aliases* if $\{w \mid A \to w \in R\} = \{w \mid A' \to w \in R\}$, which yields that $L(G, A) = L(G, A')$. It is well known that for every context-free grammar $G = (N, \Sigma, S, R)$ there is an equivalent one $G' = (N', \Sigma, S_1, R')$ such that w does not contain any nonterminal more than once for every rule $A \to w \in R'$. This can be achieved by introducing sufficiently many aliases as follows. Let m be the maximal number of occurrences of a nonterminal in the right-hand side of a rule in R. We replace each nonterminal A by new nonterminals $A_1, \ldots, A_m$ with initial nonterminal S_1. In addition, we replace each rule $A \to w$ by all the rules $A_i \to w'$, where $i \in [m]$ and w' is obtained from w by replacing the j-th occurrence of each nonterminal B in w by B_j. Thus, $A_1, \ldots, A_m$ are aliases. As an example, the grammar G with rules $S \to \sigma SS$ and $S \to a$ is transformed into the grammar G' with rules $S_1 \to \sigma S_1 S_2$, $S_2 \to \sigma S_1 S_2$, $S_1 \to a$, and $S_2 \to a$. It should be clear that $L(G') = L(G)$, and in fact, the derivation trees of G and G' are closely related (by simply introducing appropriate subscripts in the derivation trees of G or removing the introduced subscripts from the derivation trees of G').

[3] There will be four such cases only: yield functions 'yd' (see the remainder of this paragraph), rank functions 'rk' (see the first paragraph of Section 2.2), injections 'in' (see the first paragraph of Section 2.3), and tree homomorphisms $\hat{h}$ (see the third paragraph of Section 2.3).

2.2. Trees and forests

A *ranked set*, or *ranked alphabet*, is a pair $(\Sigma, \mathrm{rk}_\Sigma)$, where Σ is a (possibly infinite) set and $\mathrm{rk}_\Sigma \colon \Sigma \to \mathbb{N}_0$ is a mapping that associates a rank to every element of Σ. In what follows the elements of Σ will be called symbols. For all $k \in \mathbb{N}_0$, we let $\Sigma^{(k)} = \{\sigma \in \Sigma \mid \mathrm{rk}_\Sigma(\sigma) = k\}$ be the set of all symbols of rank k. We sometimes indicate the rank k of a symbol $\sigma \in \Sigma$ explicitly, as in $\sigma^{(k)}$. Moreover, as usual, we just write Σ for the ranked alphabet $(\Sigma, \mathrm{rk}_\Sigma)$, and whenever Σ is clear from the context, we write 'rk' instead of 'rk_Σ'. If Σ is finite, then we denote by mrk_Σ the maximal rank of the symbols in Σ; i.e., $mrk_\Sigma = \max\{\mathrm{rk}(\sigma) \mid \sigma \in \Sigma\}$. The mapping rk^* from Σ^* to $\mathbb{N}_0^*$, as defined in the paragraph on homomorphisms in Section 2.1, will also be denoted by 'rk'. It associates a multiple rank (i.e., a sequence of ranks) to every sequence of elements of Σ. The union of ranked alphabets $(\Sigma, \mathrm{rk}_\Sigma)$ and $(\Delta, \mathrm{rk}_\Delta)$ is $(\Sigma \cup \Delta, \mathrm{rk}_\Sigma \cup \mathrm{rk}_\Delta)$; it is again a ranked alphabet provided that the same rank $\mathrm{rk}_\Sigma(\gamma) = \mathrm{rk}_\Delta(\gamma)$ is assigned to all symbols $\gamma \in \Sigma \cap \Delta$.

We build trees over the ranked alphabet Σ such that the nodes are labeled by elements of Σ and the rank of the node label determines the number of its children. Formally we define trees as nonempty strings over Σ as follows. The set T_Σ of *trees over* Σ is the smallest set $T \subseteq \Sigma^+$ such that $\sigma t_1 \cdots t_k \in T$ for all $k \in \mathbb{N}_0$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots, t_k \in T$. As usual, we will also denote the string $\sigma t_1 \cdots t_k$ by the term $\sigma(t_1, \ldots, t_k)$. If we know that $t \in T_\Sigma$ and $t = \sigma(t_1, \ldots, t_k)$, then it is clear that $k \in \mathbb{N}_0$, $\sigma \in \Sigma^{(k)}$, and $t_1, \ldots, t_k \in T_\Sigma$, so unless we need stronger assumptions, we will often omit the quantifications of k, σ, and $t_1, \ldots, t_k$. It is well known that if $\sigma w \in T_\Sigma$ with $k \in \mathbb{N}_0$, $\sigma \in \Sigma^{(k)}$, and $w \in \Sigma^*$, then there are unique trees $t_1, \ldots, t_k \in T_\Sigma$ such that $w = t_1 \cdots t_k$. Any subset of T_Σ is called a tree language over Σ. A detailed treatment of trees and tree languages is presented in [41] (see also [16, 42]).

Trees can be viewed as node-labeled graphs in a well-known way. As usual, we use Dewey notation to address the nodes of a tree; these addresses will be called positions. Formally, a *position* is an element of $\mathbb{N}^*$. Thus, it is a sequence of positive integers, which, intuitively, indicates successively in which subtree the addressed node can be found. More precisely, the root is at position ε, and the position pi with $p \in \mathbb{N}^*$ and $i \in \mathbb{N}$ refers to the i-th child of the node at position p. The set $\mathrm{pos}(t) \subseteq \mathbb{N}^*$ of positions of a tree $t \in T_\Sigma$ with $t = \sigma(t_1, \ldots, t_k)$ is defined inductively by $\mathrm{pos}(t) = \{\varepsilon\} \cup \{ip \mid i \in [k], p \in \mathrm{pos}(t_i)\}$. The tree t associates a label to each of its positions, so it induces a mapping $t \colon \mathrm{pos}(t) \to \Sigma$ such that $t(p)$ is the *label* of t at position p. Formally, if $t = \sigma(t_1, \ldots, t_k)$, then $t(\varepsilon) = \sigma$ and $t(ip) = t_i(p)$. For nodes $p, p' \in \mathrm{pos}(t)$, we say as usual that p' is an *ancestor* of p if p' is a prefix of p; i.e., there exists $w \in \mathbb{N}^*$ such that $p = p'w$. A *leaf* of t is a position $p \in \mathrm{pos}(t)$ with $t(p) \in \Sigma^{(0)}$. The *yield* of t, denoted by $\mathrm{yd}(t)$, is the sequence of labels of its leaves, read from left to right. However, as usual, we assume the existence of a special symbol e of rank 0 that represents the empty string and is omitted from $\mathrm{yd}(t)$. Formally $\mathrm{yd}(t) = \mathrm{yd}_{\Sigma^{(0)} \setminus \{e\}}(t)$, where yd_B is defined in the paragraph on homomorphisms in Section 2.1.

A *forest* is a sequence of trees; i.e., an element of T_Σ^*. Note that every tree of T_Σ is a forest of length 1. A forest can be viewed as a node-labeled graph in a natural way, for instance by connecting the roots of its trees by "invisible" #-labeled directed edges, in the given order. This leads to the following obvious extension of Dewey notation to address the nodes of a forest. Formally, from now on, a *position* is an element of the set $\{\#^n p \mid n \in \mathbb{N}_0, p \in \mathbb{N}^*\} \subseteq (\mathbb{N} \cup \{\#\})^*$, where # is a special symbol not in $\mathbb{N}$. Intuitively, the root of the j-th tree of a forest is at position $\#^{j-1}$ and, as before, the position pi refers to the i-th child of the node at position p. For each forest $t = (t_1, \ldots, t_m)$ with $m \in \mathbb{N}_0$ and $t_1, \ldots, t_m \in T_\Sigma$, the set $\mathrm{pos}(t)$ of positions of t is defined by $\mathrm{pos}(t) = \bigcup_{j=1}^m \{\#^{j-1}p \mid p \in \mathrm{pos}(t_j)\}$. Moreover, for every $j \in [m]$ and $p \in \mathrm{pos}(t_j)$, we let $t(\#^{j-1}p) = t_j(p)$ be the label of t at position $\#^{j-1}p$.[4] Let $\Omega \subseteq \Sigma$ be a selection of symbols. For every $t \in T_\Sigma^*$, we let $\mathrm{pos}_\Omega(t) = \{p \in \mathrm{pos}(t) \mid t(p) \in \Omega\}$ be the set of all Ω-labeled positions of t. For every $\sigma \in \Sigma$, we simply write $\mathrm{pos}_\sigma(t)$ instead of $\mathrm{pos}_{\{\sigma\}}(t)$, and we say that σ *occurs in* t if $\mathrm{pos}_\sigma(t) \neq \emptyset$. The set of symbols in Ω that occur in t is denoted by $\mathrm{occ}_\Omega(t)$; i.e., $\mathrm{occ}_\Omega(t) = \{t(p) \mid p \in \mathrm{pos}_\Omega(t)\}$.[5] The forest t is *uniquely Ω-labeled* if no symbol in Ω occurs more than once in t; i.e., $|\mathrm{pos}_\omega(t)| \leq 1$ for every $\omega \in \Omega$. It is well known, and can easily be proved by induction on the structure of t, that $|\mathrm{pos}(t)| + m \leq 2 \cdot |\mathrm{pos}_{\Sigma^{(0)}}(t)| + |\mathrm{pos}_{\Sigma^{(1)}}(t)|$ for every forest $t \in T_\Sigma^*$ of length m.

Regular tree grammars. A *regular tree grammar* (in short, RTG) over Σ is a context-free grammar $G = (N, \Sigma, S, R)$ such that N is a ranked alphabet with $\mathrm{rk}(A) = 0$ for every $A \in N$, Σ is a ranked alphabet, and w is a tree in $T_{N \cup \Sigma}$ for every rule $A \to w$ in R. Throughout this contribution we assume

[4] These definitions are consistent with those given in the previous paragraph for trees, which are forests of length 1.

[5] Note that $\mathrm{occ}(t) = \{t_1, \ldots, t_m\}$ by Section 2.1. This will, however, never be used.

that G is in normal form; i.e., that all its rules are of the form $A \to \sigma(A_1, \dots, A_k)$ with $k \in \mathbb{N}_0$, $A, A_1, \dots, A_k \in N$, and $\sigma \in \Sigma^{(k)}$. The language $L(G)$ generated by an RTG G is a *regular tree language*. The class of all regular tree languages is denoted by RT. We assume the reader to be familiar with regular tree grammars [42, Section 6], and also more or less familiar with (linear, nondeleting) context-free tree grammars [42, Section 15], which we formally define in Section 3.

2.3. Substitution

In this subsection we define and discuss first- and second-order substitution of trees and forests. To this end, we use a fixed countably infinite alphabet $X = \{x_1, x_2, \dots\} \cup \{\square\}$ of *variables*, which is disjoint to the ranked alphabet Σ, and for every $k \in \mathbb{N}_0$ we let $X_k = \{x_i \mid i \in [k]\}$ be the first k variables from X. Note that $X_0 = \emptyset$. The use of the special variable $\square$ will be explained in Section 5 (before Lemma 42). For $Z \subseteq X$, the set $T_\Sigma(Z)$ of trees over Σ with variables in Z is defined by $T_\Sigma(Z) = T_{\Sigma \cup Z}$, where every variable $x \in Z$ has rank 0. Thus, the variables can only occur at the leaves. We will be mainly interested in the substitution of patterns. For every $k \in \mathbb{N}_0$, we define the set $P_\Sigma(X_k)$ of k-ary *patterns* to consist of all trees $t \in T_\Sigma(X_k)$ such that each variable of X_k occurs exactly once in t; i.e., $|\mathrm{pos}_x(t)| = 1$ for every $x \in X_k$.[6] Consequently, $P_\Sigma(X_0) = T_\Sigma(X_0) = T_\Sigma$, and for all distinct $i, j \in \mathbb{N}_0$ the sets $P_\Sigma(X_i)$ and $P_\Sigma(X_j)$ are disjoint. This allows us to turn the set $P_\Sigma(X) = \bigcup_{k \in \mathbb{N}_0} P_\Sigma(X_k)$ of all patterns into a ranked set such that $P_\Sigma(X)^{(k)} = P_\Sigma(X_k)$ for every $k \in \mathbb{N}_0$; in other words, for every $t \in P_\Sigma(X)$ let $\mathrm{rk}(t)$ be the unique integer $k \in \mathbb{N}_0$ such that $t \in P_\Sigma(X_k)$.[7] Since 'rk' also denotes rk^* (see the first paragraph of Section 2.2), 'rk' is also a mapping from $P_\Sigma(X)^*$ to $\mathbb{N}_0^*$. There is a natural rank-preserving *injection* $\mathrm{in}\colon \Sigma \to P_\Sigma(X)$ of the alphabet Σ into the set of patterns, which is given by $\mathrm{in}(\sigma) = \sigma(x_1, \dots, x_k)$ for every $k \in \mathbb{N}_0$ and $\sigma \in \Sigma^{(k)}$. Note that $\mathrm{in}(\sigma) = \sigma$ if $k = 0$. The mapping in^* from Σ^* to $P_\Sigma(X)^*$, as defined in Section 2.1, will also be denoted by 'in'. It is a rank-preserving injection that associates a sequence of patterns to every sequence of elements of Σ.

We start with first-order substitution, in which variables are replaced by trees. For a tree $t \in T_\Sigma(X)$, a set $Z \subseteq X$ of variables, and a mapping $f\colon Z \to T_\Sigma(X)$, the *first-order substitution* $t[f]$, also written as $t[z \leftarrow f(z) \mid z \in Z]$, yields the tree in $T_\Sigma(X)$ obtained by replacing in t every occurrence of z by $f(z)$ for every $z \in Z$. Formally, $t[f]$ is defined by induction on the structure of t as follows:

$$t[f] = \begin{cases} f(z) & \text{if } t = z \text{ with } z \in Z \\ \sigma(t_1[f], \dots, t_k[f]) & \text{if } t = \sigma(t_1, \dots, t_k) \text{ with } \sigma \in \Sigma \cup X,\ \sigma \notin Z. \end{cases}$$

We note that $t[f] = h^*(t)$, where h is the string homomorphism over $\Sigma \cup X$ such that $h(\alpha) = f(\alpha)$ if $\alpha \in Z$ and $h(\alpha) = \alpha$ otherwise.

Whereas we replace X-labeled nodes (which are leaves) in first-order substitution, in second-order substitution we replace Σ-labeled nodes (which can also be internal nodes); i.e., nodes with a label in $\Sigma^{(k)}$ for some $k \in \mathbb{N}_0$. Such a node is replaced by a k-ary pattern, in which the variables $x_1, \dots, x_k$ are used as unique placeholders for the k children of the node. In fact, second-order substitutions are just tree homomorphisms. Let Σ and Δ be ranked alphabets. A *(simple) tree homomorphism* from Σ to Δ is a rank-preserving mapping $h\colon \Sigma \to P_\Delta(X)$; i.e., $\mathrm{rk}(h(\sigma)) = \mathrm{rk}(\sigma)$ for every $\sigma \in \Sigma$.[8] It determines a mapping $\hat{h}\colon T_\Sigma(X) \to T_\Delta(X)$, and we will use $\hat{h}$ also to denote the mapping $(\hat{h})^*\colon T_\Sigma(X)^* \to T_\Delta(X)^*$ as defined in the paragraph on homomorphisms in Section 2.1. Roughly speaking, for a tree (or forest) t, the tree (or forest) $\hat{h}(t)$ is obtained from t by replacing, for every $p \in \mathrm{pos}_\sigma(t)$ with label $\sigma \in \Sigma^{(k)}$, the subtree at position p by the pattern $h(\sigma)$, into which the k subtrees at positions $p1, \dots, pk$ are (first-order) substituted for the variables $x_1, \dots, x_k$, respectively. Since $h(\sigma)$ is a pattern, these subtrees can neither be copied nor deleted, but they can be permuted. Thus, the pattern $h(\sigma)$ is "folded" into t at position p. Formally, the mapping $\hat{h}$, which we also call tree homomorphism, is defined inductively as follows for $t \in T_\Sigma(X)$:

$$\hat{h}(t) = \begin{cases} x & \text{if } t = x \text{ with } x \in X \\ h(\sigma)[x_i \leftarrow \hat{h}(t_i) \mid 1 \le i \le k] & \text{if } t = \sigma(t_1, \dots, t_k) \text{ with } \sigma \in \Sigma. \end{cases}$$

[6] Note that the variable $\square$ does not occur in patterns.

[7] Since $P_\Sigma(X) \subseteq (\Sigma \cup X)^*$ by definition, every pattern $t \in P_\Sigma(X)$ also has a multiple rank $\mathrm{rk}_{\Sigma \cup X}(t) \in \mathbb{N}_0^*$. This will, however, never be used. We also observe that we will not consider trees over the ranked set $P_\Sigma(X)$.

[8] Since $h(\sigma)$ is a pattern for every $\sigma \in \Sigma$, the tree homomorphism h is simple; i.e., linear and nondeleting. This is the only type of tree homomorphism considered in this paper (except briefly in the last section).

Clearly, $\hat{h}(t)$ only depends on the values of h for the symbols occurring in t; in other words, if g is another tree homomorphism from Σ to Δ such that $g(\sigma) = h(\sigma)$ for every $\sigma \in \mathrm{occ}_\Sigma(t)$, then $\hat{g}(t) = \hat{h}(t)$. We additionally observe that $\hat{h}(t) = \delta(\hat{h}(t_1), \ldots, \hat{h}(t_k))$ if $t = \sigma(t_1, \ldots, t_k)$ and $h(\sigma) = \mathrm{in}(\delta)$ for some $\delta \in \Delta$. A tree homomorphism h is a *projection* if for every $\sigma \in \Sigma$ there exists $\delta \in \Delta$ such that $h(\sigma) = \mathrm{in}(\delta)$. Thus, a projection is just a relabeling of the nodes of the trees. For a ranked alphabet Σ, a tree homomorphism *over* Σ is a tree homomorphism from Σ to itself.

The following lemma states elementary properties of (simple) tree homomorphisms. They can easily be proved by induction on the structure of trees in $T_\Sigma(X)$ and then extended to forests in $T_\Sigma(X)^*$.

Lemma 1 *Let h be a tree homomorphism from Σ to Δ, and let $t \in T_\Sigma(X)^*$ and $u = \hat{h}(t)$.*
(1) $|\mathrm{pos}_x(u)| = |\mathrm{pos}_x(t)|$ *for every* $x \in X$.
(2) $|\mathrm{pos}_\delta(u)| = \sum_{\sigma\in\Sigma} |\mathrm{pos}_\sigma(t)| \cdot |\mathrm{pos}_\delta(h(\sigma))|$ *for every* $\delta \in \Delta$.

By the first statement of this lemma, tree homomorphisms preserve patterns and their ranks; i.e., $\hat{h}(t) \in P_\Delta(X_k)$ for all $t \in P_\Sigma(X_k)$. Moreover, $\hat{h}(t) \in P_\Delta(X)^*$ and $\mathrm{rk}(\hat{h}(t)) = \mathrm{rk}(t)$ for all $t \in P_\Sigma(X)^*$.

Next, we recall two other easy properties of tree homomorphisms. Namely, they distribute over first-order substitution, and they are closed under composition (see [4, Corollary 8(5)]).

Lemma 2 *Let h be a tree homomorphism from Σ to Δ, let $t \in T_\Sigma(X)$, and let $f\colon Z \to T_\Sigma(X)$ for some $Z \subseteq X$. Then $\hat{h}(t[f]) = \hat{h}(t)[\hat{h} \circ f]$.*

Lemma 3 *Let h_1 and h_2 be tree homomorphisms from Σ to Ω and from Ω to Δ, respectively, and let $h = \hat{h}_2 \circ h_1$, which is a tree homomorphism from Σ to Δ. Then $\hat{h} = \hat{h}_2 \circ \hat{h}_1$.*

These lemmas have straightforward proofs. Lemma 2 can be proved by induction on the structure of t, and then Lemma 3 can be proved by showing that $\hat{h}(t) = \hat{h}_2(\hat{h}_1(t))$, again by induction on the structure of t, using Lemma 2 in the induction step.

In the remainder of this subsection we consider tree homomorphisms over Σ. Let t be a forest in $T_\Sigma(X)^*$ and let $\sigma = (\sigma_1, \ldots, \sigma_n) \in \Sigma^n$ with $n \in \mathbb{N}_0$ be a repetition-free sequence of symbols in Σ. Moreover, let $u = (u_1, \ldots, u_n)$ be a forest in $P_\Sigma(X)^n$ such that $\mathrm{rk}(u) = \mathrm{rk}(\sigma)$.[9] The *second-order substitution* $t[\sigma \leftarrow u]$ yields the forest $\hat{h}(t) \in T_\Sigma(X)^*$, where h is the tree homomorphism over Σ corresponding to $[\sigma \leftarrow u]$, which is defined by $h(\sigma_i) = u_i$ for $i \in [n]$ and $h(\tau) = \mathrm{in}(\tau)$ for $\tau \in \Sigma \setminus \{\sigma_1, \ldots, \sigma_n\}$. If $t \in P_\Sigma(X)^*$, then $t[\sigma \leftarrow u] \in P_\Sigma(X)^*$ and $\mathrm{rk}(t[\sigma \leftarrow u]) = \mathrm{rk}(t)$ by Lemma 1(1). Obviously, the order of the symbols and trees in σ and u is irrelevant: if $\sigma' = (\sigma_{i_1}, \ldots, \sigma_{i_n})$ and $u' = (u_{i_1}, \ldots, u_{i_n})$, where $(i_1, \ldots, i_n)$ is a permutation of $(1, \ldots, n)$, then $t[\sigma' \leftarrow u'] = t[\sigma \leftarrow u]$. Thus, the use of sequences is just a way of associating each symbol σ_i with its replacing tree u_i. Clearly, $t[\sigma \leftarrow u] = t$ if no symbol of σ occurs in t; i.e., if $\mathrm{occ}_\Sigma(t) \cap \mathrm{occ}(\sigma) = \emptyset$. We also note that $t[\sigma \leftarrow \mathrm{in}(\sigma)] = t$ and $\mathrm{in}(\sigma)[\sigma \leftarrow u] = u$. Finally $t[\sigma \leftarrow u] = t_1[\sigma \leftarrow u] \cdot t_2[\sigma \leftarrow u]$ if $t = t_1 t_2$ for forests t_1 and t_2.

In the next lemma, we state some additional elementary properties of second-order substitution.

Lemma 4 *Let $t \in T_\Sigma(X)^*$ be a forest and $\sigma_1, \sigma_2 \in \Sigma^*$ be repetition-free sequences of symbols. Moreover, let $u_1, u_2 \in P_\Sigma(X)^*$ be forests of patterns such that $\mathrm{rk}(u_1) = \mathrm{rk}(\sigma_1)$ and $\mathrm{rk}(u_2) = \mathrm{rk}(\sigma_2)$.*
(1) *If $\mathrm{occ}(\sigma_1) \cap \mathrm{occ}(\sigma_2) = \emptyset$ (i.e., $\sigma_1\sigma_2$ is repetition-free), then*
$$t[\sigma_1 \leftarrow u_1][\sigma_2 \leftarrow u_2] = t[\sigma_1\sigma_2 \leftarrow u_1[\sigma_2 \leftarrow u_2] \cdot u_2].$$
(2) *If $\mathrm{occ}(\sigma_1) \cap \mathrm{occ}(\sigma_2) = \emptyset$ and $\mathrm{occ}_\Sigma(u_1) \cap \mathrm{occ}(\sigma_2) = \emptyset$, then*
$$t[\sigma_1 \leftarrow u_1][\sigma_2 \leftarrow u_2] = t[\sigma_1\sigma_2 \leftarrow u_1 u_2].$$
(3) *If $\mathrm{occ}(\sigma_1) \cap \mathrm{occ}(\sigma_2) = \emptyset$ and $\mathrm{occ}_\Sigma(u_2) \cap \mathrm{occ}(\sigma_1) = \emptyset$, then*
$$t[\sigma_1 \leftarrow u_1][\sigma_2 \leftarrow u_2] = t[\sigma_2 \leftarrow u_2][\sigma_1 \leftarrow u_1[\sigma_2 \leftarrow u_2]].$$
(4) *If $\mathrm{occ}_\Sigma(t) \cap \mathrm{occ}(\sigma_2) \subseteq \mathrm{occ}(\sigma_1)$, then*
$$t[\sigma_1 \leftarrow u_1][\sigma_2 \leftarrow u_2] = t[\sigma_1 \leftarrow u_1[\sigma_2 \leftarrow u_2]].$$

Proof Let h_1 and h_2 be the tree homomorphisms over Σ that correspond to $[\sigma_1 \leftarrow u_1]$ and $[\sigma_2 \leftarrow u_2]$, as defined above. Moreover, let h be the tree homomorphism that corresponds to $[\sigma_1\sigma_2 \leftarrow u_1[\sigma_2 \leftarrow u_2] \cdot u_2]$.

[9] Recall that this means that $u_i \in P_\Sigma(X_{\mathrm{rk}(\sigma_i)})$ for every $i \in [n]$.

Provided that $\sigma_1\sigma_2$ is repetition-free, it is easy to check that $h = \hat{h}_2 \circ h_1$, and hence $\hat{h} = \hat{h}_2 \circ \hat{h}_1$ by Lemma 3. This shows the first equality. If additionally no symbol of σ_2 occurs in u_1, then $u_1[\sigma_2 \leftarrow u_2] = u_1$, which shows the second equality. The third equality is a direct consequence of the first two because $t[\sigma_1\sigma_2 \leftarrow u_1[\sigma_2 \leftarrow u_2] \cdot u_2] = t[\sigma_2\sigma_1 \leftarrow u_2 \cdot u_1[\sigma_2 \leftarrow u_2]]$. To prove the fourth equality, let g be the tree homomorphism that corresponds to $[\sigma_1 \leftarrow u_1[\sigma_2 \leftarrow u_2]]$. By Lemma 3, it now suffices to show that $\hat{h}_2(h_1(\sigma)) = g(\sigma)$ for every $\sigma \in \mathrm{occ}_\Sigma(t)$. This is obvious for $\sigma \in \mathrm{occ}(\sigma_1)$. If $\sigma \in \mathrm{occ}_\Sigma(t) \setminus \mathrm{occ}(\sigma_1)$ then, by assumption, $\sigma \notin \mathrm{occ}(\sigma_2)$, and so both sides of the equation are equal to $\mathrm{in}(\sigma)$. ■

In particular, Lemma 4(3) implies that $t[\sigma_1 \leftarrow u_1][\sigma_2 \leftarrow u_2] = t[\sigma_2 \leftarrow u_2][\sigma_1 \leftarrow u_1]$ provided that $\mathrm{occ}(\sigma_1) \cap \mathrm{occ}(\sigma_2) = \emptyset$, $\mathrm{occ}_\Sigma(u_2) \cap \mathrm{occ}(\sigma_1) = \emptyset$, and $\mathrm{occ}_\Sigma(u_1) \cap \mathrm{occ}(\sigma_2) = \emptyset$. This is called the confluence or commutativity of substitution in [11]. Similarly, Lemma 4(4) is called the associativity of substitution in [11]. As shown in the proof above, these two properties of substitution are essentially special cases of the composition of tree homomorphisms as characterized in Lemma 3.

Above, we have defined the substitution of a forest (of patterns) for a repetition-free sequence over Σ. In the next section we also need to simultaneously substitute several forests for several such sequences. That leads to the following formal definitions, which may now seem rather superfluous. Let $\mathcal{L} = \{\sigma_1, \ldots, \sigma_k\}$ be a finite subset of Σ^* such that $\sigma_1 \cdots \sigma_k$ is repetition-free, where $\sigma_1 \cdots \sigma_k = \varepsilon$ if $k = 0$. A (second-order) *substitution function* for $\mathcal{L}$ is a mapping $f\colon \mathcal{L} \to P_\Sigma(X)^*$ such that $\mathrm{rk}(f(\sigma)) = \mathrm{rk}(\sigma)$ for every $\sigma \in \mathcal{L}$. For a forest $t \in P_\Sigma(X)^*$, the *simultaneous second-order substitution* $t[f]$, also written as $t[\sigma \leftarrow f(\sigma) \mid \sigma \in \mathcal{L}]$, yields $t[f] = t[\sigma_1 \cdots \sigma_k \leftarrow f(\sigma_1) \cdots f(\sigma_k)]$. Clearly, $t[f]$ does not depend on the given order of the elements in $\mathcal{L}$. In the special case $\mathcal{L} \subseteq \Sigma$ we obtain a notion of second-order substitution that does not involve sequences, with $f\colon \mathcal{L} \to P_\Sigma(X)$. In that case we have $t[f] = t[(\sigma_1, \ldots, \sigma_k) \leftarrow (f(\sigma_1), \ldots, f(\sigma_k))]$.

3. Multiple context-free tree grammars

In this section we introduce the main formalism discussed in this contribution: the multiple context-free tree grammars. In the first subsection we define their syntax and least fixed point semantics and in the second and third subsection we discuss two alternative semantics, namely their derivation trees and their derivations, respectively. In the second subsection we also define the notion of $\mathrm{LDT}^{\mathrm{R}}$-equivalence of multiple context-free tree grammars, which formalizes grammatical similarity.

3.1. Syntax and least fixed point semantics

We start with the syntax of multiple context-free tree grammars, which we explain after the formal definition. The definition of their semantics follows after that explanation. Then we give two examples.

Definition 5 A *multiple context-free tree grammar* (in short, MCFTG) is a system $G = (N, \mathcal{N}, \Sigma, S, R)$ such that

- N is a finite ranked alphabet of *nonterminals*,
- $\mathcal{N} \subseteq N^+$ is a finite set of *big nonterminals*, which are nonempty repetition-free sequences of nonterminals, such that $\mathrm{occ}(A) \neq \mathrm{occ}(A')$ for all distinct $A, A' \in \mathcal{N}$,
- Σ is a finite ranked alphabet of *terminals* such that $\Sigma \cap N = \emptyset$ and $mrk_\Sigma \geq 1$,[10]
- $S \in \mathcal{N} \cap N^{(0)}$ is the *initial (big) nonterminal* (of length 1 and rank 0), and
- R is a finite set of *rules* of the form $A \to (u, \mathcal{L})$, where $A \in \mathcal{N}$ is a big nonterminal, $u \in P_{N\cup\Sigma}(X)^+$ is a uniquely N-labeled forest (of patterns) such that $\mathrm{rk}(u) = \mathrm{rk}(A)$, and $\mathcal{L} \subseteq \mathcal{N}$ is a set of big nonterminals such that $\{\mathrm{occ}(B) \mid B \in \mathcal{L}\}$ is a partition of $\mathrm{occ}_N(u)$.[11] □

For a given rule $\rho = A \to (u, \mathcal{L})$, the big nonterminal A, denoted by $\mathrm{lhs}(\rho)$, is called the left-hand side of ρ, the forest u, denoted by $\mathrm{rhs}(\rho)$, is called the right-hand side of ρ, and the big nonterminals of $\mathcal{L}$, denoted by $\mathcal{L}(\rho)$, are called the *links* of ρ.

The *multiplicity* (or *fan-out*) of the MCFTG G, which is denoted by $\mu(G)$, is the maximal length of its big nonterminals. The *width* of G, which is denoted by $\theta(G)$, is the maximal rank of its nonterminals. And the *rule-width* (or *rank*) of G, which is denoted by $\lambda(G)$, is the maximal number of links of its rules. Thus $\mu(G) = \max\{|A| \mid A \in \mathcal{N}\}$, $\theta(G) = mrk_N = \max\{\mathrm{rk}(A) \mid A \in N\}$, and $\lambda(G) = \max\{|\mathcal{L}(\rho)| \mid \rho \in R\}$.

[10] To avoid trivialities, we do not consider the case where all symbols of Σ have rank 0.

[11] Thus, $\mathrm{occ}_N(u) = \bigcup_{B\in\mathcal{L}} \mathrm{occ}(B)$ and $\mathrm{occ}(B) \cap \mathrm{occ}(B') = \emptyset$ for all distinct $B, B' \in \mathcal{L}$.

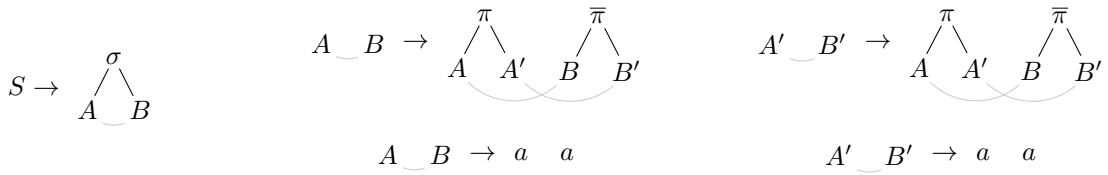


Figure 1: Rules of the MRTG G of Example 6.

Next, we define two syntactic restrictions. An MCFTG G is a *multiple regular tree grammar* (in short, MRTG) if $\theta(G) = 0$, and it is a *(simple) context-free tree grammar* (in short, spCFTG) if $\mu(G) = 1$; i.e., $\mathcal{N} \subseteq N$. In an MRTG all nonterminals thus have rank 0, and in an spCFTG all big nonterminals are nonterminals since their length is exactly 1. Consequently, in an spCFTG we may simply assume that $\mathcal{N} = N$, and thus there is no need to specify $\mathcal{N}$ for it. In the literature, a rule $A \to (u, \mathcal{L})$ of an spCFTG is usually written as $\mathrm{in}(A) \to u$, in which $\mathrm{in}(A) = A(x_1, \ldots, x_{\mathrm{rk}(A)})$ and $\mathcal{L}$ can be omitted because it must be equal to $\mathrm{occ}_N(u)$. Since the right-hand side u of this rule is a pattern, our context-free tree grammars are simple; i.e., linear and nondeleting.

Let us discuss the requirements on the components of G in more detail. Each big nonterminal is a nonempty repetition-free sequence $A = (A_1, \ldots, A_n)$ of nonterminals from N. Repetition-freeness of A requires that all these nonterminals A_i are distinct (cf. Section 2.1). The requirement that 'occ' is injective on $\mathcal{N}$ (i.e., that $\mathrm{occ}(A) \neq \mathrm{occ}(A')$ for all distinct $A, A' \in \mathcal{N}$) means that $\mathcal{N}$ can be viewed as consisting of sets of nonterminals, where each set is equipped with a fixed linear order (viz. the set $\mathrm{occ}(A) = \{A_1, \ldots, A_n\}$ with the order $\sqsubseteq$ such that $A_1 \sqsubset \cdots \sqsubset A_n$). Moreover, since the alphabet N is ranked, every big nonterminal A has a (multiple) rank $\mathrm{rk}(A) = (\mathrm{rk}(A_1), \ldots, \mathrm{rk}(A_n)) \in \mathbb{N}_0^n$ (cf. Section 2.2), and similarly, every forest $u = (u_1, \ldots, u_n)$ with $u_1, \ldots, u_n \in P_{N\cup\Sigma}(X)$ has a (multiple) rank $\mathrm{rk}(u) = (\mathrm{rk}(u_1), \ldots, \mathrm{rk}(u_n)) \in \mathbb{N}_0^n$ (cf. Section 2.3). Thus, a rule $A \to (u, \mathcal{L})$ of G is of the form $(A_1, \ldots, A_n) \to ((u_1, \ldots, u_n), \mathcal{L})$ where $n \in \mathbb{N}_0$, $A_i \in N$ and $u_i \in P_{N\cup\Sigma}(X_{\mathrm{rk}(A_i)})$ for every $i \in [n]$, and $\mathcal{L} \subseteq \mathcal{N}$. The use of sequences is irrelevant; it is just a way of associating each $A_i \in \mathrm{occ}(A)$ with the corresponding pattern u_i, thus facilitating the formal description of the syntax and semantics of G. Additionally, in the above rule, u is uniquely N-labeled, which means that also in u no nonterminal occurs more than once (cf. Section 2.2). This requirement, which is not essential but technically convenient, is similar to the restriction discussed for context-free grammars at the end of Section 2.1. Moreover, the set $\{\mathrm{occ}(B) \mid B \in \mathcal{L}\}$ forms a partition of $\mathrm{occ}_N(u)$. Since each big nonterminal B is repetition-free, 'occ' is injective on $\mathcal{N}$, and u is uniquely N-labeled, we obtain that each big nonterminal from $\mathcal{L}$ occurs "spread-out" exactly once in u and no other nonterminals occur in u. More precisely, for each big nonterminal $B = (C_1, \ldots, C_m) \in \mathcal{L}$ with $C_1, \ldots, C_m \in N$, there is a unique repetition-free sequence $p_B = (p_1, \ldots, p_m) \in \mathrm{pos}_N(u)^m$ of positions such that $(u(p_1), \ldots, u(p_m)) = (C_1, \ldots, C_m)$, and we have that $\mathrm{occ}(p_B) \cap \mathrm{occ}(p_{B'}) = \emptyset$ for every other $B' \in \mathcal{L}$ and $\mathrm{pos}_N(u) = \bigcup_{B\in\mathcal{L}} \mathrm{occ}(p_B)$. Note that if $\mathcal{L} = \{B_1, \ldots, B_k\}$ with $B_1, \ldots, B_k \in \mathcal{N}$, then the concatenation $B_1 \cdots B_k \in N^*$ of the elements of $\mathcal{L}$ is repetition-free and $\mathrm{occ}(B_1 \cdots B_k) = \mathrm{occ}_N(u)$.

Intuitively, the application of the above rule $\rho = A \to (u, \mathcal{L})$ consists of the simultaneous application of the n spCFTG rules $A_i(x_1, \ldots, x_{\mathrm{rk}(A_i)}) \to u_i$ to an occurrence of the "spread-out" big nonterminal $A = (A_1, \ldots, A_n)$ and the introduction of (occurrences of) the new "spread-out" big nonterminals from $\mathcal{L}$. Every big nonterminal $B = (C_1, \ldots, C_m) \in \mathcal{L}$, as above, can be viewed as a link between the positions $p_1, \ldots, p_m$ of u with labels $C_1, \ldots, C_m$ as well as a link between the corresponding positions after the application of ρ (see Figure 1). The rule ρ can only be applied to positions with labels $A_1, \ldots, A_n$ that are joined by such a link. Thus, rule applications are "local" in the sense that a rule can rewrite only nonterminals that were previously introduced together in a single step of the derivation, just as for the local unordered scattered context grammar of [78], which is equivalent to the multiple context-free (string) grammar. However, since it is technically a bit problematic to define such derivation steps between trees in $T_{N\cup\Sigma}$ that are not necessarily uniquely N-labeled (because it additionally requires to keep track of each link as a sequence of positions rather than as a big nonterminal), we prefer to define the language generated by the MCFTG G through a least fixed point semantics similar to that of multiple context-free (string) grammars in [87]. As will be discussed in Section 3.2, this is closely related to a semantics in terms of derivation trees, similar to that of (string-based) linear context-free rewriting systems in [92]. The derivations of an MCFTG will be considered in Section 3.3.

In an spCFTG, a nonterminal A of rank k can be viewed as a generator of trees in $P_\Sigma(X_k)$ using

derivations that start with $A(x_1, \ldots, x_k)$. In the same fashion, a big nonterminal A of an MCFTG generates nonempty forests in $P_\Sigma(X)^*$ of the same rank as A, as defined next. Let $G = (N, \mathcal{N}, \Sigma, S, R)$ be an MCFTG. For every big nonterminal $A \in \mathcal{N}$ we define the *forest language generated by* A, denoted by $L(G, A)$, as follows. For all big nonterminals $A \in \mathcal{N}$ simultaneously, $L(G, A) \subseteq P_\Sigma(X)^*$ is the smallest set of forests such that for every rule $A \to (u, \mathcal{L}) \in R$, if $f \colon \mathcal{L} \to P_\Sigma(X)^*$ is a substitution function for $\mathcal{L}$ such that $f(B) \in L(G, B)$ for every $B \in \mathcal{L}$, then $u[f] \in L(G, A)$. Note that $u[f]$ is a simultaneous second-order substitution as defined at the end of Section 2.3. The fact that f is a substitution function for $\mathcal{L}$ means that $\mathrm{rk}(f(B)) = \mathrm{rk}(B)$ for every $B \in \mathcal{L}$, which implies that $\mathrm{rk}(t) = \mathrm{rk}(A)$ for every $t \in L(G, A)$; in particular, t is a nonempty forest of the same length as A. The *tree language* $L(G)$ *generated by* G is defined by $L(G) = L(G, S) \subseteq T_\Sigma$. Two MCFTGs G_1 and G_2 are *equivalent* if $L(G_1) = L(G_2)$.[12] A tree language is *multiple context-free* (*multiple regular*, (simple) *context-free*) if it is generated by an MCFTG (MRTG, spCFTG). The corresponding class of generated tree languages is denoted by MCFT (MRT, $\mathrm{CFT}_{\mathrm{sp}}$).

As observed above, each big nonterminal can be viewed as a nonempty subset of N, together with a fixed linear order on its elements. It is easy to see that the tree language $L(G)$ generated by G does not depend on that order. For a given big nonterminal $A = (A_1, \ldots, A_n)$ and a given permutation $A' = (A_{i_1}, \ldots, A_{i_n})$ of A, we can change every rule $A \to ((u_1, \ldots, u_n), \mathcal{L})$ into the rule $A' \to ((u_{i_1}, \ldots, u_{i_n}), (\mathcal{L} \setminus \{A\}) \cup \{A'\})$, provided that we also change $\mathcal{L}(\rho)$ into $(\mathcal{L}(\rho) \setminus \{A\}) \cup \{A'\}$ for every other rule $\rho \in R$.

The restriction that the right-hand side of a rule of G must be uniquely N-labeled can be compensated for by the appropriate use of aliases. Two big nonterminals $A, A' \in \mathcal{N}$ are said to be *aliases* if $\{(u, \mathcal{L}) \mid A \to (u, \mathcal{L}) \in R\} = \{(u, \mathcal{L}) \mid A' \to (u, \mathcal{L}) \in R\}$. It is not difficult to see that $L(G, A) = L(G, A')$ for aliases A and A'. Of course, in examples, we need not specify the rules of an alias (but we often will). Additionally, to improve the readability of examples, we will write a rule $A \to (u, \mathcal{L})$ as $\mathrm{in}(A) \to u$ and specify $\mathcal{L}$ separately. Recall from Section 2.3 that if $A = (A_1, \ldots, A_n)$ and $\mathrm{rk}(A_i) = k_i$ for every $i \in [n]$, then

$$\mathrm{in}(A) = (A_1(x_1, \ldots, x_{k_1}), \ldots, A_n(x_1, \ldots, x_{k_n})) \ .$$

If all the big nonterminals of G are mutually disjoint, in the sense that they have no nonterminals in common (i.e., $\mathrm{occ}(B) \cap \mathrm{occ}(B') = \emptyset$ for all distinct $B, B' \in \mathcal{N}$), then it is not even necessary to specify $\mathcal{L}$ because it clearly is equal to $\{B \in \mathcal{N} \mid \mathrm{occ}(B) \subseteq \mathrm{occ}_N(u)\}$.

Example 6 We first consider the MRTG $G = (N, \mathcal{N}, \Sigma, S, R)$ such that (i) $N = \{S, A, B, A', B'\}$, (ii) $\mathcal{N} = \{S, (A, B), (A', B')\}$, and (iii) $\Sigma = \{\sigma^{(2)}, \pi^{(2)}, \bar{\pi}^{(2)}, a^{(0)}\}$. Thus, $\mu(G) = 2$. And $\theta(G) = 0$ because G is a multiple regular tree grammar. The big nonterminal (A', B') is an alias of (A, B). The set R contains the rules (illustrated in Figure 1)

$$\begin{array}{lll} S \to \sigma(A, B) & (A, B) \to (\pi(A, A'), \bar{\pi}(B, B')) & (A', B') \to (\pi(A, A'), \bar{\pi}(B, B')) \\ & (A, B) \to (a, a) & (A', B') \to (a, a) \ . \end{array}$$

Since the big nonterminals in $\mathcal{N}$ are mutually disjoint, the set $\mathcal{L}$ of links of each rule is uniquely determined. In fact, $\mathcal{L} = \{(A, B)\}$ for the leftmost rule in the first line, $\mathcal{L} = \{(A, B), (A', B')\}$ for the two remaining rules in the first line, and $\mathcal{L} = \emptyset$ for the two rules in the second line. The tree language $L(G)$ generated by G consists of all trees $\sigma(t, \bar{t})$, where t is a tree over $\{\pi, a\}$ and $\bar{t}$ is the same tree with every π replaced by $\bar{\pi}$. For readers familiar with the multiple context-free grammars of [87] we note that this tree language can be generated by such a grammar with nonterminals S and C, where C corresponds to our big nonterminal (A, B) and its alias, using the three rules

- $S \to f[C]$ with $f(x_{11}, x_{12}) = \sigma x_{11} x_{12}$,
- $C \to g[C, C]$ with $g(x_{11}, x_{12}, x_{21}, x_{22}) = (\pi x_{11} x_{21}, \bar{\pi} x_{12} x_{22})$, and
- $C \to (a, a)$.

Note that the variables x_{11}, x_{12}, x_{21}, and x_{22} of [87] correspond to our nonterminals A, B, A', and B', respectively. In fact, every tree language in MRT can be generated by a multiple context-free grammar, just as every regular tree language can be generated by a context-free grammar (see Section 2.2). We will prove in Section 7 (Theorem 67) that this even holds for MCFT. □

[12] When viewing G_1 and G_2 as specifications of the string languages $\mathrm{yd}(L(G_1))$ and $\mathrm{yd}(L(G_2))$, they are *strongly equivalent* if $L(G_1) = L(G_2)$ and *weakly equivalent* if $\mathrm{yd}(L(G_1)) = \mathrm{yd}(L(G_2))$.

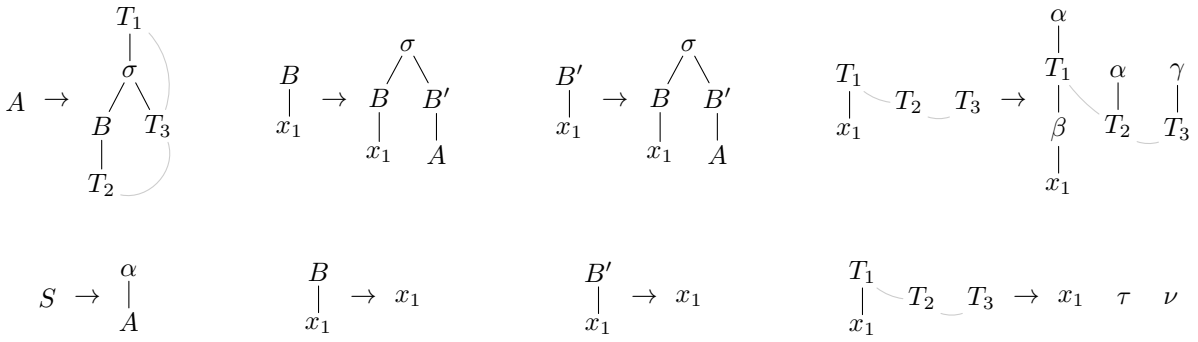


Figure 2: Rules of the MCFTG G of Example 7.

Example 7 As a second example we consider the MCFTG $G = (N, \mathcal{N}, \Sigma, S, R)$ such that

- $N = \{S^{(0)}, A^{(0)}, B^{(1)}, B'^{(1)}, T_1^{(1)}, T_2^{(0)}, T_3^{(0)}\}$ and $\mathcal{N} = \{S, A, B, B', (T_1, T_2, T_3)\}$, and
- $\Sigma = \{\sigma^{(2)}, \alpha^{(1)}, \beta^{(1)}, \gamma^{(1)}, \tau^{(0)}, \nu^{(0)}\}$.

Consequently, $\mu(G) = 3$ and $\theta(G) = 1$. The (big) nonterminal B' is an alias of B. The set R consists of the following rules $\rho_1, \ldots, \rho_6$ and the two rules ρ'_3 and ρ'_4 with left-hand side B' (illustrated in Figure 2).

$$
\begin{array}{llll}
\rho_1\colon & S \to \alpha(A) & \rho_2\colon & A \to T_1(\sigma(B(T_2), T_3)) \\
\rho_3\colon & B(x_1) \to \sigma(B(x_1), B'(A)) & \rho'_3\colon & B'(x_1) \to \sigma(B(x_1), B'(A)) \\
\rho_4\colon & B(x_1) \to x_1 & \rho'_4\colon & B'(x_1) \to x_1 \\
\rho_5\colon & (T_1(x_1), T_2, T_3) \to (\alpha(T_1(\beta(x_1))),\ \alpha(T_2),\ \gamma(T_3)) & \rho_6\colon & (T_1(x_1), T_2, T_3) \to (x_1, \tau, \nu)\ .
\end{array}
$$

Since, again, all big nonterminals in $\mathcal{N}$ are mutually disjoint, the sets of links of these rules are uniquely determined. They are, in fact, as follows:

$$
\begin{array}{lll}
\mathcal{L}(\rho_1) = \{A\} & \mathcal{L}(\rho_3) = \mathcal{L}(\rho'_3) = \{B, B', A\} & \mathcal{L}(\rho_2) = \{B, (T_1, T_2, T_3)\} \\
 & \mathcal{L}(\rho_4) = \mathcal{L}(\rho'_4) = \mathcal{L}(\rho_6) = \emptyset & \mathcal{L}(\rho_5) = \{(T_1, T_2, T_3)\}\ .
\end{array}
$$

Let $T = (T_1, T_2, T_3)$. The rule ρ_6 shows that $(x_1, \tau, \nu) \in L(G, T)$. We can write the rule ρ_5 also as $T \to (\alpha T_1 \beta x_1,\ \alpha T_2,\ \gamma T_3)$. Substituting (x_1, τ, ν) for T in $u_5 = \mathrm{rhs}(\rho_5)$ we obtain that $L(G, T)$ also contains the forest $u_5[(T_1, T_2, T_3) \leftarrow (x_1, \tau, \nu)] = (\alpha\beta x_1,\ \alpha\tau,\ \gamma\nu)$. Then, substituting this forest for T in u_5 we obtain that $L(G, T)$ also contains $(\alpha\alpha\beta\beta x_1,\ \alpha\alpha\tau,\ \gamma\gamma\nu)$. Continuing in this way we see that $L(G, T) = \{(\alpha^n \beta^n x_1, \alpha^n \tau, \gamma^n \nu) \mid n \in \mathbb{N}_0\}$. If we temporarily view A as a terminal, then $B(x_1)$ generates all trees $t \in T_{\{\sigma, A, x_1\}}$ such that the left-most leaf of t has label x_1 and all other leaves have label A. The right-hand side $u_2 = T_1(\sigma(B(T_2), T_3))$ of ρ_2 generates all trees $u_2[B \leftarrow t, T \leftarrow t']$ with t as above and $t' \in L(G, T)$; i.e., all trees $\alpha^n \beta^n \sigma(t[x_1 \leftarrow \alpha^n \tau], \gamma^n \nu)$. This should give an idea of the form of the trees in $L(G, A)$, and hence of the trees in $L(G)$. □

3.2. Derivation trees

The least fixed point semantics of an MCFTG $G = (N, \mathcal{N}, \Sigma, S, R)$ naturally leads to the notion of a derivation tree of G that we define now. We assume that for every rule ρ of G, the links in $\mathcal{L}(\rho)$ are linearly ordered by an arbitrary, fixed order $\sqsubseteq$. Whenever we write $\mathcal{L}(\rho) = \{B_1, \ldots, B_k\}$ with $B_i \in \mathcal{N}$ for all $i \in [k]$, we will assume that $B_1 \sqsubset \cdots \sqsubset B_k$. The *derivation tree grammar* of G is the RTG $G_{\mathrm{der}} = (N_{\mathrm{der}}, R, S, R_{\mathrm{der}})$ defined as follows.[13] First, $N_{\mathrm{der}} = \mathcal{N}$; i.e., its nonterminals (of rank 0) are the big nonterminals of G. Its initial nonterminal is S, which is the initial (big) nonterminal of G. Second, its terminal ranked alphabet is the set R of rules of G such that the rule ρ has rank $\mathrm{rk}(\rho) = |\mathcal{L}(\rho)|$.[14] Finally, the set R_{der} consists of all rules $A \to \rho(B_1, \ldots, B_k)$ such that $\rho \in R$, $\mathrm{lhs}(\rho) = A$, and $\mathcal{L}(\rho) = \{B_1, \ldots, B_k\}$. For $A \in \mathcal{N}$, a *derivation tree* of G of type A is a tree $d \in T_{\mathcal{N} \cup R}$

[13] See Section 2.2 for the definition of a regular tree grammar (RTG). Note that, in this contribution, RTGs are in normal form.

[14] Note that, therefore, the rule-width of G (as defined after Definition 5) is $\lambda(G) = mrk_R$, the maximal rank of its rules.

such that $A \Rightarrow^*_{G_{\mathrm{der}}} d$. Obviously, every derivation tree has a unique type, viz. $\mathrm{lhs}(d(\varepsilon))$; i.e., the left-hand side of the rule that labels its root. We will denote the set of derivation trees of G of type A by $DL(G_{\mathrm{der}}, A)$. Note that $L(G_{\mathrm{der}}, A) = DL(G_{\mathrm{der}}, A) \cap T_R$. To capture the semantics of G, only the derivation trees in $L(G_{\mathrm{der}}) \subseteq T_R$ are relevant, but we will need the other derivation trees for technical reasons in proofs. As in the case of context-free grammars, it can be checked locally whether a tree $d \in T_{\mathcal{N} \cup R}$ is a derivation tree. In fact, let us say that the type of a position $p \in \mathrm{pos}(d)$ is either $d(p)$ if $d(p) \in \mathcal{N}$, or $\mathrm{lhs}(d(p))$ if $d(p) \in R$. Then d is a derivation tree if and only if for every position $p \in \mathrm{pos}_R(d)$ with $\mathcal{L}(d(p)) = \{B_1, \ldots, B_k\}$, the child pi of p has type B_i for every $i \in [k]$.

The *value* of a derivation tree d of type A, denoted by $\mathrm{val}(d)$, is a forest in $P_{N \cup \Sigma}(X)^+$ of the same rank as A in G, and is defined inductively as follows. If $d = A \in \mathcal{N}$, then $\mathrm{val}(d) = \mathrm{in}(A)$. If $d = \rho(d_1, \ldots, d_k)$ for some $\rho = A \to (u, \mathcal{L}) \in R$ with $\mathcal{L} = \{B_1, \ldots, B_k\}$ (and thus d_i is of type B_i for every $i \in [k]$), then $\mathrm{val}(d) = u[B_i \leftarrow \mathrm{val}(d_i) \mid 1 \leq i \leq k]$. The value $\mathrm{val}(d)$ of the derivation tree d can clearly be computed in linear time. We also observe here that its computation can be realized by a macro tree transducer [13, 34] (see Lemma 73 in Section 8). Since that macro tree transducer is finite-copying, 'val' can also be realized by a deterministic MSO-transducer (see [26]).

Example 8 The derivation tree grammar G_{der} of the grammar G of Example 7 has the following eight rules, where $T = (T_1, T_2, T_3)$ and the linear order of the links of each rule of G is fixed as indicated in Example 7:

$$\begin{array}{ll} S \to \rho_1(A) & A \to \rho_2(B, T) \\ B \to \rho_3(B, B', A) & B' \to \rho'_3(B, B', A) \\ B \to \rho_4 & B' \to \rho'_4 \\ T \to \rho_5(T) & T \to \rho_6 \ . \end{array}$$

Rules of Example 7:

ρ_1:	$S \to \alpha(A)$	ρ_2:	$A \to T_1(\sigma(B(T_2), T_3))$
ρ_3:	$B(x_1) \to \sigma(B(x_1), B'(A))$	ρ'_3:	$B'(x_1) \to \sigma(B(x_1), B'(A))$
ρ_4:	$B(x_1) \to x_1$	ρ'_4:	$B'(x_1) \to x_1$
ρ_5:	$(T_1(x_1), T_2, T_3) \to (\alpha(T_1(\beta(x_1))), \alpha(T_2), \gamma(T_3))$	ρ_6:	$(T_1(x_1), T_2, T_3) \to (x_1, \tau, \nu)$.

An example of a derivation tree of type A is $d = \rho_2(\rho_3(\rho_4, B', A), \rho_5(\rho_6))$, which is shown in Figure 3. Obviously, $\mathrm{val}(\rho_4) = x_1$ and we have $\mathrm{val}(\rho_6) = (x_1, \tau, \nu)$. Then $\mathrm{val}(\rho_5(\rho_6))$ is obtained by substituting (x_1, τ, ν) for $T = (T_1, T_2, T_3)$ in the right-hand side of rule ρ_5. We saw in Example 7 that the result is $(\alpha\beta x_1, \alpha\tau, \gamma\nu)$. Similarly, $\mathrm{val}(\rho_3(\rho_4, B', A))$ is obtained from $\mathrm{rhs}(\rho_3)$ by substituting $\mathrm{val}(\rho_4) = x_1$ for B (and simultaneously substituting $\mathrm{in}(B')$ for B' and $\mathrm{in}(A)$ for A, without effect). The result is $\sigma(x_1, B'(A))$. Finally, $\mathrm{val}(d)$ is obtained from $\mathrm{rhs}(\rho_2)$ by substituting $\sigma(x_1, B'(A))$ for B and $(\alpha\beta x_1, \alpha\tau, \gamma\nu)$ for T. Hence $\mathrm{val}(d) = \alpha\beta(\sigma(\sigma(\alpha\tau, B'(A)), \gamma\nu))$. The process is illustrated in Figure 3. An example of a derivation tree in $L(G_{\mathrm{der}}, S)$ is

$$d' = \rho_1(d[(B', A) \leftarrow (\rho'_4, \rho_2(\rho_4, \rho_6))]) \ ,$$

which equals $\rho_1(\rho_2(\rho_3(\rho_4, \rho'_4, \rho_2(\rho_4, \rho_6)), \rho_5(\rho_6)))$. Clearly, $\mathrm{val}(\rho'_4) = x_1$ and $\mathrm{val}(\rho_2(\rho_4, \rho_6)) = \sigma(\tau, \nu)$. It is straightforward to compute $\mathrm{val}(d') = \alpha\alpha\beta(\sigma(\sigma(\alpha\tau, \sigma(\tau, \nu)), \gamma\nu)) = \alpha(\mathrm{val}(d)[(B', A) \leftarrow (x_1, \sigma(\tau, \nu))])$, which shows that 'val' distributes over substitution. □

From the least fixed point semantics we immediately obtain a characterization by derivation trees.

Theorem 9 *$L(G, A) = \mathrm{val}(L(G_{\mathrm{der}}, A))$ for every $A \in \mathcal{N}$. In particular, $L(G) = \mathrm{val}(L(G_{\mathrm{der}}))$.*

Proof Obviously, the sets $\mathrm{val}(L(G_{\mathrm{der}}, A))$ satisfy the fixed point requirement for all $A \in \mathcal{N}$, which says that for every rule $\rho = A \to (u, \mathcal{L}) \in R$ and substitution function f for $\mathcal{L}$ such that $f(B)$ is in $\mathrm{val}(L(G_{\mathrm{der}}, B))$ for every $B \in \mathcal{L}$, we have that $u[f] \in \mathrm{val}(L(G_{\mathrm{der}}, A))$. In fact, if $\mathcal{L} = \{B_1, \ldots, B_k\}$ and $f(B_i) = \mathrm{val}(d_i)$ for all $i \in [k]$, then $u[B \leftarrow f(B) \mid B \in \mathcal{L}]$ is equal to $\mathrm{val}(\rho(d_1, \ldots, d_k))$ by definition of 'val'. This shows that $L(G, A) \subseteq \mathrm{val}(L(G_{\mathrm{der}}, A))$ for every $A \in \mathcal{N}$. In the other direction, it is easy to show that $\mathrm{val}(d) \in L(G, A)$ for every $d \in L(G_{\mathrm{der}}, A)$ and every $A \in \mathcal{N}$ by induction on the structure of the derivation tree d. ■

This theorem implies that the emptiness problem is decidable for $L(G)$ and $L(G, A)$. In fact, $L(G) = \emptyset$ if and only if $L(G_{\mathrm{der}}) = \emptyset$, which is decidable because G_{der} is an RTG; and similarly for $L(G, A)$. It is now also very easy to see that $L(G, A) = L(G, A')$ for aliases A and A': if $\rho = A \to (u, \mathcal{L})$ and $\rho' = A' \to (u, \mathcal{L})$ are rules and $d = \rho(d_1, \ldots, d_k)$ is in $L(G_{\mathrm{der}}, A)$, then $d' = \rho'(d_1, \ldots, d_k)$ is in $L(G_{\mathrm{der}}, A')$ and $\mathrm{val}(d) = \mathrm{val}(d')$, under the assumption that $\mathcal{L}$ has the same linear order in ρ and ρ'.

We will need three simple properties of derivation trees, which are stated in the next three lemmas. The first is a generalization of Lemma 1(2) and states that for every derivation tree of G, the number of occurrences of a terminal in $\mathrm{val}(d)$ is the sum of its occurrences in the right-hand sides of the rules that occur in d. Also, the number of occurrences of a nonterminal in $\mathrm{val}(d)$ is equal to the number of its "occurrences" (as part of a big nonterminal) in d.

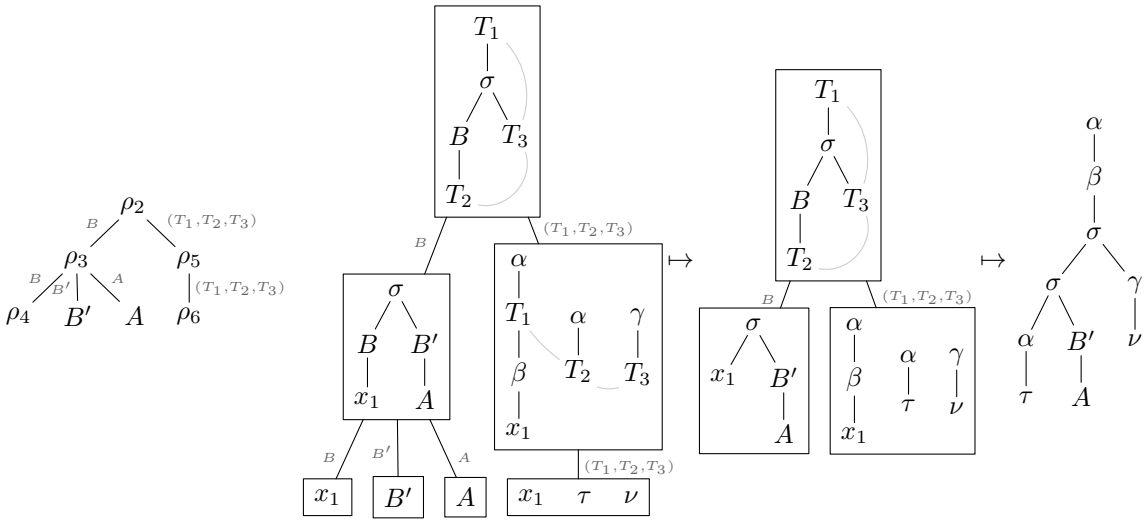


Figure 3: Derivation tree of the MCFTG G of Example 7 and illustration of the (bottom-up) computation of its value.

Lemma 10 *Let $d \in DL(G_{\mathrm{der}}, A)$ with $A \in \mathcal{N}$, and let $\sigma \in \Sigma$ and $C \in N$.*
(1) $|\mathrm{pos}_\sigma(\mathrm{val}(d))| = \sum_{p \in \mathrm{pos}_R(d)} |\mathrm{pos}_\sigma(\mathrm{rhs}(d(p)))|$.
(2) $|\mathrm{pos}_C(\mathrm{val}(d))| = \sum_{B \in \mathcal{N}_C} |\mathrm{pos}_B(d)|$, *where* $\mathcal{N}_C = \{B \in \mathcal{N} \mid C \in \mathrm{occ}(B)\}$.
(3) $\mathrm{val}(d) \in T_\Sigma$ *if and only if* $d \in T_R$.

Proof The proofs of (1) and (2) can be achieved by induction on the structure of d. The statements are obvious for $d = A \in \mathcal{N}$ because we obtain $0 = 0$ in (1), $1 = 1$ in (2) if $C \in \mathrm{occ}(A)$, and $0 = 0$ in (2) otherwise. Let us now consider $d = \rho(d_1, \ldots, d_k)$ for some rule $\rho = A \to (u, \mathcal{L})$ with $\mathcal{L} = \{B_1, \ldots, B_k\}$. By the definition of 'val' we have $\mathrm{val}(d) = u[B_i \leftarrow \mathrm{val}(d_i) \mid 1 \leq i \leq k]$, which equals the second-order substitution $u[B_1 \cdots B_k \leftarrow \mathrm{val}(d_1) \cdots \mathrm{val}(d_k)]$ by the definition of simultaneous second-order substitution. Let h be the tree homomorphism over $N \cup \Sigma$ corresponding to $[B_1 \cdots B_k \leftarrow \mathrm{val}(d_1) \cdots \mathrm{val}(d_k)]$. It is now straightforward to prove (1) and (2) using Lemma 1(2) and the induction hypotheses for $d_1, \ldots, d_k$. It follows from (2) that $\mathrm{occ}_N(\mathrm{val}(d)) = \bigcup_{B \in \mathrm{occ}_{\mathcal{N}}(d)} \mathrm{occ}(B)$, which proves (3). ∎

The second property is that 'val' distributes over second-order substitution, of which an example was presented at the end of Example 8. It can be viewed as a generalization of Lemma 4(4). For convenience, and because it is all we will need, we only prove this for the case where just one big nonterminal is replaced.

Lemma 11 *Let $A, B \in \mathcal{N}$, and let $d \in DL(G_{\mathrm{der}}, A)$ and $d' \in DL(G_{\mathrm{der}}, B)$ be derivation trees of type A and B such that $B \in \mathrm{occ}_{\mathcal{N}}(d)$. Then* $\mathrm{val}(d[B \leftarrow d']) = \mathrm{val}(d)[B \leftarrow \mathrm{val}(d')]$.

Proof As in Lemma 10, we proceed by induction on the structure of d. For $d = A \in \mathcal{N}$ both sides of the equation are equal to $\mathrm{val}(d')$ if $B = A$ and equal to $\mathrm{in}(A)$ otherwise. Now we consider $d = \rho(d_1, \ldots, d_k)$ for some $\rho = A \to (u, \mathcal{L})$ with $\mathcal{L} = \{B_1, \ldots, B_k\}$. Then

$$\begin{aligned}
&\mathrm{val}(d)[B \leftarrow \mathrm{val}(d')] \\
&= u[B_1 \cdots B_k \leftarrow \mathrm{val}(d_1) \cdots \mathrm{val}(d_k)]\,[B \leftarrow \mathrm{val}(d')] = u\big[B_1 \cdots B_k \leftarrow (\mathrm{val}(d_1) \cdots \mathrm{val}(d_k))[B \leftarrow \mathrm{val}(d')]\,\big] \\
&= u\big[B_1 \cdots B_k \leftarrow \mathrm{val}(d_1)[B \leftarrow \mathrm{val}(d')] \cdots \mathrm{val}(d_k)[B \leftarrow \mathrm{val}(d')]\,\big] \\
&= u\big[B_1 \cdots B_k \leftarrow \mathrm{val}(d_1[B \leftarrow d']) \cdots \mathrm{val}(d_k[B \leftarrow d'])\big] = \mathrm{val}(\rho(d_1[B \leftarrow d'], \ldots, d_k[B \leftarrow d'])) \\
&= \mathrm{val}(d[B \leftarrow d'])\ ,
\end{aligned}$$

where the second equality is by Lemma 4(4) and the fourth by the induction hypotheses. ∎

We will use the following simple third property in the proofs of Lemmas 28 and 32.

Lemma 12 *Let $F \subseteq R$, $\mathcal{N}' \subseteq \mathcal{N}$, and $\mathcal{D}_B = DL(G_{\mathrm{der}}, B) \cap T_{\mathcal{N}' \cup F}$ for every $B \in \mathcal{N}$. Moreover, let $A \in \mathcal{N}$, $t \in \mathrm{val}(\mathcal{D}_A)$, and $L_{\langle A,t \rangle} = \{d \in \mathcal{D}_A \mid \mathrm{val}(d) = t\}$. If $\mathrm{val}(\mathcal{D}_B)$ is finite for every $B \in \mathcal{N}$, then $L_{\langle A,t \rangle}$ is a regular tree language.*

Proof An RTG for $L_{\langle A,t\rangle}$ has the nonterminals $\langle B, v\rangle$ with $B \in \mathcal{N}$ and $v \in \mathrm{val}(\mathcal{D}_B)$, of which the nonterminal $\langle A, t\rangle$ is initial. For every rule $\rho = B \to (u, \mathcal{L})$ of G with $\rho \in F$ and $\mathcal{L} = \{B_1, \dots, B_k\}$, it has all the rules $\langle B, v\rangle \to \rho(\langle B_1, v_1\rangle, \dots, \langle B_k, v_k\rangle)$ such that $v_i \in \mathrm{val}(\mathcal{D}_{B_i})$ for every $i \in [k]$, and $v = u[B_i \leftarrow v_i \mid 1 \le i \le k]$. Moreover, for every $B \in \mathcal{N}'$ it has the rule $\langle B, \mathrm{in}(B)\rangle \to B$. This grammar can be viewed as a deterministic bottom-up finite tree automaton [41, 42] that, for every derivation tree $d \in T_{\mathcal{N}' \cup F}$, computes the type of d and its value $\mathrm{val}(d)$. ∎

Let us turn to the comparison of the derivation trees of two MCFTGs G and G'. We can define G and G' to be "$\mathcal{X}$-equivalent", where $\mathcal{X}$ is a class of tree transductions, if there are value-preserving tree transductions in $\mathcal{X}$ from the derivation trees of each grammar to those of the other grammar. The idea here is that G and G' are grammatically closely related if $\mathcal{X}$ is a relatively simple class of tree transductions. For that purpose we choose the class $\mathcal{X} = \mathrm{LDT}^{\mathrm{R}}$, which we define now. To define tree transducers we use the infinite alphabet $Y = \{y_1, y_2, \dots\}$ of *input variables* to avoid confusion with the set X of variables used in MCFTGs (the set X will also be used as output variables, or parameters, for macro tree transducers in Section 8). For every $k \in \mathbb{N}_0$, we let $Y_k = \{y_i \mid i \in [k]\}$.

A *linear deterministic top-down tree transducer with regular look-ahead* (in short, $\mathrm{LDT}^{\mathrm{R}}$-transducer) from Ω to Σ is a system $M = (Q, \Omega, \Sigma, q_0, R)$, where Q is a finite set of *states*, Ω and Σ are finite ranked alphabets of *input* and *output symbols* with $Q \cap \Sigma = \emptyset$, $q_0 \in Q$ is the *initial state*, and R is a finite set of *rules*. Each rule in R is of the form

$$\langle q, \omega(y_1 \colon L_1, \dots, y_k \colon L_k) \colon L_0\rangle \to \zeta \ ,$$

where $q \in Q$, $k \in \mathbb{N}_0$, $\omega \in \Omega^{(k)}$, $L_0, L_1, \dots, L_k$ are regular tree languages over Ω (specified, e.g., by RTGs), and $\zeta \in T_{(Q \times Y_k) \cup \Sigma}$ using the ranked alphabet $Q \times Y_k$, in which every element has rank 0. Additionally, we require that each $y \in Y_k$ occurs at most once in ζ (linearity property), and that if $\langle q, \omega(y_1 \colon L'_1, \dots, y_k \colon L'_k) \colon L'_0\rangle \to \zeta'$ is another rule in R (for the same q and ω), then there exists an index $0 \le i \le k$ such that $L_i \cap L'_i = \emptyset$ (determinism property). If $L_i = T_\Omega$ in the above rule, then we omit '$: L_i$'. An $\mathrm{LDT}^{\mathrm{R}}$-transducer is called an LDT-transducer (without regular look-ahead) if $L_i = T_\Omega$ for every $0 \le i \le k$ in every rule.

For every input tree $s \in T_\Omega$ and every state $q \in Q$, we define the *q-translation* of s by M, denoted by $M_q(s)$, inductively as follows. If $s = \omega(s_1, \dots, s_k)$, the above rule is in R, $s \in L_0$, and $s_i \in L_i$ for every $i \in [k]$, then

$$M_q(s) = \zeta[\langle q', y_i\rangle \leftarrow M_{q'}(s_i) \mid q' \in Q, 1 \le i \le k] \ .$$

We observe that $M_q(s)$ is undefined if there does not exist an appropriate rule or, using the rule above, $M_{q'}(s_i)$ is undefined for some $\langle q', y_i\rangle$ that occurs in ζ. Moreover, the *tree transduction realized by* M, also denoted by M, is the partial function $M \colon T_\Omega \to T_\Sigma$, which is given by $M(s) = M_{q_0}(s)$ for every $s \in T_\Omega$. The tree $M(s)$, provided it is defined, is also called the *translation* of s by M. We denote by $\mathrm{LDT}^{\mathrm{R}}$ the class of all tree transductions realized by $\mathrm{LDT}^{\mathrm{R}}$-transducers. Note that every tree homomorphism $\hat{h}$ from Ω to Σ can be realized by an LDT-transducer with one state q and with the rules $\langle q, \omega(y_1, \dots, y_k)\rangle \to h(\omega)[x_i \leftarrow \langle q, y_i\rangle \mid 1 \le i \le k]$ for every $k \in \mathbb{N}_0$ and $\omega \in \Omega^{(k)}$. We need the following two basic properties of $\mathrm{LDT}^{\mathrm{R}}$.

Proposition 13 $\mathrm{LDT}^{\mathrm{R}}$ *is closed under composition.*

Proof This is stated after [17, Theorem 2.11]. Part (2) of its proof shows the statement because the constructions in the proofs of [17, Lemmas 2.9 and 2.10] preserve linearity. ∎

An $\mathrm{LDT}^{\mathrm{R}}$-transducer M is a *finite-state relabeling* if, in each of its rules as above, ζ is of the form $\sigma(\langle q_1, y_1\rangle, \dots, \langle q_k, y_k\rangle)$ for some $\sigma \in \Sigma^{(k)}$ and $q_1, \dots, q_k \in Q$. Such a transducer just changes the labels of the nodes of the input tree. Note that every projection is a finite-state relabeling.

Proposition 14 *For every* $\mathrm{LDT}^{\mathrm{R}}$*-transducer* $M = (Q, \Omega, \Sigma, q_0, R)$ *there is a polynomial time algorithm that, for every RTG* H *over* Σ *as input, outputs an RTG* H' *over* Ω *such that* $L(H') = M^{-1}(L(H))$*. If* M *is a finite-state relabeling, then there is a linear time algorithm for the same task.*

Proof It is well known that the class RT is closed under inverse $\mathrm{LDT}^{\mathrm{R}}$-transductions [17, Lemma 1.2 and Theorem 2.6]. We now show that the transformation can be realized in polynomial time, for fixed M. By [17, Theorem 2.8] and (the proof of) [15, Theorem 3.5], the transduction M can be

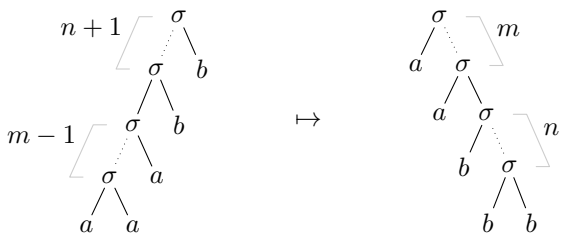


Figure 4: Translating left-recursive into right-recursive trees.

written as the "bimorphism" $\{(\hat{\pi}(t), \hat{h}(t)) \mid t \in K\}$, where K is a regular tree language over a finite alphabet Δ, π is a projection from Δ to Ω, and h is a tree homomorphism from Δ to Σ. Therefore $M^{-1}(L(H)) = \hat{\pi}(\hat{h}^{-1}(L(H)) \cap K)$. Hence, since the intersection with K and the projection $\hat{\pi}$ can be realized in linear time because K and π are fixed, we may assume in the remainder of this proof that M is a tree homomorphism h from Ω to Σ.

Now let $H = (N, \Sigma, S, R_H)$. As mentioned at the end of Section 2.2, we assume that H is in normal form; i.e., that the rules in R_H are of the form $A_0 \to \sigma(A_1, \ldots, A_m)$ with $m \in \mathbb{N}_0$, $\sigma \in \Sigma^{(m)}$, and $A_1, \ldots, A_m \in N$. We construct $H' = (N, \Omega, S, R')$ such that for every $k \in \mathbb{N}_0$, $\omega \in \Omega^{(k)}$, and $A_0, A_1, \ldots, A_k \in N$, if $A_0 \Rightarrow_H^* h(\omega)[x_i \leftarrow A_i \mid 1 \leq i \leq k]$, then the rule $A_0 \to \omega(A_1, \ldots, A_k)$ is in R'. It is straightforward to show that $L(H', A) = \hat{h}^{-1}(L(H, A))$ for every $A \in N$. It should be clear that the construction of H' takes polynomial time (in the size of H). In fact, it takes time $O(n^k)$ where n is the size of H and $k = mrk_\Omega + 1$ (and recall that mrk_Ω is the maximal rank of the symbols in Ω). If M is a finite-state relabeling, then it can be checked that h is also a projection. Hence the set R' can be constructed such that if $h(\omega) = \mathrm{in}(\sigma)$ and $A_0 \to \sigma(A_1, \ldots, A_k)$ is in R_H, then $A_0 \to \omega(A_1, \ldots, A_k)$ is in R'. That construction only takes linear time. ■

We now define $\mathcal{X}$-equivalence of MCFTGs G and G' for $\mathcal{X} = \mathrm{LDT^R}$. However, for future use, we give a more general definition that involves a tree transformation φ and implies that $L(G') = \varphi(L(G))$.

Definition 15 Let $G = (N, \mathcal{N}, \Sigma, S, R)$ and $G' = (N', \mathcal{N}', \Sigma', S', R')$ be MCFTGs, and let φ be a mapping from T_Σ to $T_{\Sigma'}$. The grammar G' is $\mathrm{LDT^R}$-*φ-equivalent* to the grammar G if there exist tree transductions $M\colon T_R \to T_{R'}$ and $M'\colon T_{R'} \to T_R$ in $\mathrm{LDT^R}$ such that

(1) $M(d) \in L(G'_{\mathrm{der}})$ and $\mathrm{val}(M(d)) = \varphi(\mathrm{val}(d))$ for every $d \in L(G_{\mathrm{der}})$, and vice versa,

(2) $M'(d') \in L(G_{\mathrm{der}})$ and $\varphi(\mathrm{val}(M'(d'))) = \mathrm{val}(d')$ for every $d' \in L(G'_{\mathrm{der}})$.

In particular, $M(d)$ must be defined for every $d \in L(G_{\mathrm{der}})$, and similarly for $M'(d')$.

The grammars G and G' are $\mathrm{LDT^R}$-*equivalent* if $\Sigma = \Sigma'$ and φ is the identity on T_Σ. □

It directly follows from item (1) and Theorem 9 that $\varphi(L(G)) \subseteq L(G')$, and $L(G') \subseteq \varphi(L(G))$ follows from item (2). Hence $L(G') = \varphi(L(G))$. In particular, $\mathrm{LDT^R}$-equivalent MCFTGs are equivalent. Since $\mathrm{LDT^R}$ is closed under composition by Proposition 13, $\mathrm{LDT^R}$-equivalence of MCFTGs is an equivalence relation. That is, of course, not true for $\mathrm{LDT^R}$-φ-equivalence in general.

It should be noted that the notion of $\mathrm{LDT^R}$-φ-equivalence is independent of the linear order of the links in the rules of G and G'. In fact, if $\rho = A \to (u, \mathcal{L})$ is a rule of G with $\mathcal{L} = \{B_1, \ldots, B_k\}$ and we change that order into $\{B_{i_1}, \ldots, B_{i_k}\}$, where $(i_1, \ldots, i_k)$ is a permutation of $(1, \ldots, k)$, then a tree homomorphism h over R can transform the old derivation trees into the new ones via $h(\rho) = \rho(x_{i_1}, \ldots, x_{i_k})$. That proves the observation because tree homomorphisms are in $\mathrm{LDT^R}$ and $\mathrm{LDT^R}$ is closed under composition. Thus, whenever we construct a new grammar G or G', we can choose those orders in a convenient way.

As observed above, $\mathrm{LDT^R}$-equivalent grammars G and G' are grammatically closely related by means of the $\mathrm{LDT^R}$-transducers M and M'. Consequently, their parsing problems are closely related as well because the transducer M' transforms a derivation tree of G' with value $t \in T_\Sigma$ in linear time into one of G with the same value t. Moreover, if H' is an RTG that generates all derivation trees of G' with value t, then an RTG H can be constructed in polynomial time that generates all derivation trees of G with value t. This follows from Proposition 14 because $L(H) = M^{-1}(L(H')) \cap L(G_{\mathrm{der}})$. The parsing problem for MCFTGs will be discussed in more detail in Section 7.2.

An important example of a tree transduction that *cannot* be realized by an $\mathrm{LDT^R}$-transducer is the transformation of a left-recursive tree into a right-recursive tree with the same yield. More precisely, for

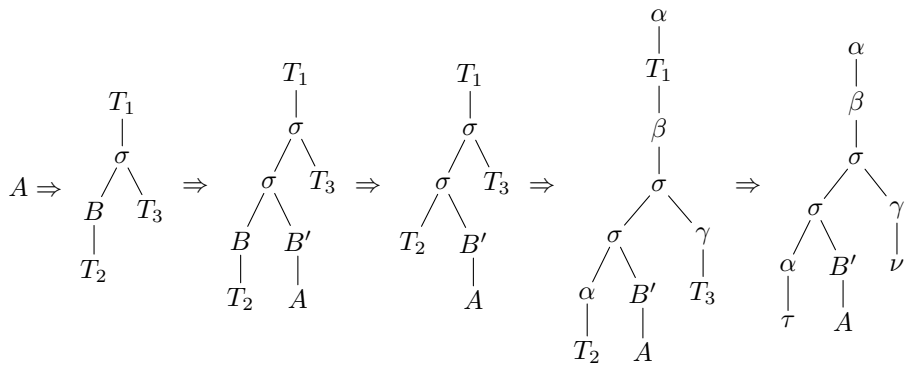


Figure 5: A naive (leftmost) derivation of the grammar G of Example 7, corresponding to the derivation tree in Example 8 and in Figure 3. All big nonterminals of G are mutually disjoint and all the trees in this derivation are uniquely N-labeled.

the ranked alphabet $\Sigma = \{\sigma^{(2)}, a^{(0)}, b^{(0)}\}$, the tree transformation

$$\tau = \{(\sigma^{m+n} a^m b^{n+1},\ (\sigma a)^m (\sigma b)^n b) \mid m, n \in \mathbb{N}_0\} \ ,$$

which translates the left-recursive tree $\sigma^{m+n} a^m b^{n+1}$ into the right-recursive tree $(\sigma a)^m (\sigma b)^n b$ (see Figure 4), cannot be realized by any $\mathrm{LDT^R}$-transducer; i.e., $\tau \notin \mathrm{LDT^R}$. This can be proved by a classical pumping argument; if there would be such a transducer, then the language $\{b^{n+1} a^m b^{n+1} a^m \mid m, n \in \mathbb{N}_0\}$ would be linear context-free. By a similar argument one can show that there is no yield-preserving $\mathrm{LDT^R}$-transducer that translates the derivation trees of the left-recursive context-free grammar with rules $S \to Sa$, $S \to Sb$, $S \to a$, and $S \to b$ into the derivation trees of an equivalent context-free grammar in Greibach Normal Form. In other words, the transformation of a context-free grammar into Greibach Normal Form involves a grammatical transformation of derivation trees that cannot be realized by any $\mathrm{LDT^R}$-transducer.

3.3. Derivations

In this subsection we present a rewriting semantics of MCFTGs, inspired by the level grammars of [88]. The definitions and results of this subsection will not be utilized in the other sections, but we hope that they improve the intuition of the reader concerning MCFTGs.

Let $G = (N, \mathcal{N}, \Sigma, S, R)$ be an MCFTG. In a naive approach we would define the derivation steps of G on trees $t \in T_{N \cup \Sigma}$ and the application of a rule $A \to (u, \mathcal{L})$ to t leading to a derivation step $t \Rightarrow t[A \leftarrow u]$, provided that $\mathrm{occ}(A) \subseteq \mathrm{occ}_N(t)$. Such a naive derivation is shown in Figure 5 for the grammar G of Example 7. Assuming that all big nonterminals of G are mutually disjoint (as in Example 7), this naive derivation step works if A occurs exactly once in t (e.g., when t is uniquely N-labeled). However, it fails if A occurs several times in t because the rule is then applied to all occurrences simultaneously. Moreover, if $A = (A_1, A_2)$ with $A_1, A_2 \in N$, then it is unclear which occurrences of A_1 and A_2 are linked. If not all big nonterminals of G are mutually disjoint, then it is not clear at all which nonterminals in t are linked (even when t is uniquely N-labeled). Thus, we additionally have to keep track of how the nonterminal occurrences in t are linked together to form occurrences of big nonterminals. To facilitate this, we change for each position $p \in \mathrm{pos}_N(t)$ of t the label $t(p)$ into an appropriate label $\langle t(p), \ell \rangle$, where $\ell \in \mathbb{N}^*$ is a position, which is also called *link identifier*. Nonterminal occurrences with the same link identifier ℓ are linked, and we only derive uniquely $(N \times \mathbb{N}^*)$-labeled trees. We note that the positions p and ℓ need not coincide. In fact, ℓ is a position of the derivation tree corresponding to the derivation.

We need additional notation for the formalization. As in the previous subsection, we assume that for every rule ρ of G the set $\mathcal{L}(\rho)$ of links is linearly ordered. For a big nonterminal $A = (A_1, \ldots, A_n) \in \mathcal{N}$ and a link identifier $\ell \in \mathbb{N}^*$, we define $A \otimes \ell = (\langle A_1, \ell \rangle, \ldots, \langle A_n, \ell \rangle) \in (N \times \mathbb{N}^*)^+$. Moreover, for $\ell \in \mathbb{N}^*$ and a rule $\rho = A \to (u, \mathcal{L}) \in R$ with $\mathcal{L} = \{B_1, \ldots, B_k\}$, we define $(u, \mathcal{L}) \otimes \ell = u[B_i \leftarrow \mathrm{in}(B_i \otimes \ell i) \mid 1 \le i \le k]$. Note that $(u, \mathcal{L}) \otimes \ell$ is a forest obtained from u by appropriately relabeling its N-labeled positions.

Now let $t_1, t_2 \in T_{(N \times \mathbb{N}^*) \cup \Sigma}$ be trees, $\rho = A \to (u, \mathcal{L}) \in R$ be a rule, and $\ell \in \mathbb{N}^*$ be a link identifier. We define the *derivation step* $t_1 \Rightarrow_G^{\rho,\ell} t_2$ if $\mathrm{occ}(A \otimes \ell) = \mathrm{occ}_{N \times \{\ell\}}(t_1)$ and $t_2 = t_1[A \otimes \ell \leftarrow (u, \mathcal{L}) \otimes \ell]$. Intuitively, $A \otimes \ell$ occurs in t_1 (and no other nonterminals with link identifier ℓ occur in t_1) and the occurrence of $A \otimes \ell$ is replaced by $(u, \mathcal{L}) \otimes \ell$. We write $t_1 \Rightarrow_G t_2$ if there exist ρ and ℓ such that $t_1 \Rightarrow_G^{\rho,\ell} t_2$.

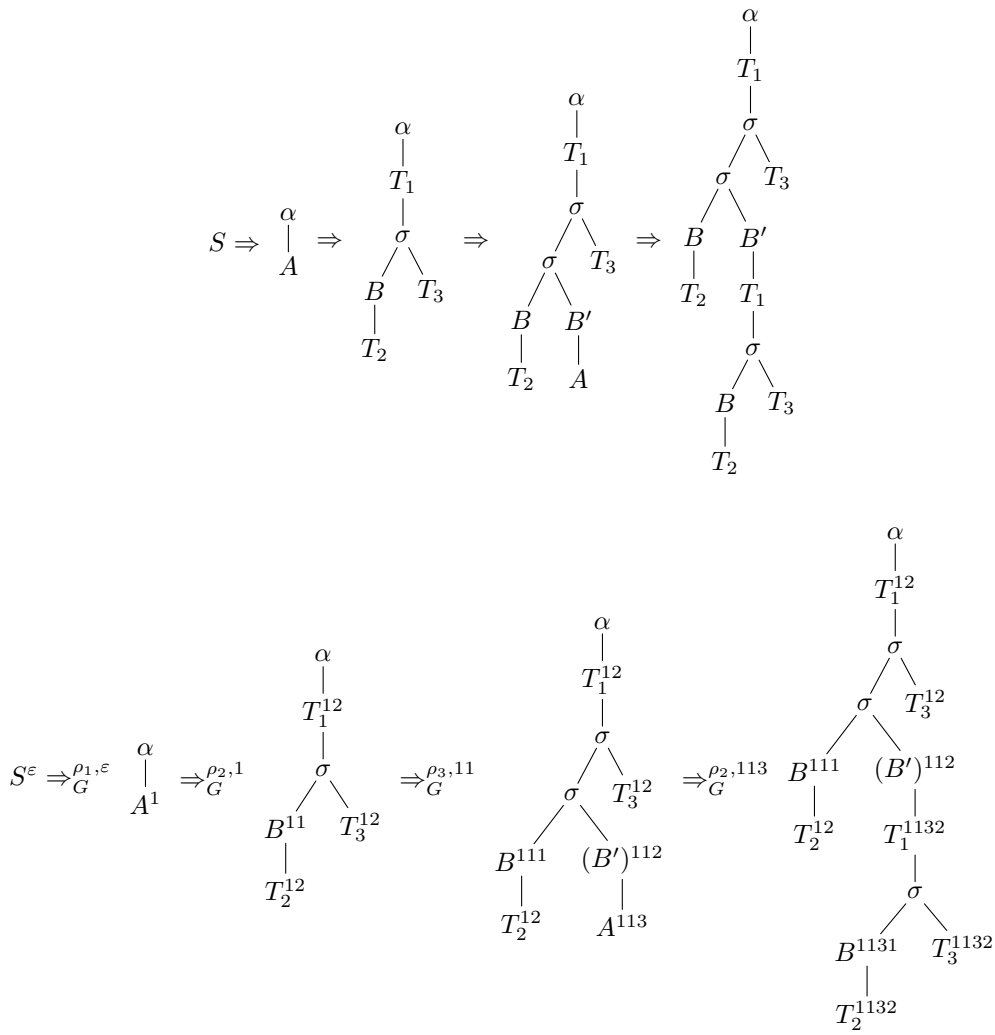


Figure 6: Derivation of the grammar G of Example 7; naive in the top part and as formalized in the bottom part.

Example 16 Let us consider the derivation tree $d = \rho_1(\rho_2(\rho_3(B, B', \rho_2(B, T)), T))$ of the grammar G of Examples 7 and 8, where $T = (T_1, T_2, T_3)$. Starting with S and successively applying the rules ρ_1, ρ_2, ρ_3, and ρ_2 according to the naive approach yields the derivation presented in the top part of Figure 6. It can be checked that the final tree in this derivation is $\mathrm{val}(d)$. However, now we are in trouble because B, T_1, T_2, and T_3 occur twice. With the help of the derivation steps as defined above and the shorthand C^ℓ for $\langle C, \ell\rangle$ with $C \in N$ and $\ell \in \mathbb{N}^*$ we obtain the derivation presented in the bottom part of Figure 6. In its final tree the occurrences B^{111} and B^{1131} of B can be rewritten independently, and the occurrences of T are distinguished as $T \otimes 12 = (T_1^{12}, T_2^{12}, T_3^{12})$ and $T \otimes 1132 = (T_1^{1132}, T_2^{1132}, T_3^{1132})$ and can be rewritten independently by ρ_5 or ρ_6. Note that $111 = (1, 1, 1)$ and $1131 = (1, 1, 3, 1)$ are the positions of d with label B, $12 = (1, 2)$ and $1132 = (1, 1, 3, 2)$ are the positions of d with label T, and $112 = (1, 1, 2)$ is the unique position of d with label B'. □

We wish to prove that $L(G) = \{t \in T_\Sigma \mid S \otimes \varepsilon \Rightarrow_G^* t\}$; note that $S \otimes \varepsilon = \langle S, \varepsilon\rangle$. To that end, we define an infinite MCFTG G^∞ using the properly annotated nonterminals and show that it is equivalent to G. An infinite MCFTG is defined as in Definition 5 except that N, $\mathcal{N}$, and R are allowed to be infinite (and similarly, in an infinite RTG N and R are allowed to be infinite). It is easy to check that all the definitions and results for MCFTGs discussed until now are also valid for infinite MCFTGs and infinite RTGs. In particular, the derivation tree grammar G^∞_{der} of G^∞ is infinite.

The infinite MCFTG is given by $G^\infty = (N^\infty, \mathcal{N}^\infty, \Sigma, S^\infty, R^\infty)$ with nonterminals $N^\infty = N \times \mathbb{N}^*$, big nonterminals $\mathcal{N}^\infty = \mathcal{N} \otimes \mathbb{N}^* = \{A \otimes \ell \mid A \in \mathcal{N}, \ell \in \mathbb{N}^*\}$, initial nonterminal $S^\infty = S \otimes \varepsilon$, and rules R^∞ determined as follows. If $\ell \in \mathbb{N}^*$ and $\rho = A \to (u, \mathcal{L}) \in R$ with $\mathcal{L} = \{B_1, \ldots, B_k\}$, then R^∞ contains the rule $\rho \otimes \ell = A \otimes \ell \to ((u, \mathcal{L}) \otimes \ell,\ \mathcal{L} \otimes \ell)$, where $\mathcal{L} \otimes \ell = \{B_1 \otimes \ell 1, \ldots, B_k \otimes \ell k\}$. Note that ρ can be reconstructed from $\rho \otimes \ell$.

Lemma 17 $L(G^\infty) = L(G)$.

Proof If d is a derivation tree of G^∞, then we denote by $\mathrm{rem}(d)$ the derivation tree of G that is obtained by removing all link identifiers ℓ from the labels of its nodes; i.e., a label $\rho \otimes \ell \in R^\infty$ is changed into ρ, and $A \otimes \ell \in \mathcal{N}^\infty$ is changed into A. It is straightforward to show by induction on the structure of d that $d \in L(G^\infty_{\mathrm{der}}, A \otimes \ell)$ implies both $\mathrm{rem}(d) \in L(G_{\mathrm{der}}, A)$ and $\mathrm{val}(\mathrm{rem}(d)) = \mathrm{val}(d)$. Indeed, if $d = (\rho \otimes \ell)(d_1, \ldots, d_k)$, then $\mathrm{rem}(d) = \rho(\mathrm{rem}(d_1), \ldots, \mathrm{rem}(d_k))$ and

$$\begin{aligned}
\mathrm{val}(d) &= ((u, \mathcal{L}) \otimes \ell)[B_i \otimes \ell i \leftarrow \mathrm{val}(d_i) \mid 1 \le i \le k] \\
&= u[B_i \leftarrow \mathrm{in}(B_i \otimes \ell i) \mid 1 \le i \le k]\,[B_i \otimes \ell i \leftarrow \mathrm{val}(d_i) \mid 1 \le i \le k] \\
&= u\big[B_i \leftarrow \mathrm{in}(B_i \otimes \ell i)[B_i \otimes \ell i \leftarrow \mathrm{val}(d_i) \mid 1 \le i \le k] \mid 1 \le i \le k\big] = u[B_i \leftarrow \mathrm{val}(d_i) \mid 1 \le i \le k] \\
&= u[B_i \leftarrow \mathrm{val}(\mathrm{rem}(d_i)) \mid 1 \le i \le k] = \mathrm{val}(\mathrm{rem}(d)) \ ,
\end{aligned}$$

where the third equality is by Lemma 4(4) and the fifth by the induction hypotheses. Taking $A \otimes \ell = S \otimes \varepsilon$, we thus obtain that $L(G^\infty) \subseteq L(G)$ by Theorem 9. In the other direction, we consider a derivation tree $d \in L(G_{\mathrm{der}}, S)$, and let d' be the tree such that $\mathrm{pos}(d') = \mathrm{pos}(d)$ and $d'(p) = d(p) \otimes p$ for every $p \in \mathrm{pos}(d)$; i.e., we change the label $d(p)$ of each position p into $d(p) \otimes p$. Obviously, $d' \in L(G^\infty_{\mathrm{der}}, S \otimes \varepsilon)$ and $\mathrm{rem}(d') = d$. Hence, by the above, d' has the same value as d, which shows that $L(G) \subseteq L(G^\infty)$. ■

Lemma 18 *Let $d \in DL(G^\infty_{\mathrm{der}}, S \otimes \varepsilon)$ and $A \otimes \ell \in \mathcal{N}^\infty$. Then $A \otimes \ell \in \mathrm{occ}_{\mathcal{N}^\infty}(d)$ if and only if* $\mathrm{occ}(A \otimes \ell) = \mathrm{occ}_{N \times \{\ell\}}(\mathrm{val}(d))$.

Proof We first observe that for every position $p \in \mathrm{pos}(d)$ there exists $\alpha \in \mathcal{N} \cup R$ such that $d(p) = \alpha \otimes p$, cf. the proof of Lemma 17. Thus, if $A \otimes \ell$ occurs in d then it occurs exactly once in d and no $B \otimes \ell$ occurs in d with $B \neq A$.

Let $A \otimes \ell \in \mathrm{occ}_{\mathcal{N}^\infty}(d)$. Then $\mathrm{occ}(A \otimes \ell) \subseteq \mathrm{occ}_{N \times \{\ell\}}(\mathrm{val}(d))$ by Lemma 10(2). Moreover, if $\langle C, \ell\rangle \in \mathrm{occ}_{N \times \{\ell\}}(\mathrm{val}(d))$ then there exists $B \in \mathcal{N}$ such that $C \in \mathrm{occ}(B)$ and $B \otimes \ell \in \mathrm{occ}_{\mathcal{N}^\infty}(d)$. From the above observation we obtain that $B = A$ and so $\langle C, \ell\rangle \in \mathrm{occ}(A \otimes \ell)$.

Now let $\mathrm{occ}(A \otimes \ell) = \mathrm{occ}_{N \times \{\ell\}}(\mathrm{val}(d))$. From the inclusion $\mathrm{occ}(A \otimes \ell) \subseteq \mathrm{occ}_{N \times \{\ell\}}(\mathrm{val}(d))$ we obtain, by Lemma 10(2) and the above observation, that there exists $B \in \mathcal{N}$ such that $B \otimes \ell \in \mathrm{occ}_{\mathcal{N}^\infty}(d)$ and $\mathrm{occ}(A \otimes \ell) \subseteq \mathrm{occ}(B \otimes \ell)$. Hence $\mathrm{occ}(A \otimes \ell) = \mathrm{occ}(B \otimes \ell)$ by the previous paragraph, and so $A = B$ by the second item of Definition 5. ■

Theorem 19 $L(G) = \{t \in T_\Sigma \mid S \otimes \varepsilon \Rightarrow_G^* t\}$.

Proof By Lemma 17, Theorem 9 and Lemma 10(3), it suffices to prove the following claim:

> For every $t \in T_{(N\times\mathbb{N}^*)\cup\Sigma}$ we have $S \otimes \varepsilon \Rightarrow_G^* t$ if and only if there exists $d \in DL(G^\infty_{\mathrm{der}}, S \otimes \varepsilon)$ such that $\mathrm{val}(d) = t$.

(If) The proof is by induction on the length n of a derivation $S\otimes\varepsilon \Rightarrow^n_{G^\infty_{\mathrm{der}}} d$ required for $d \in DL(G^\infty_{\mathrm{der}}, S\otimes\varepsilon)$. The claim is obvious for $n = 0$; i.e., for $d = S \otimes \varepsilon$. Otherwise, we consider the last step of the derivation $S\otimes\varepsilon \Rightarrow^{n-1}_{G^\infty_{\mathrm{der}}} d' \Rightarrow_{G^\infty_{\mathrm{der}}} d$, and let $A\otimes\ell \to (\rho\otimes\ell)(B_1\otimes\ell 1, \ldots, B_k\otimes\ell k)$ be the rule of G^∞_{der} that was applied in the last step, where $\rho = A \to (u, \mathcal{L})$ with $\mathcal{L} = \{B_1, \ldots, B_k\}$ is the corresponding rule of G. Clearly, since $A \otimes \ell$ occurs exactly once in d' (as observed in the proof of Lemma 18),

$$d = d'[A \otimes \ell \leftarrow (\rho \otimes \ell)(B_1 \otimes \ell 1, \ldots, B_k \otimes \ell k)] \ .$$

Since $\mathrm{val}((\rho \otimes \ell)(B_1 \otimes \ell 1, \ldots, B_k \otimes \ell k)) = (u, \mathcal{L}) \otimes \ell$, we obtain $\mathrm{val}(d) = \mathrm{val}(d')[A \otimes \ell \leftarrow (u, \mathcal{L}) \otimes \ell]$ from Lemma 11. Hence $S \otimes \varepsilon \Rightarrow_G^* \mathrm{val}(d') \Rightarrow_G^{\rho,\ell} \mathrm{val}(d)$ by the induction hypothesis, Lemma 18 and the definition of $\Rightarrow_G^{\rho,\ell}$.

(Only if) The proof is by induction on the length n of a derivation $S \otimes \varepsilon \Rightarrow_G^n t$. It is again obvious for $n = 0$. Otherwise, we consider the last step of the derivation $S \otimes \varepsilon \Rightarrow_G^{n-1} t' \Rightarrow_G t$. By the induction hypothesis there exists $d' \in DL(G^\infty_{\mathrm{der}}, S \otimes \varepsilon)$ such that $\mathrm{val}(d') = t'$. Moreover, by the definition of $\Rightarrow_G$, there exist a rule $\rho = A \to (u, \mathcal{L}) \in R$ and a link identifier ℓ such that $\mathrm{occ}(A \otimes \ell) = \mathrm{occ}_{N\times\{\ell\}}(t')$ and $t = t'[A \otimes \ell \leftarrow (u, \mathcal{L}) \otimes \ell]$. Then $A \otimes \ell$ occurs in d' by Lemma 18. Defining d as displayed above, we obtain from Lemma 11 that $\mathrm{val}(d) = \mathrm{val}(d')[A \otimes \ell \leftarrow (u, \mathcal{L}) \otimes \ell]$; i.e., $\mathrm{val}(d) = t$. ∎

In exactly the same way it can be proved that $L(G, A) = \{t \in P_\Sigma(X)^+ \mid \mathrm{in}(A \otimes \varepsilon) \Rightarrow_G^* t\}$ for every $A \in \mathcal{N}$, after extending the notion of derivation step to forests in $P_{(N\times\mathbb{N}^*)\cup\Sigma}(X)^+$. We finally mention that it is straightforward to prove that for every $t \in T_{(N\times\mathbb{N}^*)\cup\Sigma}$, if $S \otimes \varepsilon \Rightarrow_G^* t$, then (1) t is uniquely $(N \times \mathbb{N}^*)$-labeled and (2) there is a unique finite subset $\mathcal{L}$ of $\mathcal{N} \otimes \mathbb{N}^*$ such that the set $\{\mathrm{occ}(B) \mid B \in \mathcal{L}\}$ is equal to the set $\{\mathrm{occ}_{N\times\{\ell\}}(t) \neq \emptyset \mid \ell \in \mathbb{N}^*\}$. Thus, $\mathcal{L}$ is the set of big nonterminals (of G^∞) that can be rewritten in t. For instance, for the last tree of Figure 6 we have $\mathcal{L} = \{B \otimes 111, B \otimes 1131, B' \otimes 112, T \otimes 12, T \otimes 1132\}$.

4. Normal forms

In this section, we establish a number of normal forms for MCFTGs. We start in Section 4.1 with some basic normal forms. In Section 4.2 we define the notions of finite ambiguity and lexicalization, and then we prove a Growing Normal Form that is already part of our lexicalization procedure. Along the way we show the decidability of finite ambiguity. Finally we establish one additional basic normal form. From now on, let $G = (N, \mathcal{N}, \Sigma, S, R)$ be the considered MCFTG.

4.1. Basic normal forms

The MCFTG G is *start-separated* if $\mathrm{pos}_S(u) = \emptyset$ for every rule $A \to (u, \mathcal{L}) \in R$. In other words, the initial nonterminal S is not allowed in the right-hand sides of the rules. It is clear that G can be transformed into an $\mathrm{LDT}^{\mathrm{R}}$-equivalent start-separated MCFTG G'. We simply take a new initial nonterminal S', all original rules, and for every rule $\rho = S \to (u, \mathcal{L}) \in R$ we add the rule $\rho' = S' \to (u, \mathcal{L})$. Then we obviously have that $L(G'_{\mathrm{der}}, S') = \{\rho'(d_1, \ldots, d_k) \mid \rho(d_1, \ldots, d_k) \in L(G_{\mathrm{der}}, S)\}$, and there exist LDT-transducers that change $\rho(d_1, \ldots, d_k)$ into $\rho'(d_1, \ldots, d_k)$ and vice versa. The MCFTGs of Examples 6 and 7 are start-separated.

Convention. From now on, we assume, without loss of generality (by Proposition 13), and without mentioning it, that every MCFTG is start-separated. Each rule of the form $S \to (u, \mathcal{L})$ is called an *initial* rule. We call a rule $A \to (u, \mathcal{L})$ *terminal* if $u \in P_\Sigma(X)^+$; i.e., u does not contain nonterminal symbols or equivalently $\mathcal{L} = \emptyset$. Such a rule will also be written $A \to u$. Note that a rule may be both initial and terminal. A rule is called *proper* if it is not both initial and terminal.

The MCFTG G is *reduced* if every big nonterminal $A \in \mathcal{N} \setminus \{S\}$ is reachable and useful. A big nonterminal $A \in \mathcal{N}$ is *reachable* if $S \hookrightarrow_G^* A$, where for all $B, B' \in \mathcal{N}$ we define $B \hookrightarrow_G B'$ if there is a

rule $B \to (u, \mathcal{L}) \in R$ such that $B' \in \mathcal{L}$. Moreover, A is *useful* if $L(G, A) \neq \emptyset$. Clearly, G is reduced if and only if the RTG G_{der} is reduced (in the usual, analogous sense); this is obvious for reachability and follows from Theorem 9 for usefulness. As in the case of context-free grammars, we may and will always assume that a given MCFTG G is reduced, which can be achieved by removing all nonreachable and useless big nonterminals together with the rules in which they occur. Since this is the same procedure for G_{der}, we have that $L(G'_{\mathrm{der}}) = L(G_{\mathrm{der}})$ for the resulting grammar G', and hence, trivially, G' is $\mathrm{LDT}^{\mathrm{R}}$-equivalent to G. The MCFTGs of Examples 6 and 7 are reduced.

Let $G' = (N', \mathcal{N}', \Sigma, S', R')$ be another MCFTG. We say that G' is a *renaming* of G if there exists a rank-preserving bijection $\beta \colon \mathcal{N} \to \mathcal{N}'$ such that $S' = \beta(S)$ and $R' = \{\rho_\beta \mid \rho \in R\}$, where for every rule $\rho = A \to (u, \mathcal{L}) \in R$ we let $\rho_\beta = \beta(A) \to (u[B \leftarrow \mathrm{in}(\beta(B)) \mid B \in \mathcal{L}], \beta(\mathcal{L}))$, where $\beta(\mathcal{L}) = \{\beta(B_1), \ldots, \beta(B_k)\}$ if $\mathcal{L} = \{B_1, \ldots, B_k\}$. Note that ρ can easily be reconstructed from ρ_β (by applying β^{-1}); i.e., the mapping $\rho \mapsto \rho_\beta$ is also a bijection, from R to R'.

Lemma 20 *For all MCFTGs G and G', if G' is a renaming of G, then G and G' are* $\mathrm{LDT}^{\mathrm{R}}$*-equivalent.*

Proof Let β be the required bijection. For every tree $d \in T_R$, let $M(d)$ be obtained from d by changing every label ρ into ρ_β. In this manner we obtain a bijection $M \colon T_R \to T_{R'}$. Obviously, $d \in L(G_{\mathrm{der}}, A)$ if and only if $M(d) \in L(G'_{\mathrm{der}}, \beta(A))$. Additionally, we can easily show that $\mathrm{val}(M(d)) = \mathrm{val}(d)$ by induction on the structure of d. Indeed, let $d = \rho(d_1, \ldots, d_k)$ for a rule $\rho = A \to (u, \mathcal{L}) \in R$ with $\mathcal{L} = \{B_1, \ldots, B_k\}$ and $d_i \in L(G_{\mathrm{der}}, B_i)$ for every $i \in [k]$. We have $\mathrm{val}(M(d_i)) = \mathrm{val}(d_i)$ for every $i \in [k]$ by the induction hypotheses. Clearly, $M(d) = \rho_\beta(M(d_1), \ldots, M(d_k))$, and hence $\mathrm{val}(M(d)) = u[B_i \leftarrow \mathrm{in}(\beta(B_i)) \mid 1 \leq i \leq k]\,[f]$, where f is the substitution function for $\beta(\mathcal{L})$ such that $f(\beta(B_i)) = \mathrm{val}(M(d_i)) = \mathrm{val}(d_i)$ for every $i \in [k]$. It now follows from Lemma 4(4) that $\mathrm{val}(M(d)) = u[B_i \leftarrow \mathrm{in}(\beta(B_i)[f]) \mid 1 \leq i \leq k]$, which equals $u[B_i \leftarrow \mathrm{val}(d_i) \mid 1 \leq i \leq k] = \mathrm{val}(d)$. The transformation $M \colon T_R \to T_{R'}$ as well as its inverse $M^{-1} \colon T_{R'} \to T_R$ are tree homomorphisms (even projections), and every tree homomorphism can be realized by an $\mathrm{LDT}^{\mathrm{R}}$-transducer, which shows the $\mathrm{LDT}^{\mathrm{R}}$-equivalence. ∎

The previous lemma shows that the actual identity of nonterminals constituting a big nonterminal is irrelevant in MCFTGs. We say that the MCFTG G *has disjoint big nonterminals* if $\mathrm{occ}(A) \cap \mathrm{occ}(A') = \emptyset$ for all distinct $A, A' \in \mathcal{N}$. The MCFTGs of Examples 6 and 7 indeed have disjoint big nonterminals. Clearly, every MCFTG G has a renaming that has disjoint big nonterminals. Consequently, we may always assume that a given MCFTG G has disjoint big nonterminals. As observed before Example 6, the specification of the set of links of a rule is then no longer necessary. Indeed we could have required disjoint big nonterminals in Definition 5, but this would have been technically inconvenient, as we will see, e.g., in the proof of Lemma 22.

We say that the MCFTG G is *free-choice* if the following holds. For every rule $A \to (u, \mathcal{L}) \in R$ and every $\mathcal{L}' \subseteq \mathcal{N}$ that satisfies the requirement in the last item of Definition 5, we require that $A \to (u, \mathcal{L}')$ is also a rule of G. This means that the rules of G can be specified as $A \to u$, which stands for all possible rules $A \to (u, \mathcal{L})$. Obviously, if G has disjoint big nonterminals, then it is free-choice because the links are uniquely determined by $\mathcal{N}$ and u. Thus, we may always assume that a given MCFTG is free-choice. Free-choice MCFTGs with the derivation semantics of Section 3.3 generalize the local unordered scattered context grammars (LUSCGs) of [78], which are an equivalent formulation of multiple context-free (string) grammars.

The next easy result is not a normal form result in the usual sense of the word, but shows that the class MCFT is closed under (simple) tree homomorphisms; for much stronger closure properties of MCFT we refer to Section 8. Nevertheless, a special case of this result can be used in proofs to assume that the right-hand sides of a given MCFTG G are not only uniquely N-labeled but also uniquely Σ-labeled.

Let h be a tree homomorphism from Σ to Σ' where Σ' is a finite ranked alphabet disjoint to N. We define the MCFTG $G_h = (N, \mathcal{N}, \Sigma', S, R')$ such that

$$R' = \{A \to (\hat{h}(u), \mathcal{L}) \mid A \to (u, \mathcal{L}) \in R\} \ ,$$

where h is extended to a tree homomorphism from $N \cup \Sigma$ to $N \cup \Sigma'$ by defining $h(C) = \mathrm{in}(C)$ for every $C \in N$. We refer to Definition 15 for the notion of $\mathrm{LDT}^{\mathrm{R}}$-$\hat{h}$-equivalence.

Lemma 21 *For every MCFTG G and every tree homomorphism h (as above), the MCFTG G_h (as defined above) is* $\mathrm{LDT}^{\mathrm{R}}$*-$\hat{h}$-equivalent to G. Hence $L(G_h) = \hat{h}(L(G))$.*

Proof The proof is similar to the one of Lemma 20. Let $G' = G_h = (N, \mathcal{N}, \Sigma', S, R')$. For every rule $\rho = A \to (u, \mathcal{L}) \in R$, let ρ_h be the rule $A \to (\hat{h}(u), \mathcal{L}) \in R'$, in which the links of $\mathcal{L}$ have the same order as in ρ. For every tree $d \in T_R$, let $M(d)$ be obtained from d by changing every label ρ into ρ_h. This defines a surjection $M\colon T_R \to T_{R'}$. Obviously, $d \in L(G_{\mathrm{der}}, A)$ if and only if $M(d) \in L(G'_{\mathrm{der}}, A)$ for every $A \in \mathcal{N}$. We now show, by induction on the structure of d, that $\mathrm{val}(M(d)) = \hat{h}(\mathrm{val}(d))$. Indeed, let $d = \rho(d_1, \ldots, d_k)$ with $\rho = A \to (u, \mathcal{L})$ and $\mathcal{L} = \{B_1, \ldots, B_k\}$, and by the induction hypotheses $\mathrm{val}(M(d_i)) = \hat{h}(\mathrm{val}(d_i))$ for every $i \in [k]$. Then $M(d) = \rho_h(M(d_1), \ldots, M(d_k))$, and hence we have

$$\begin{aligned}\mathrm{val}(M(d)) &= \hat{h}(u)[B_i \leftarrow \hat{h}(\mathrm{val}(d_i)) \mid 1 \le i \le k] \\ &= \hat{h}(u[B_i \leftarrow \mathrm{val}(d_i) \mid 1 \le i \le k]) = \hat{h}(\mathrm{val}(d)) \ ,\end{aligned}$$

where the second equality is by Lemma 4(3) applied to $\sigma_1 = B_1 \cdots B_k$ and $\mathrm{occ}(\sigma_2) = \Sigma$. This shows that $\hat{h}(L(G)) \subseteq L(G')$.

For every rule $\rho' \in R'$, let ρ'_h be a fixed rule $\rho \in R$ such that $\rho_h = \rho'$. For every tree $d' \in T_{R'}$, let $M'(d')$ be obtained from d' by changing every label ρ' into ρ'_h. This defines a mapping $M'\colon T_{R'} \to T_R$. Obviously $M(M'(d')) = d'$ and hence, by the above, if $d' \in L(G'_{\mathrm{der}}, A)$ then $M'(d') \in L(G_{\mathrm{der}}, A)$ and $\mathrm{val}(M'(d')) = \mathrm{val}(d')$. This shows that $L(G') \subseteq \hat{h}(L(G))$.

The transformations M and M' can be realized by projections, and hence by $\mathrm{LDT}^{\mathrm{R}}$-transducers. ∎

We say that the pair (G, h) is a *cover* of the MCFTG G_h if h is a projection; i.e., for every $\sigma \in \Sigma$ there exists $\sigma' \in \Sigma'$ such that $h(\sigma) = \mathrm{in}(\sigma')$. We define the MCFTG G to be *uniquely terminal labeled* if for every rule $\rho \in R$:

(1) the right-hand side $\mathrm{rhs}(\rho)$ is uniquely Σ-labeled, and

(2) $\mathrm{occ}_\Sigma(\mathrm{rhs}(\rho)) \cap \mathrm{occ}_\Sigma(\mathrm{rhs}(\rho')) = \emptyset$ for every other rule $\rho' \in R$.

Clearly, every MCFTG G has a cover (G_{u}, h) such that G_{u} is uniquely terminal labeled. Although the tree languages $L(G) = \hat{h}(L(G_{\mathrm{u}}))$ and $L(G_{\mathrm{u}})$ differ in general, this may be viewed as a normal form of G.

The last basic normal form that we consider in this subsection is permutation-freeness. Let Ω be a ranked alphabet (such as $N \cup \Sigma$). For a tree $t \in T_\Omega(X)$ the string $\mathrm{yd}_X(t) \in X^*$ is the sequence of occurrences of variables in t, from left to right.[15] Clearly, if $t \in P_\Omega(X_k)$, then $\mathrm{yd}_X(t)$ is a permutation $x_{i_1} \cdots x_{i_k}$ of $x_1 \cdots x_k$. We say that a pattern $t \in P_\Omega(X)$ is *permutation-free* if $\mathrm{yd}_X(t) = x_1 \cdots x_k$ for $k = \mathrm{rk}(t)$, and we denote the set of permutation-free patterns over Ω by $PF_\Omega(X)$. For $t \in P_\Omega(X)$ we define $\mathrm{pf}(t) \in PF_\Omega(X)$ as follows: if $\mathrm{yd}_X(t) = x_{i_1} \cdots x_{i_k}$, then $\mathrm{pf}(t)$ is the unique permutation-free pattern such that $t = \mathrm{pf}(t)[x_1 \leftarrow x_{i_1}, \ldots, x_k \leftarrow x_{i_k}]$. For a forest $t = (t_1, \ldots, t_n)$ we define $\mathrm{yd}^*_X(t) = (\mathrm{yd}_X(t_1), \ldots, \mathrm{yd}_X(t_n))$ and $\mathrm{pf}^*(t) = (\mathrm{pf}(t_1), \ldots, \mathrm{pf}(t_n))$. We say that a tree homomorphism h over Ω is permutation-free if $h(\omega)$ is permutation-free for every $\omega \in \Omega$. We observe that, for such a tree homomorphism, $\mathrm{yd}_X(\hat{h}(t)) = \mathrm{yd}_X(t)$ for every $t \in T_\Omega(X)$, as can easily be shown by induction on the structure of t, and $\hat{h}(\mathrm{pf}(t)) = \mathrm{pf}(\hat{h}(t))$ for every $t \in P_\Omega(X)$ by Lemma 2.

The MCFTG G is *permutation-free* if $\mathrm{rhs}(\rho) \in PF_{N \cup \Sigma}(X)^+$ for every rule $\rho \in R$. Intuitively, permutation-free MCFTGs are easier to understand than arbitrary MCFTGs because the application of a rule to a node of a tree does not involve a permutation of the subtrees at the children of that node; thus, a rule application does not affect the global structure of the tree. The MCFTG G of Example 7 is trivially permutation-free because every nonterminal of G has rank 0 or 1.

Lemma 22 *For every MCFTG G there is an* $\mathrm{LDT}^{\mathrm{R}}$*-equivalent MCFTG G' that is permutation-free. Moreover, $\theta(G') = \theta(G)$, $\mu(G') = \mu(G)$, and $\lambda(G') = \lambda(G)$.*

Proof We construct the grammar $G' = (N', \mathcal{N}', \Sigma, S', R')$, in which $S' = \langle S, \varepsilon \rangle$ and N' is the set of all pairs $\langle C, \pi \rangle$ such that $C \in N$ and π is a permutation of $x_1 \cdots x_{\mathrm{rk}(C)}$. The rank of $\langle C, \pi \rangle$ is the same as the rank of C. The set of big nonterminals $\mathcal{N}'$ consists of all $(\langle A_1, \pi_1 \rangle, \ldots, \langle A_n, \pi_n \rangle)$ with $(A_1, \ldots, A_n) \in \mathcal{N}$ and $\langle A_i, \pi_i \rangle \in N'$ for every $i \in [n]$.[16] A big nonterminal $A' = (\langle A_1, \pi_1 \rangle, \ldots, \langle A_n, \pi_n \rangle)$

[15] The yield of t with respect to X is defined in the paragraph on homomorphisms in Section 2.1.

[16] Note that if G has disjoint big nonterminals, then that is in general not the case for G'. Thus, this property of an MCFTG G is not preserved when information is added to the nonterminals of G, which is the reason that we did not require it in Definition 5.

will also be denoted by $\mathrm{pair}(A,\pi)$, where $A = (A_1,\ldots,A_n)$ and $\pi = (\pi_1,\ldots,\pi_n)$, and we define $\mathrm{rem}(A') = A = (A_1,\ldots,A_n)$. Intuitively, if A generates $t = (t_1,\ldots,t_n)$ with $t_i \in P_\Sigma(X_{\mathrm{rk}(A_i)})$ and $\mathrm{yd}^*_X(t) = (\mathrm{yd}_X(t_1),\ldots,\mathrm{yd}_X(t_n)) = (\pi_1,\ldots,\pi_n)$, then A' generates $\mathrm{pf}^*(t) = (\mathrm{pf}(t_1),\ldots,\mathrm{pf}(t_n))$. To define the rules of G' we need the (permuting) tree homomorphism h over $N' \cup \Sigma$ that is defined by $h(\langle C,\pi\rangle) = \langle C,\pi\rangle\pi$ for every $\langle C,\pi\rangle \in N'$ and $h(\sigma) = \mathrm{in}(\sigma)$ for every $\sigma \in \Sigma$. For example, if $\pi = x_3x_2x_1x_4$, then $h(\langle C,\pi\rangle) = \langle C,\pi\rangle(x_3,x_2,x_1,x_4)$; in other words, h permutes the subtrees of $\langle C,\pi\rangle$ according to the permutation π.

Let $\rho = A \to (u,\mathcal{L})$ be a rule of G with $\mathcal{L} = \{B_1,\ldots,B_k\}$. Moreover, let $B'_1,\ldots,B'_k$ be big nonterminals in $\mathcal{N}'$ such that $\mathrm{rem}(B'_i) = B_i$ for every $i \in [k]$, and let $u' = u[B_i \leftarrow \mathrm{in}(B'_i) \mid 1 \le i \le k]$ and $\overline{\pi} = \mathrm{yd}^*_X(\hat{h}(u'))$. Then R' contains the rule

$$\rho_{B'_1\cdots B'_k} = \mathrm{pair}(A,\overline{\pi}) \to (\mathrm{pf}^*(\hat{h}(u')),\ \{B'_1,\ldots,B'_k\}) \ .$$

Note that this rule satisfies the requirements of Definition 5 by Lemma 1. Note also that ρ can be reconstructed from $\rho_{B'_1\cdots B'_k}$. This completes the construction of G'.

To show that $L(G) \subseteq L(G')$ we prove that for every $A \in \mathcal{N}$ and every derivation tree $d \in L(G_{\mathrm{der}},A)$ there exists a derivation tree $d' \in L(G'_{\mathrm{der}},\mathrm{pair}(A,\pi))$ such that $\pi = \mathrm{yd}^*_X(\mathrm{val}(d))$ and $\mathrm{val}(d') = \mathrm{pf}^*(\mathrm{val}(d))$. For every derivation tree $d \in \bigcup_{B\in\mathcal{N}} L(G_{\mathrm{der}},B)$, we let $\mathrm{bign}(d) = \mathrm{pair}(A,\mathrm{yd}^*_X(\mathrm{val}(d)))$, where A is the type of d. The proof is by induction on the structure of d. Simultaneously we prove that $\mathrm{bign}(d)$ can be defined inductively. Let $d = \rho(d_1,\ldots,d_k)$, where ρ is as shown above. By the induction hypotheses, let $B'_i = \mathrm{bign}(d_i) = \mathrm{pair}(B_i,\pi_i)$ such that $\pi_i = \mathrm{yd}^*_X(\mathrm{val}(d_i))$, and let $d'_i \in L(G'_{\mathrm{der}},B'_i)$ be such that $\mathrm{val}(d'_i) = \mathrm{pf}^*(\mathrm{val}(d_i))$, for every $i \in [k]$. We define $\mathrm{bign}(d)$ to be the left-hand side of the rule $\rho_{B'_1\cdots B'_k}$. Moreover, we take $d' = \rho_{B'_1\cdots B'_k}(d'_1,\ldots,d'_k)$. Additionally, let $[g'_{\mathrm{pf}}]$ abbreviate the (simultaneous) second-order substitution $[B'_i \leftarrow \mathrm{pf}^*(\mathrm{val}(d_i)) \mid 1 \le i \le k]$, and let $[g']$ and $[g]$ abbreviate the second-order substitutions $[B'_i \leftarrow \mathrm{val}(d_i) \mid 1 \le i \le k]$ and $[B_i \leftarrow \mathrm{val}(d_i) \mid 1 \le i \le k]$. Then the definition of 'val' gives $\mathrm{val}(d') = \mathrm{pf}^*(\hat{h}(u'))[B'_i \leftarrow \mathrm{val}(d'_i) \mid 1 \le i \le k] = \mathrm{pf}^*(\hat{h}(u'))[g'_{\mathrm{pf}}] = \mathrm{pf}^*(\hat{h}(u')[g'_{\mathrm{pf}}])$, where the last equality holds because the permutation-free tree homomorphism g'_{pf} corresponding to the substitution $[g'_{\mathrm{pf}}]$ commutes with 'pf' as observed before this lemma. We now show that

$$\hat{h}(u')[g'_{\mathrm{pf}}] = u'[g'] = u[g] = \mathrm{val}(d) \ .$$

The first equality holds by Lemma 3 because the composition of the tree homomorphisms h and $\hat{g}'_{\mathrm{pf}}$ is equal to the tree homomorphism g' corresponding to the substitution $[g']$ for every symbol in $\mathrm{occ}_{N'\cup\Sigma}(u')$, as shown next. In fact, let $B'_i = \beta\langle C,\pi\rangle\gamma$ with $\langle C,\pi\rangle \in N'$ and $\beta,\gamma \in (N')^*$, and let $\mathrm{val}(d_i) = \varphi t\psi$ with $t \in P_\Sigma(X_{\mathrm{rk}(C)})$, $\varphi,\psi \in P_\Sigma(X)^*$, and $|\beta| = |\varphi|$. From $\pi_i = \mathrm{yd}^*_X(\mathrm{val}(d_i))$, we obtain that $\pi = \mathrm{yd}_X(t)$. Now we have $g'_{\mathrm{pf}}(\langle C,\pi\rangle) = \mathrm{pf}(t)$ and therefore $\hat{g}'_{\mathrm{pf}}(h(\langle C,\pi\rangle)) = \hat{g}'_{\mathrm{pf}}(\langle C,\pi\rangle\pi) = t = g'(\langle C,\pi\rangle)$.[17] The second equality follows easily from Lemma 4(4), and the last equality is again by the definition of 'val'. Hence, we have shown that $\mathrm{val}(d') = \mathrm{pf}^*(\mathrm{val}(d))$, and it remains to show that the permutation $\overline{\pi}$ in the left-hand side of $\rho_{B'_1\cdots B'_k}$ fulfills $\overline{\pi} = \mathrm{yd}^*_X(\mathrm{val}(d))$. By the calculation above, $\mathrm{yd}^*_X(\mathrm{val}(d)) = \mathrm{yd}^*_X(\hat{h}(u')[g'_{\mathrm{pf}}])$. In addition, $\overline{\pi} = \mathrm{yd}^*_X(\hat{h}(u'))$ by the definition of $\rho_{B'_1\cdots B'_k}$. Since g'_{pf} is permutation-free, these values are the same, as observed before this lemma. This proves that $L(G) \subseteq L(G')$.

It is easy to see that the above transformation from d to d' can be realized by an $\mathrm{LDT}^{\mathrm{R}}$-transducer M with one state q. In fact, it should be clear from the inductive definition of $\mathrm{bign}(d)$ that the set $L_{A'} = \{d \in \bigcup_{B\in\mathcal{N}} L(G_{\mathrm{der}},B) \mid \mathrm{bign}(d) = A'\}$ is a regular tree language for every $A' \in \mathcal{N}'$. Then, for the above rule ρ, the transducer M has all the rules

$$\langle q,\ \rho(y_1\colon L_{B'_1},\ldots,y_k\colon L_{B'_k})\rangle \to \rho_{B'_1\cdots B'_k}(\langle q,y_1\rangle,\ldots,\langle q,y_k\rangle) \ .$$

Note that M is a finite-state relabeling.

To show that $L(G') \subseteq L(G)$, we observe that for every derivation tree $d' \in L(G'_{\mathrm{der}})$ the derivation tree $d \in L(G_{\mathrm{der}})$, which is obtained from d' by changing every label $\rho_{B'_1\cdots B'_k}$ into ρ, satisfies $M(d) = d'$ and hence $\mathrm{val}(d) = \mathrm{val}(d')$. Since this transformation from d' to d is a projection, it can be realized by an LDT-transducer. ∎

[17] To be precise, if $\mathrm{yd}_X(t) = \pi = x_{i_1}\cdots x_{i_m}$, then $\hat{g}'_{\mathrm{pf}}(\langle C,\pi\rangle\pi) = \mathrm{pf}(t)[x_1 \leftarrow x_{i_1},\ldots,x_m \leftarrow x_{i_m}] = t$.

4.2. Lexical normal forms

We first recall the notion of finite ambiguity from [50, 65, 85].[18] We distinguish a subset $\Delta \subseteq \Sigma$ of *lexical* symbols, which are the symbols that are preserved by the lexical yield mapping. The *lexical yield* of a tree $t \in T_\Sigma$ is the string $\mathrm{yd}_\Delta(t) \in \Delta^*$, as defined in Section 2.1. It is the string of occurrences of lexical symbols in t, from left to right; all other symbols are simply dropped.

Definition 23 The tree language $L \subseteq T_\Sigma$ has *finite Δ-ambiguity* if $\{t \in L \mid \mathrm{yd}_\Delta(t) = w\}$ is finite for every $w \in \Delta^*$. The MCFTG G has finite Δ-ambiguity if $L(G)$ has finite Δ-ambiguity. □

Roughly speaking, we can say that the language L has finite Δ-ambiguity if each $w \in \Delta^*$ has finitely many syntactic trees in L, where t is a syntactic tree of w if w is its lexical yield. Note that $|\mathrm{yd}_\Delta(t)| = |\mathrm{pos}_\Delta(t)|$; thus, L has finite Δ-ambiguity if and only if $\{t \in L \mid |\mathrm{pos}_\Delta(t)| = n\}$ is finite for every $n \in \mathbb{N}_0$. Note also that if $\Sigma^{(0)} \cup \Sigma^{(1)} \subseteq \Delta$ or $\Sigma \setminus \Sigma^{(0)} \subseteq \Delta$, then every tree language $L \subseteq T_\Sigma$ has finite Δ-ambiguity.

Example 24 For the MCFTG G of Example 7 we consider the set $\Delta = \Sigma \setminus \{\sigma, \gamma\} = \{\alpha, \beta, \tau, \nu\}$ of lexical symbols. It should be clear from Example 7 that in each tree of $L(G)$ the number of occurrences of γ coincides with the number of occurrences of β. Since $\Delta \cup \{\gamma\} = \Sigma^{(0)} \cup \Sigma^{(1)}$, this implies that $L(G)$ as well as G have finite Δ-ambiguity. Similarly, the number of occurrences of ν in a tree of $L(G)$ coincides with the number of occurrences of τ, and the number of occurrences of β is half the number of occurrences of α. Hence G also has finite $\{\alpha, \tau\}$-ambiguity, but for convenience we will continue to use the lexical symbols Δ in examples. □

In this contribution, we want to lexicalize MCFTGs, which means that for each MCFTG G that has finite Δ-ambiguity, we want to construct an equivalent MCFTG G' such that each proper rule[19] contains at least one lexical symbol. Let us formalize our lexicalization property.

Definition 25 The forest t is *Δ-lexicalized* if $\mathrm{pos}_\Delta(t) \neq \emptyset$. The rule $A \to (u, \mathcal{L})$ is Δ-lexicalized if u is Δ-lexicalized. The MCFTG G is Δ-lexicalized if all its proper rules are Δ-lexicalized. A forest or rule is *Δ-free* if it is not Δ-lexicalized. The rule $A \to (u, \mathcal{L})$ is *doubly* Δ-lexicalized if $|\mathrm{pos}_\Delta(u)| \geq 2$, and it is *singly* Δ-lexicalized if $|\mathrm{pos}_\Delta(u)| = 1$. □

Clearly, for every derivation tree d, the value $\mathrm{val}(d)$ is Δ-free if and only if all rules that occur in d are Δ-free by Lemma 10(1). For the grammar G of Example 7 with $\Delta = \{\alpha, \beta, \tau, \nu\}$ as in Example 24, the rules

$$\rho_1 = S \to \alpha(A) \quad \rho_5 = (T_1(x_1), T_2, T_3) \to (\alpha(T_1(\beta(x_1))), \alpha(T_2), \gamma(T_3)) \quad \rho_6 = (T_1(x_1), T_2, T_3) \to (x_1, \tau, \nu)$$

are Δ-lexicalized (ρ_1 singly and both ρ_5 and ρ_6 doubly), whereas rule $\rho_4 = B(x_1) \to x_1$ is not even Σ-lexicalized.

Thus, for each MCFTG G that has finite Δ-ambiguity, we want to construct an equivalent MCFTG G' that is Δ-lexicalized. This notion of lexicalization is also called strong lexicalization [50, 65, 85] because it requires strong equivalence of G and G'; i.e., $L(G') = L(G)$. Weak lexicalization [50] just requires weak equivalence of G and G'; i.e., $\mathrm{yd}_\Delta(L(G')) = \mathrm{yd}_\Delta(L(G))$. Clearly, with slight adaptations, these definitions can be applied to any type of context-free-like grammar that has terminal (ranked or unranked) alphabet Σ. In the literature only two cases are considered: $\Delta = \Sigma$ for unranked alphabets and $\Delta = \Sigma^{(0)} \setminus \{e\}$ for ranked alphabets. It seems to be quite natural and relevant to consider arbitrary Δ.

It should be intuitively clear (and will be shown below) that an MCFTG that does not have finite Δ-ambiguity cannot be lexicalized (with respect to Δ). Thus, we will prove that an MCFTG can be lexicalized (with respect to Δ) if and only if it has finite Δ-ambiguity. Moreover, we will prove that this property is decidable.

To lexicalize an MCFTG of finite ambiguity, we need an auxiliary normal form (stated in Theorem 37). It generalizes the Growing Normal Form of [89, 90] for spCFTGs. In the remainder of this section the MCFTG $G = (N, \mathcal{N}, \Sigma, S, R)$ is not assumed to have finite Δ-ambiguity unless this is explicitly mentioned. We only assume that G is start-separated and reduced. A rule ρ is *monic* if $|\mathcal{L}(\rho)| = 1$; i.e., $\mathcal{L}(\rho)$ is a singleton or equivalently ρ has rank 1 in G_{der}.

[18] It should not be confused with the notion of finite ambiguity of [43, 62].

[19] Recall from the beginning of Section 4.1 that a rule is proper if it is not both initial and terminal.

Definition 26 The MCFTG G is Δ-*growing* if all its non-initial terminal rules are doubly Δ-lexicalized, and all its monic rules are Δ-lexicalized. It is *almost* Δ-*growing* if all its non-initial terminal rules and all its monic rules are Δ-lexicalized. □

The application of a proper rule of a Δ-growing MCFTG increases the sum of the number of occurrences of lexical symbols and the number of occurrences of big nonterminals. In this section we will prove that for every MCFTG G of finite Δ-ambiguity there is an equivalent Δ-growing MCFTG (see Theorem 37). The instance of this result for spCFTGs and $\Delta = \Sigma$ is due to [90, Proposition 2] and fully proved in [89]. Note that if G is almost Σ-growing, then *all* its terminal rules are Σ-lexicalized. Note also that every Δ-growing MCFTG is almost Δ-growing, and that every Δ-lexicalized MCFTG is almost Δ-growing. The grammar G of Example 7 with $\Delta = \{\alpha, \beta, \tau, \nu\}$ as in Example 24 is *not* almost Δ-growing because of rule $\rho_4 = B(x_1) \to x_1$.

If the MCFTG G is almost Δ-growing, then all its rules satisfy the requirements for a Δ-growing grammar except the non-initial terminal rules, which might be singly Δ-lexicalized. The application of such a rule does not change the sum of the number of occurrences of lexical symbols and the number of occurrences of big nonterminals because a big nonterminal is replaced by a lexical symbol. This leads to the following lemma.

Lemma 27 *If G is almost Δ-growing, then G has finite Δ-ambiguity and*

$$|\mathrm{pos}(d)| \leq 2 \cdot (|\mathrm{pos}_\Delta(\mathrm{val}(d))| + |\mathrm{pos}_\mathcal{N}(d)|) + 1 \leq 2 \cdot |\mathrm{pos}_{N \cup \Delta}(\mathrm{val}(d))| + 1 \qquad (\dagger)$$

for every derivation tree d of G; i.e., for every $d \in \bigcup_{A \in \mathcal{N}} DL(G_{\mathrm{der}}, A)$.

Proof We begin with (†). Let R_{it} be the set of all initial terminal rules. The first inequality is clearly fulfilled for $d \in R_{\mathrm{it}}$, and it suffices to show that $|\mathrm{pos}(d)| + 1 \leq 2 \cdot (|\mathrm{pos}_\Delta(\mathrm{val}(d))| + |\mathrm{pos}_\mathcal{N}(d)|)$ for the remaining derivation trees $d \notin R_{\mathrm{it}}$. For every such tree d we have

$$|\mathrm{pos}(d)| + 1 \leq 2 \cdot \Big(|\mathrm{pos}_\mathcal{N}(d)| + |\mathrm{pos}_{R^{(0)}}(d)| + |\mathrm{pos}_{R^{(1)}}(d)| \Big) \ ,$$

where $R^{(0)}$ and $R^{(1)}$ are the sets of terminal and monic rules, respectively (see Section 2.2). Since G is almost Δ-growing and $\mathrm{pos}_{R_{\mathrm{it}}}(d) = \emptyset$, we obtain

$$|\mathrm{pos}_{R^{(0)}}(d)| + |\mathrm{pos}_{R^{(1)}}(d)| \leq \sum_{p \in \mathrm{pos}_R(d)} |\mathrm{pos}_\Delta(\mathrm{rhs}(d(p)))| = |\mathrm{pos}_\Delta(\mathrm{val}(d))| \ ,$$

where the last equality holds by Lemma 10(1). The second inequality in (†) follows from the first because $|\mathrm{pos}_\mathcal{N}(d)| \leq |\mathrm{pos}_N(\mathrm{val}(d))|$ by Lemma 10(2).

For the first part of the statement, we consider the set $L_w = \{t \in L(G) \mid \mathrm{yd}_\Delta(t) = w\}$ for some $w \in \Delta^*$. For every derivation tree $d \in L(G_{\mathrm{der}})$ we have $\mathrm{pos}_\mathcal{N}(d) = \emptyset$, and consequently we obtain $|\mathrm{pos}_\Delta(\mathrm{val}(d))| + |\mathrm{pos}_\mathcal{N}(d)| = |\mathrm{yd}_\Delta(\mathrm{val}(d))|$. Hence $|\mathrm{pos}(d)| \leq 2 \cdot |w| + 1$ if $\mathrm{val}(d) \in L_w$, utilizing (†). This shows that $D_w = \{d \in L(G_{\mathrm{der}}) \mid \mathrm{val}(d) \in L_w\}$ is finite, and so L_w is finite because $L_w = \mathrm{val}(D_w)$ by Theorem 9. ■

The previous result also shows that if G does not have finite Δ-ambiguity, then there is no Δ-lexicalized MCFTG equivalent to G, as we observed above.

Our first goal (in proving Theorem 37) is to make sure that all the non-initial terminal rules are Δ-lexicalized; i.e., contain a lexical symbol. However, for later use, we start by proving a more general lemma that will allow us to remove every non-initial terminal rule of which the right-hand side has a certain property $\mathcal{F}$ subject to certain requirements. In particular, the value of a derivation tree d has property $\mathcal{F}$ if and only if d only contains rules of a corresponding subset $F \subseteq R$ of rules. Additionally, each big nonterminal can only generate finitely many forests with property $\mathcal{F}$. An example of such a property is Σ-freeness. The next construction generalizes the removal of epsilon-rules $A \to \varepsilon$ from a context-free grammar [48].

Lemma 28 *Let $\mathcal{F} \subseteq P_\Sigma(X)^+$ and $F \subseteq R$. If*

(1) *$L(G,A) \cap \mathcal{F}$ is finite for every $A \in \mathcal{N}$, and*

(2) *$\mathrm{val}(d) \in \mathcal{F}$ if and only if $d \in T_F$, for every $d \in \bigcup_{A \in \mathcal{N}} L(G_{\mathrm{der}}, A)$,*

then there is an $\mathrm{LDT}^{\mathrm{R}}$-equivalent MCFTG $G' = (N, \mathcal{N}, \Sigma, S, R')$ such that $\mathrm{rhs}(\rho) \notin \mathcal{F}$ for every non-initial terminal rule $\rho \in R'$.

Proof For the effectiveness of the constructions in this proof, we assume that $\mathcal{F}$ is a decidable subset of $P_\Sigma(X)^+$, and that the elements of $L(G,A) \cap \mathcal{F}$ are effectively given for every $A \in \mathcal{N}$. For $A \in \mathcal{N}$, let $\mathcal{F}_A = L(G,A) \cap \mathcal{F}$, which is finite by (1). Moreover, $\mathcal{F}_A = \mathrm{val}(L(G_{\mathrm{der}}, A) \cap T_F)$ by (2) and Theorem 9. For every $A \in \mathcal{N}$ and $t \in \mathcal{F}_A$, let $L_{\langle A,t \rangle} = \{d \in L(G_{\mathrm{der}}, A) \cap T_F \mid \mathrm{val}(d) = t\}$. By Lemma 12 applied with $\mathcal{N}' = \emptyset$, the tree language $L_{\langle A,t \rangle}$ is regular.

We now construct the MCFTG $G' = (N, \mathcal{N}, \Sigma, S, R')$. The rule $\rho_{S,t} = S \to t$ is in R' for every $t \in \mathcal{F}_S$. Moreover, for every rule $\rho = A \to (u, \mathcal{L})$ of G and every substitution function f for $\mathcal{L}$ such that $f(B) \in \mathcal{F}_B \cup \{\mathrm{in}(B)\}$ for every $B \in \mathcal{L}$, the set R' contains the rule

$$\rho_f = A \to (u[f], \{B \in \mathcal{L} \mid f(B) = \mathrm{in}(B)\}) \ ,$$

provided that $u[f] \notin \mathcal{F}$. The linear order on $\mathcal{L}(\rho_f)$ is inherited from the one on $\mathcal{L}$. To be precise, let $\mathcal{L} = \{B_1, \ldots, B_k\}$ and $\Phi = \{i \in [k] \mid f(B_i) \in \mathcal{F}_{B_i}\}$. Moreover, let $[k] \setminus \Phi = \{i_1, \ldots, i_n\}$ with $i_1 < \cdots < i_n$. Then $\mathcal{L}(\rho_f) = \{B_{i_1}, \ldots, B_{i_n}\}$. This ends the construction of G', so no other rules are in R'.

First, we prove that for every derivation tree $d \in L(G_{\mathrm{der}}, A) \setminus T_F$ a derivation tree $d' \in L(G'_{\mathrm{der}}, A)$ with $\mathrm{val}(d') = \mathrm{val}(d)$ exists. This shows $L(G) \subseteq L(G')$ because $L(G) = \mathrm{val}(L(G_{\mathrm{der}}, S) \setminus T_F) \cup \mathcal{F}_S$. The proof proceeds by induction on the structure of d. Let $d = \rho(d_1, \ldots, d_k)$ for some $k \in \mathbb{N}_0$, rule $\rho = A \to (u, \mathcal{L}) \in R$ with $\mathcal{L} = \{B_1, \ldots, B_k\}$, and $d_i \in L(G_{\mathrm{der}}, B_i)$ for every $i \in [k]$. Let $\Phi = \{i \in [k] \mid d_i \in T_F\}$, and let f be the substitution function for $\mathcal{L}$ such that $f(B_i) = \mathrm{val}(d_i)$ if $i \in \Phi$ and $f(B_i) = \mathrm{in}(B_i)$ otherwise. Note that $f(B_i) \in \mathcal{F}_{B_i}$ for every $i \in \Phi$ by (2). Since $d \notin T_F$ we have $u[f] \notin \mathcal{F}$. In fact, if $u[f] \in \mathcal{F} \subseteq P_\Sigma(X)^+$, then $f(B_i) \neq \mathrm{in}(B_i)$ for all $i \in [k]$ by Lemma 1(2), which yields that $u[f] = u[B_i \leftarrow \mathrm{val}(d_i) \mid 1 \leq i \leq k] = \mathrm{val}(d)$ is in $\mathcal{F}$ and thus that $d \in T_F$ by (2). Consequently, $\rho_f \in R'$. Now let $[k] \setminus \Phi = \{i_1, \ldots, i_n\}$ with $i_1 < \cdots < i_n$. By the induction hypothesis, there exists a derivation tree $d'_{i_j} \in L(G'_{\mathrm{der}}, B_{i_j})$ with $\mathrm{val}(d'_{i_j}) = \mathrm{val}(d_{i_j})$ for every $j \in [n]$. We now take $d' = \rho_f(d'_{i_1}, \ldots, d'_{i_n}) \in L(G'_{\mathrm{der}}, A)$ and prove that $\mathrm{val}(d') = \mathrm{val}(d)$. Let $[g]$ abbreviate $[B_i \leftarrow \mathrm{val}(d_i) \mid i \in \{i_1, \ldots, i_n\}]$. Then $\mathrm{val}(d') = u[f][g]$. By Lemma 4(4) this implies that $\mathrm{val}(d') = u[B_i \leftarrow f(B_i)[g] \mid 1 \leq i \leq k]$. Clearly, $f(B_i)[g] = \mathrm{val}(d_i)$ for every $i \in [k]$, which shows that $\mathrm{val}(d') = \mathrm{val}(d)$.

It should be clear that the transformation from d to d', as defined above, can be realized by an $\mathrm{LDT}^{\mathrm{R}}$-transducer M from R to R'. It has one state q, and for its look-ahead it uses the regular tree languages $L_{\langle A,t \rangle}$, defined above for $A \in \mathcal{N}$ and $t \in \mathcal{F}_A$ in addition to the regular tree language $L_0 = T_R \setminus T_F$. All subtrees in T_F are deleted by M. The translation of derivation trees $d = \rho(d_1, \ldots, d_k) \in L(G_{\mathrm{der}}, A) \setminus T_F$ (as discussed above) is realized by the rules $\langle q, \rho(y_1 \colon L_{b_1}, \ldots, y_k \colon L_{b_k}) \colon L_0 \rangle \to \rho_f(\langle q, y_{i_1} \rangle, \ldots, \langle q, y_{i_n} \rangle)$ such that $b_i \in \{0\} \cup \{\langle B_i, t_i \rangle \mid t_i \in \mathcal{F}_{B_i}\}$ for all $i \in [k]$, where $f(B_i) = \mathrm{in}(B_i)$ if $b_i = 0$ and $f(B_i) = t_i$ if $b_i = \langle B_i, t_i \rangle$, and $\{i \in [k] \mid b_i = 0\} = \{i_1, \ldots, i_n\}$ with $i_1 < \cdots < i_n$. The translation of derivation trees $d \in L(G_{\mathrm{der}}) \cap T_F$ is realized by the rules $\langle q, \rho(y_1, \ldots, y_k) \colon L_{\langle S,t \rangle} \rangle \to \rho_{S,t}$ with $t \in \mathcal{F}_S$.

Second, we show that $L(G') \subseteq L(G)$. For every $A \in \mathcal{N}$ and $t \in \mathcal{F}_A$, let $d_{A,t}$ be a fixed derivation tree in $L_{\langle A,t \rangle}$, which can be constructed from the regular tree grammar that generates $L_{\langle A,t \rangle}$. Since $\mathcal{F}_S \subseteq L(G)$, it suffices to prove that for every derivation tree $d' \in L(G'_{\mathrm{der}}, A)$ of which the root is labeled with a rule ρ_f, a derivation tree $d \in L(G_{\mathrm{der}}, A) \setminus T_F$ can be constructed such that $\mathrm{val}(d) = \mathrm{val}(d')$. The proof proceeds by induction on the structure of d'. Let $d' = \rho_f(d'_{i_1}, \ldots, d'_{i_n})$ with the same notation as in the construction of G'. By the induction hypotheses, there are derivation trees $d_{i_1}, \ldots, d_{i_n}$ of G such that $d_{i_j} \notin T_F$ and $\mathrm{val}(d_{i_j}) = \mathrm{val}(d'_{i_j})$ for every $j \in [n]$. We now take $d = \rho(d_1, \ldots, d_k)$, where $d_i = d_{B_i, f(B_i)}$ for every $i \in \Phi = [k] \setminus \{i_1, \ldots, i_n\}$. Thus $d_i \in T_F$ and $\mathrm{val}(d_i) = f(B_i)$ for every $i \in \Phi$. Then $d \notin T_F$ because if we suppose $d \in T_F$, then $d_1, \ldots, d_k \in T_F$, which yields $\Phi = [k]$ and the equality

$$u[f] = u[B_i \leftarrow f(B_i) \mid 1 \leq i \leq k] = u[B_i \leftarrow \mathrm{val}(d_i) \mid 1 \leq i \leq k] = \mathrm{val}(d) \ ,$$

which in turn yields the statement $u[f] \in \mathcal{F}$, contradicting the fact that $\rho_f \in R'$. It is easy to check that the $\mathrm{LDT}^{\mathrm{R}}$-transducer M, in the proof of $L(G) \subseteq L(G')$, transforms d into d'. Hence $\mathrm{val}(d) = \mathrm{val}(d')$.

The transformation from d' to d, as defined above, can easily be realized by an LDT-transducer M' with one state q. For every rule ρ' of G', fix either ρ and f with $\rho' = \rho_f$ or S and t with $\rho' = \rho_{S,t}$ (there

may be more than one such choice). In the first case, M' has the rule $\langle q, \rho'(y_1, \ldots, y_n)\rangle \to \rho(t_1, \ldots, t_k)$, where $t_i = d_{B_i, f(B_i)}$ for every $i \in \Phi$ and $t_{i_j} = \langle q, y_j\rangle$ for every $j \in [n]$. In the second case, it has the rule $\langle q, \rho'\rangle \to d_{S,t}$. This ends the proof that G and G' are $\mathrm{LDT}^{\mathrm{R}}$-equivalent. ■

In the next lemma we show how Lemma 28 can be used to remove Δ-free non-initial terminal rules.

Lemma 29 *Let $F \subseteq R$ be the set of Δ-free rules. If $\mathrm{val}(L(G_{\mathrm{der}}, A) \cap T_F)$ is finite for every $A \in \mathcal{N}$, then there is an $\mathrm{LDT}^{\mathrm{R}}$-equivalent MCFTG G' such that all its non-initial terminal rules are Δ-lexicalized. Moreover, if G is almost Σ-growing, then so is G'.*

Proof For the purpose of effectiveness, we assume that the elements of $\mathrm{val}(L(G_{\mathrm{der}}, A) \cap T_F)$ are effectively given for every $A \in \mathcal{N}$. Let $\mathcal{F}$ be the set of Δ-free forests in $P_\Sigma(X)^+$. As observed before, for every derivation tree d, the value $\mathrm{val}(d)$ is Δ-free if and only if all rules that occur in d are Δ-free. Thus, $\mathcal{F}$ and F satisfy requirement (2) of Lemma 28. Hence, for every $A \in \mathcal{N}$ the set $\mathcal{F}_A$, given by $\mathcal{F}_A = L(G, A) \cap \mathcal{F} = \mathrm{val}(L(G_{\mathrm{der}}, A) \cap T_F)$, is finite and its elements are effectively given. Thus, $\mathcal{F}$ also satisfies requirement (1) of Lemma 28.

Let G' be the $\mathrm{LDT}^{\mathrm{R}}$-equivalent MCFTG as constructed in the proof of Lemma 28. Then all non-initial terminal rules of G' are Δ-lexicalized. Assume now that G is almost Σ-growing. Since all non-initial terminal rules of G are Σ-lexicalized, the elements of $L(G, A)$, and hence of $\mathcal{F}_A$, are Σ-lexicalized (by Theorem 9 and Lemma 10(1)). Now consider a rule ρ of G and a substitution function f for $\mathcal{L}(\rho)$ such that $f(B) \in \mathcal{F}_B \cup \{\mathrm{in}(B)\}$ for every $B \in \mathcal{L}(\rho)$. If there is at least one $B \in \mathcal{L}$ such that $f(B) \in \mathcal{F}_B$, then the rule ρ_f of G' is Σ-lexicalized by Lemma 1(2). Otherwise, we obviously have $\rho_f = \rho$ and ρ satisfies the requirements by assumption. Hence G' is almost Σ-growing. ■

We now remove the Σ-free terminal rules from G.

Lemma 30 *For every MCFTG G there is an $\mathrm{LDT}^{\mathrm{R}}$-equivalent MCFTG G' of which all terminal rules are Σ-lexicalized.*

Proof As in the previous lemma, let F be the set of Σ-free rules in R, and let $\mathcal{F}$ be the set of Σ-free forests in $P_\Sigma(X)^+$. Then $\mathrm{val}(L(G_{\mathrm{der}}, A) \cap T_F) = L(G, A) \cap \mathcal{F}$ as demonstrated in the proof of Lemma 29. Clearly, a forest $t \in P_\Sigma(X)^+$ is Σ-free if and only if $t \in x_1^+$; i.e., t is of the form $(x_1, \ldots, x_1)$. Such a forest t can only be generated by a big nonterminal of rank $(1, \ldots, 1)$. Hence, $L(G, A) \cap \mathcal{F}$ is either empty or equal to $\{x_1^k\}$ with $k = |A|$. Moreover, $\mathrm{val}(L(G_{\mathrm{der}}, A) \cap T_F)$ can be computed because it is empty if and only if the regular tree language $L(G_{\mathrm{der}}, A) \cap T_F$ is empty. By Lemma 29 there is an $\mathrm{LDT}^{\mathrm{R}}$-equivalent MCFTG G', of which all non-initial terminal rules are Σ-lexicalized. Obviously, the initial terminal rules of an MCFTG are also Σ-lexicalized. ■

Example 31 In the MCFTG G of Example 7, the rules $\rho_4 = B(x_1) \to x_1$ and $\rho_4' = B'(x_1) \to x_1$ are the only Σ-free rules. The construction in the proof of Lemma 28 asks us to apply these rules in all possible ways to the right-hand sides of the other rules. Thus, we change the set R of rules by removing rules ρ_4 and ρ_4' and adding the following rules:

$$\begin{array}{lll}
A \to T_1(\sigma(T_2, T_3)) & & \\
B(x_1) \to \sigma(x_1, B'(A)) & B(x_1) \to \sigma(B(x_1), A) & B(x_1) \to \sigma(x_1, A) \\
B'(x_1) \to \sigma(x_1, B'(A)) & B'(x_1) \to \sigma(B(x_1), A) & B'(x_1) \to \sigma(x_1, A) \ .
\end{array}$$

In the resulting MCFTG G', which we will call G again, all terminal rules are Σ-lexicalized. In fact, G is now both Σ-lexicalized and Σ-growing, and all its terminal rules are Δ-lexicalized for $\Delta = \{\alpha, \beta, \tau, \nu\}$ as in Example 24. □

Our second goal is to make sure that all monic rules (i.e., rules whose right-hand side contains exactly one big nonterminal) are Δ-lexicalized. In the next construction we remove Δ-free monic rules thereby generalizing the removal of chain rules $A \to B$ from a context-free grammar [48].

Lemma 32 *Suppose that all non-initial terminal rules of G are Δ-lexicalized. Let $F \subseteq R$ be the set of Δ-free monic rules. If $\mathrm{val}(DL(G_{\mathrm{der}}, A) \cap T_{\mathcal{N} \cup F})$ is finite for every $A \in \mathcal{N}$, then there is an $\mathrm{LDT}^{\mathrm{R}}$-equivalent almost Δ-growing MCFTG G'.*

Proof Let $\mathcal{F}_A = \mathrm{val}(DL(G_{\mathrm{der}}, A) \cap T_{\mathcal{N} \cup F})$ for every $A \in \mathcal{N}$. Again, for the purpose of effectiveness, we assume that the elements of $\mathcal{F}_A$ are effectively given. Note that $\mathrm{in}(A) \in \mathcal{F}_A$. Every forest $t \in \mathcal{F}_A$ is of the form $\mathrm{val}(d)$ with $d \in DL(G_{\mathrm{der}}, A) \cap T_{\mathcal{N} \cup F}$, and every such derivation tree d is of the form $d = wB$ with $w \in F^*$ and $B \in \mathcal{N}$. Hence t is Δ-free because all rules that occur in d are Δ-free. Moreover, by Lemma 10(2), t is uniquely N-labeled and $\mathrm{occ}_N(t) = \mathrm{occ}(B)$. In other words, the big nonterminal B occurs exactly once in t, and no other nonterminals occur in t. We will denote B by B_t. Note that, since G is start-separated, if $B_t = S$ then $A = S$ because $w = \varepsilon$. For every $t \in \mathcal{F}_A$, let $d_{A,t} \in T_{\mathcal{N} \cup F}$ be a particular derivation tree of G of type A such that $\mathrm{val}(d_{A,t}) = t$. Such a derivation tree can be computed by Lemma 12 applied with $\mathcal{N}' = \mathcal{N}$.

We construct the MCFTG $G' = (N, \mathcal{N}, \Sigma, S, R')$ such that for every big nonterminal $A \in \mathcal{N}$, tree $t \in \mathcal{F}_A$, and rule $\rho = B_t \to (u, \mathcal{L}) \in R \setminus F$, the rule $\rho_{A,t} = A \to (t[B_t \leftarrow u], \mathcal{L})$ is in R', where the links in $\mathcal{L}$ have the same order as in the rule ρ. Since $\rho \notin F$, it is straightforward to check that $\rho_{A,t}$ satisfies the requirements for G' to be almost Δ-growing: (i) If ρ is Δ-lexicalized, then so is $\rho_{A,t}$ because u is substituted for B_t. (ii) If $\rho_{A,t}$ is monic, then ρ is monic and hence Δ-lexicalized because $\rho \notin F$. (iii) If ρ is initial (i.e., $B_t = S$), then $\rho_{A,t}$ is initial (because $A = S$); thus, if $\rho_{A,t}$ is non-initial terminal, then ρ is non-initial terminal and hence Δ-lexicalized by assumption on G.

To show the correctness of G', we first prove that for every derivation tree $d \in L(G_{\mathrm{der}}, A)$ there is a derivation tree $d' \in L(G'_{\mathrm{der}}, A)$ with $\mathrm{val}(d') = \mathrm{val}(d)$. Clearly, d has the unique form $d = w\rho(d_1, \ldots, d_k)$ such that $w \in F^*$, $\rho \notin F$, and $d_1, \ldots, d_k \in T_R$. Let $\rho = B \to (u, \mathcal{L})$ with $\mathcal{L} = \{B_1, \ldots, B_k\}$, and let $t = \mathrm{val}(wB) \in \mathcal{F}_A$. By the induction hypothesis there is a derivation tree $d'_i \in L(G'_{\mathrm{der}}, B_i)$ with $\mathrm{val}(d'_i) = \mathrm{val}(d_i)$ for every $i \in [k]$. We take $d' = \rho_{A,t}(d'_1, \ldots, d'_k)$. Then

$$\begin{aligned}\mathrm{val}(d') &= t[B \leftarrow u][B_i \leftarrow \mathrm{val}(d_i) \mid 1 \le i \le k] = t[B \leftarrow u[B_i \leftarrow \mathrm{val}(d_i) \mid 1 \le i \le k]\,] \\ &= t[B \leftarrow \mathrm{val}(\rho(d_1, \ldots, d_k))] = \mathrm{val}(w\rho(d_1, \ldots, d_k)) = \mathrm{val}(d) \ ,\end{aligned}$$

where the second equality holds by Lemma 4(4) and the penultimate equality holds by Lemma 11. This shows that $L(G) \subseteq L(G')$.

The $\mathrm{LDT}^{\mathrm{R}}$-transducer M that transforms d into d', as above, uses the tree languages

$$L_{A,t} = \{wd \in L(G_{\mathrm{der}}, A) \mid w \in F^*,\ d \in L(G_{\mathrm{der}}, B_t),\ d(\varepsilon) \notin F,\ \mathrm{val}(wB_t) = t\}$$

as look-ahead, where $A \in \mathcal{N}$ and $t \in \mathcal{F}_A$. It is easy to see that $L_{A,t}$ is regular. An RTG that generates $L_{A,t}$ can be obtained from the grammar for the regular tree language $L_{\langle A,t \rangle}$ in the proof of Lemma 12 as follows. First, add the nonterminals and rules of G_{der}. Second, replace every rule $\langle B, \mathrm{in}(B) \rangle \to B$ by all rules $\langle B, \mathrm{in}(B) \rangle \to \rho(B_1, \ldots, B_k)$, where $B \to \rho(B_1, \ldots, B_k)$ is a rule of G_{der} and $\rho \notin F$. The transducer M has initial state q_0 and the states $q_{A,t}$ for every $A \in \mathcal{N}$ and $t \in \mathcal{F}_A$. For every rule $\rho \in R \setminus F$, the transducer M has the rule $\langle q_0, \rho(y_1, \ldots, y_k) \rangle \to \rho(\langle q_0, y_1 \rangle, \ldots, \langle q_0, y_k \rangle)$ and all the rules $\langle q_{A,t}, \rho(y_1, \ldots, y_k) \rangle \to \rho_{A,t}(\langle q_0, y_1 \rangle, \ldots, \langle q_0, y_k \rangle)$. Moreover, for every rule $\rho \in F$, the transducer M has all rules $\langle q_0, \rho(y_1) \colon L_{A,t} \rangle \to \langle q_{A,t}, y_1 \rangle$ and $\langle q_{A,t}, \rho(y_1) \rangle \to \langle q_{A,t}, y_1 \rangle$. It should be clear that M indeed transforms d into d'.

Next, we prove that for every derivation tree $d' \in L(G'_{\mathrm{der}}, A)$ there is a derivation tree $d \in L(G_{\mathrm{der}}, A)$ with $\mathrm{val}(d) = \mathrm{val}(d')$. The proof is by induction on d', so let $d' = \rho_{A,t}(d'_1, \ldots, d'_k)$ with ρ, A, and t as in the construction of G'. By the induction hypothesis, there is a derivation tree d_i of G such that $\mathrm{val}(d_i) = \mathrm{val}(d'_i)$ for every $i \in [k]$. We now take $d = d_{A,t}[B_t \leftarrow \rho(d_1, \ldots, d_k)]$, where the derivation tree $d_{A,t}$ was defined at the end of the first paragraph of this proof. Since $d_{A,t}$ is of the form wB_t with $w \in F^*$, and hence $d = w\rho(d_1, \ldots, d_k)$, it should be clear that the construction in the proof of $L(G) \subseteq L(G')$ (i.e., the $\mathrm{LDT}^{\mathrm{R}}$-transducer M) transforms d into d', which implies that $\mathrm{val}(d) = \mathrm{val}(d')$.

The transformation from d' to d, as defined above, can easily be realized by an LDT-transducer M' with one state q. For every rule ρ' of G', fix ρ, A, and t such that $\rho' = \rho_{A,t}$. Then M' has the rule $\langle q, \rho'(y_1, \ldots, y_k) \rangle \to d_{A,t}[B_t \leftarrow \rho(\langle q, y_1 \rangle, \ldots, \langle q, y_k \rangle)]$. We finally observe that the transformation from d to d' can also be realized by an LDT-transducer (without look-ahead), but the above transducer M is easier to understand. ∎

Lemma 33 *For every MCFTG G there is an* $\mathrm{LDT}^{\mathrm{R}}$*-equivalent almost Σ-growing MCFTG G'.*

Proof By Lemma 30 and Proposition 13, we may assume that all terminal rules of G are Σ-lexicalized. Let $F \subseteq R$ be the set of Σ-free monic rules. The statement holds using Lemma 32 if we prove that $\mathcal{F}_A = \mathrm{val}(DL(G_{\mathrm{der}}, A) \cap T_{\mathcal{N}\cup F})$ is finite and that its elements can be computed for every $A \in \mathcal{N}$. For every big nonterminal A, let $\mathcal{M}_A$ be the set of all Σ-free forests t in $P_{N\cup\Sigma}(X)^+$ such that $\mathrm{rk}(t) = \mathrm{rk}(A)$, t is uniquely N-labeled, and $\mathrm{occ}_N(t) = \mathrm{occ}(B)$ for some $B \in \mathcal{N}$. Clearly $\mathcal{M}_A$ is finite because $|\mathrm{pos}_{N\cup\Sigma}(t)| = |\mathrm{pos}_N(t)| \leq \mu(G)$ and $|\mathrm{pos}_X(t)| \leq \mu(G) \cdot \theta(G)$. As argued in the beginning of the proof of Lemma 32, $\mathcal{F}_A \subseteq \mathcal{M}_A$. Consequently, $\mathcal{F}_A$ is finite and its elements can be computed by a standard iteration because the sets $\mathcal{F}_A$ with $A \in \mathcal{N}$ are the smallest sets of forests such that (i) $\mathrm{in}(A) \in \mathcal{F}_A$ and (ii) if $A \to (u, \{B\}) \in F$ and $t \in \mathcal{F}_B$, then $u[B \leftarrow t] \in \mathcal{F}_A$. ∎

Let G be an almost Σ-growing MCFTG. Then, for every forest t, there are only finitely many derivation trees d such that $\mathrm{val}(d) = t$ by inequality (†) of Lemma 27. This implies that the finiteness problem is decidable for $L(G)$ and $L(G, A)$. In fact, $L(G)$ is finite if and only if $L(G_{\mathrm{der}})$ is finite, which is decidable because G_{der} is an RTG. Moreover, if $L(G)$ is finite, then the elements of $L(G)$ can be computed because the elements of $L(G_{\mathrm{der}})$ can be computed and $L(G) = \mathrm{val}(L(G_{\mathrm{der}}))$. Similar statements hold for $L(G, A)$. Thus, by Lemma 33, the finiteness problem is decidable for MCFTGs.

We now show that if G is almost Σ-growing, then the requirements of Lemmas 28 and 32 are fulfilled.

Lemma 34 *Let G be almost Σ-growing. Moreover, let F be the set of all Δ-free rules and $F' \subseteq F$ be the set of all Δ-free monic rules. Finally, let $\mathcal{F}_A = \mathrm{val}(L(G_{\mathrm{der}}, A) \cap T_F)$ and $\mathcal{F}'_A = \mathrm{val}(DL(G_{\mathrm{der}}, A) \cap T_{\mathcal{N}\cup F'})$ for every $A \in \mathcal{N}$.*

(1) *It is decidable for $A \in \mathcal{N}$ whether or not $\mathcal{F}_A$ (respectively, $\mathcal{F}'_A$) is finite, and if so, its elements can be computed.*
(2) *If G has finite Δ-ambiguity, then $\mathcal{F}_A$ and $\mathcal{F}'_A$ are finite for every $A \in \mathcal{N}$.*

Proof For (1) we observe that since G is almost Σ-growing, inequality (†) of Lemma 27 implies that $\mathcal{F}_A$ is finite if and only if $L(G_{\mathrm{der}}, A) \cap T_F$ is finite. The latter is a regular tree language, and it is decidable whether or not it is finite. Moreover, if so, its elements, and thus also the elements of $\mathcal{F}_A$, can be computed. The same argument holds for $\mathcal{F}'_A$.

For (2) we assume that G has finite Δ-ambiguity and that $\mathcal{F}_A$ is infinite. Since we may assume that G and G_{der} are reduced, there is a derivation tree $d_0 \in DL(G_{\mathrm{der}}, S)$ with $|\mathrm{pos}_{\mathcal{N}}(d_0)| = |\mathrm{pos}_A(d_0)| = 1$. Let $D_0 = \{d_0[A \leftarrow d] \mid d \in L(G_{\mathrm{der}}, A) \cap T_F\} \subseteq L(G_{\mathrm{der}})$. Since $\mathcal{F}_A$ is infinite, also $L(G_{\mathrm{der}}, A) \cap T_F$ is infinite, and thus D_0 is infinite by Lemma 1. Since G is almost Σ-growing, the set $L_0 = \mathrm{val}(D_0)$ is an infinite subset of $L(G)$. Now, for every derivation tree $d' \in T_{\mathcal{N}\cup R}$, let $\mathrm{pr}_\Delta(d') = \sum_{p\in\mathrm{pos}_R(d')} |\mathrm{pos}_\Delta(\mathrm{rhs}(d'(p)))|$. Lemma 10(1) and Lemma 1 yield $|\mathrm{pos}_\Delta(\mathrm{val}(d_0[A \leftarrow d]))| = \mathrm{pr}_\Delta(d_0[A \leftarrow d]) = \mathrm{pr}_\Delta(d_0) + \mathrm{pr}_\Delta(d) = \mathrm{pr}_\Delta(d_0)$ for every $d \in L(G_{\mathrm{der}}, A) \cap T_F$, where the last equality uses $d \in T_F$. Consequently, $|\mathrm{pos}_\Delta(t)| \leq \mathrm{pr}_\Delta(d_0)$ for every tree t in the infinite set L_0, which contradicts the finite Δ-ambiguity of $L(G)$.

A similar proof works for $\mathcal{F}'_A$. Since $DL(G_{\mathrm{der}}, A) \cap T_{\mathcal{N}\cup F'}$ is infinite, there exists $B \in \mathcal{N}$ such that $DL(G_{\mathrm{der}}, A) \cap T_{\{B\}\cup F'}$ is infinite. Since G_{der} is reduced, there exists a derivation tree $d_1 \in L(G_{\mathrm{der}}, B)$. Now let $D'_0 = \{d_0[A \leftarrow d[B \leftarrow d_1]] \mid d \in DL(G_{\mathrm{der}}, A) \cap T_{\{B\}\cup F'}\} \subseteq L(G_{\mathrm{der}})$. By similar arguments as above, we then obtain that $|\mathrm{pos}_\Delta(t)| \leq \mathrm{pr}_\Delta(d_0) + \mathrm{pr}_\Delta(d_1)$ for every tree t in the infinite set $L'_0 = \mathrm{val}(D'_0) \subseteq L(G)$, which again contradicts the finite Δ-ambiguity of $L(G)$. ∎

Now we are able to turn G into an equivalent almost Δ-growing MCFTG, provided that it has finite Δ-ambiguity.

Lemma 35 *It is decidable whether or not the MCFTG G has finite Δ-ambiguity, and if so, there is an* $\mathrm{LDT}^{\mathrm{R}}$*-equivalent almost Δ-growing MCFTG G'.*

Proof By Lemma 33 we may assume that G is almost Σ-growing. By Lemma 34 it is decidable whether $\mathcal{F}_A$ is finite for every $A \in \mathcal{N}$, and if not, then G does not have finite Δ-ambiguity. If they are, then we may assume by Lemma 29 that all non-initial terminal rules of G are Δ-lexicalized. Again by Lemma 34, it is decidable whether $\mathcal{F}'_A$ is finite for every $A \in \mathcal{N}$, and if not, then G does not have finite Δ-ambiguity. If they are, then we may assume by Lemma 32 that G is almost Δ-growing. Finally, in this case G has finite Δ-ambiguity by Lemma 27. ∎

Example 36 The MCFTG G of Example 31 is already Σ-growing. Moreover, all its terminal rules are Δ-lexicalized for $\Delta = \{\alpha, \beta, \tau, \nu\}$. Let us turn G into an almost Δ-growing grammar by Lemma 32. We omit parentheses around the arguments of unary terminals. The set F of Δ-free monic rules of G consists of the rules $A \to T_1(\sigma(T_2, T_3))$, $B(x_1) \to \sigma(x_1, A)$, and $B'(x_1) \to \sigma(x_1, A)$. Next, for each big nonterminal $A' \in \mathcal{N}$ we compute the sets $\mathcal{F}_{A'} = \mathrm{val}(DL(G_{\mathrm{der}}, A') \cap T_{\mathcal{N} \cup F})$ and obtain

$$\begin{array}{lll} \mathcal{F}_T = \{\mathrm{in}(T)\} & \mathcal{F}_A = \{\mathrm{in}(A),\, T_1(\sigma(T_2, T_3))\} & \mathcal{F}_B = \{\mathrm{in}(B),\, \sigma(x_1, A),\, \sigma(x_1, T_1(\sigma(T_2, T_3)))\} \\ & \mathcal{F}_S = \{\mathrm{in}(S)\} & \mathcal{F}_{B'} = \{\mathrm{in}(B'),\, \sigma(x_1, A),\, \sigma(x_1, T_1(\sigma(T_2, T_3)))\} \ , \end{array}$$

where $T = (T_1, T_2, T_3)$, which are all finite. The construction in the proof of Lemma 32 asks us to apply

- the rules $\rho_5 = (T_1(x_1), T_2, T_3) \to (\alpha T_1(\beta x_1), \alpha T_2, \gamma T_3)$ and $\rho_6 = (T_1(x_1), T_2, T_3) \to (x_1, \tau, \nu)$ for T to $T_1(\sigma(T_2, T_3)) \in \mathcal{F}_A$ and $\sigma(x_1, T_1(\sigma(T_2, T_3))) \in \mathcal{F}_B \cap \mathcal{F}_{B'}$, and
- the rule $\rho_2 = A \to T_1(\sigma(B(T_2), T_3))$ for A to $\sigma(x_1, A) \in \mathcal{F}_B \cap \mathcal{F}_{B'}$.

Consequently, we change the set of rules of G by removing the above three Δ-free monic rules and adding the following 5 rules, and the 3 additional rules that make B' an alias of B:

$$\begin{array}{lll} A \to \alpha T_1(\beta\sigma(\alpha T_2, \gamma T_3)) & A \to \sigma(\tau, \nu) & \\ B(x_1) \to \sigma(x_1, \alpha T_1(\beta\sigma(\alpha T_2, \gamma T_3))) & B(x_1) \to \sigma(x_1, \sigma(\tau, \nu)) & B(x_1) \to \sigma(x_1, T_1(\sigma(B(T_2), T_3))) \ . \end{array}$$

The resulting grammar G', which we will again call G, now has the following rules (and the rules required to make B' an alias of B):

$$\begin{array}{lll} A \to \alpha T_1(\beta\sigma(\alpha T_2, \gamma T_3)) & A \to \sigma(\tau, \nu) & A \to T_1(\sigma(B(T_2), T_3)) \\ B(x_1) \to \sigma(x_1, \alpha T_1(\beta\sigma(\alpha T_2, \gamma T_3))) & B(x_1) \to \sigma(x_1, \sigma(\tau, \nu)) & B(x_1) \to \sigma(x_1, T_1(\sigma(B(T_2), T_3))) \\ B(x_1) \to \sigma(B(x_1), B'(A)) & B(x_1) \to \sigma(x_1, B'(A)) & B(x_1) \to \sigma(B(x_1), A) \\ T \to (\alpha T_1(\beta x_1),\, \alpha T_2,\, \gamma T_3) & S \to \alpha A & T \to (x_1, \tau, \nu) \end{array}$$

with $T = (T(x_1), T_2, T_3)$. This MCFTG G is not only almost Δ-growing, but even Δ-growing. It is also almost $\{\alpha, \tau\}$-growing, which proves that $L(G)$ has finite $\{\alpha, \tau\}$-ambiguity by Lemma 27 (as observed in Example 24). The only rules of G (without rules with left-hand side B') that are not Δ-lexicalized are

$$\begin{array}{lll} A \to T_1(\sigma(B(T_2), T_3)) & B(x_1) \to \sigma(B(x_1), A) & \\ B(x_1) \to \sigma(x_1, T_1(\sigma(B(T_2), T_3))) & B(x_1) \to \sigma(B(x_1), B'(A)) & B(x_1) \to \sigma(x_1, B'(A)) \ . \end{array}$$

It is easy to lexicalize this grammar. The first non-lexicalized rule $\rho_2 = A \to T_1(\sigma(B(T_2), T_3))$ can be replaced by the two lexicalized rules $A \to \alpha T_1(\beta(\sigma(B(\alpha T_2), \gamma T_3)))$ and $A \to \sigma(B(\tau), \nu)$ that are obtained from ρ_2 by applying the two rules for T to its right-hand side. By Lemma 4(4) this process preserves $L(G)$, and it should be clear that the resulting grammar is $\mathrm{LDT}^{\mathrm{R}}$-equivalent to G. Now all four rules for A are lexicalized. The remaining non-lexicalized rule in the first column can be replaced by two lexicalized rules in the same way. Finally, the same process can be used for all the remaining non-lexicalized rules by applying the four lexicalized rules for A to their right-hand sides; this does, however, not preserve $\mathrm{LDT}^{\mathrm{R}}$-equivalence.[20] □

It remains to construct an equivalent Δ-growing MCFTG, which is the main result of this section.

Theorem 37 *It is decidable whether or not the MCFTG G has finite Δ-ambiguity, and if so, there is an $\mathrm{LDT}^{\mathrm{R}}$-equivalent Δ-growing MCFTG G'. Moreover, $\theta(G') = \theta(G)$ and $\mu(G') = \mu(G)$.*

Proof By Lemma 35 it suffices to show that if G is almost Δ-growing, then there is an $\mathrm{LDT}^{\mathrm{R}}$-equivalent Δ-growing MCFTG G'. Consequently, it remains to remove all non-initial terminal rules that are singly Δ-lexicalized, using the construction in the proof of Lemma 28. Let $\mathcal{F} = \{t \in P_\Sigma(X)^+ \mid |\mathrm{pos}_\Delta(t)| = 1\}$, and let F be the set of all (terminal) rules $A \to u \in R$ such that $u \in \mathcal{F}$. Note that $T_F = F$. Since G is almost Δ-growing, $\mathrm{val}(d) \in \mathcal{F}$ if and only if $d \in T_F$, for every $d \in L(G_{\mathrm{der}}, A)$. In fact, since all non-initial terminal rules are Δ-lexicalized, $\mathrm{pos}_\Delta(\mathrm{val}(d')) \neq \emptyset$ for every $d' \in L(G_{\mathrm{der}}, B)$ with $B \in \mathcal{N} \setminus \{S\}$. Hence, if $d = \rho(d_1, \ldots, d_k)$ with $k \geq 1$, then either $k \geq 2$ and both $\mathrm{val}(d_1)$ and $\mathrm{val}(d_2)$ contribute a lexical position

[20] The resulting MCFTG is $\mathcal{X}$-equivalent to G for the class $\mathcal{X}$ of tree transductions realized by *finite-copying* deterministic top-down tree transducers with regular look-ahead.

to $\mathrm{val}(d)$, or $k = 1$ and both $\mathrm{val}(d_1)$ and the right-hand side of ρ contribute a lexical position to $\mathrm{val}(d)$ because the monic rule ρ is Δ-lexicalized. Thus, $\mathcal{F}$ satisfies requirement (2) of Lemma 28. Additionally, $\mathcal{F}$ satisfies requirement (1) of Lemma 28 because $L(G, A) \cap \mathcal{F} = \mathrm{val}(L(G_{\mathrm{der}}, A) \cap T_F) = \{u \mid A \to u \in F\}$. Let $\mathcal{F}_A = \{u \mid A \to u \in F\}$ for every $A \in \mathcal{N}$, and let G' be the $\mathrm{LDT}^{\mathrm{R}}$-equivalent MCFTG as constructed in the proof of Lemma 28. If $\rho = A \to (u, \mathcal{L})$ is a rule of G, and f is a substitution function for $\mathcal{L}$ such that $f(B) \in \mathcal{F}_B \cup \{\mathrm{in}(B)\}$ for every $B \in \mathcal{L}$, then the new rule $\rho_f = A \to (u[f], \{B \in \mathcal{L} \mid f(B) = \mathrm{in}(B)\})$ is either equal to the old rule ρ (because $f(B) = \mathrm{in}(B)$ for all $B \in \mathcal{L}$) or is Δ-lexicalized (because $f(B) \in \mathcal{F}$ for some $B \in \mathcal{L}$). This implies that G' is almost Δ-growing. Moreover, ρ_f is a rule of G' only if $u[f] \notin \mathcal{F}$, so G' does not have non-initial terminal rules that are singly Δ-lexicalized, and hence is Δ-growing.

We finally observe that G' has the same ranked alphabet N of nonterminals and the same set $\mathcal{N}$ of big nonterminals as G, as one can easily check from the constructions in Lemmas 28 and 32. That implies that $\theta(G') = \theta(G)$ and $\mu(G') = \mu(G)$. ■

Example 38 We have seen that the new grammar G in Example 36 is almost $\{\alpha, \tau\}$-growing. However, it is not $\{\alpha, \tau\}$-growing because the right-hand side of each terminal rule has exactly one lexical position (always labeled τ). Let F be the set of all terminal rules of G; i.e.,

$$F = \big\{A \to \sigma(\tau, \nu),\ B(x_1) \to \sigma(x_1, \sigma(\tau, \nu)),\ B'(x_1) \to \sigma(x_1, \sigma(\tau, \nu)),\ (T_1(x_1), T_2, T_3) \to (x_1, \tau, \nu)\big\}\ .$$

In the construction in the proof of Theorem 37 we apply the rules of F in all possible ways to the right-hand sides of the other rules of G (and then remove the rules F). As an example, the rule $B(x_1) \to \sigma(x_1, B'(A))$ is replaced by itself and the following three additional $\{\alpha, \tau\}$-growing rules

$$B(x_1) \to \sigma\big(x_1, \underbrace{\sigma(A, \sigma(\tau, \nu))}_{B'(A)}\big) \quad B(x_1) \to \sigma\big(x_1, B'(\underbrace{\sigma(\tau, \nu)}_{A})\big) \quad \text{and} \quad B(x_1) \to \sigma\big(x_1, \underbrace{\sigma(\underbrace{\sigma(\tau, \nu)}_{A}, \sigma(\tau, \nu))}_{B'(\sigma(\tau,\nu))}\big)\ ,$$

in which we marked the substitutions. □

Since every MCFTG has finite Σ-ambiguity, we obtain the following result from Theorem 37. It generalizes the corresponding result of [89, 90] for spCFTGs, which is the special case $\mu(G) = 1$.

Corollary 39 *For every MCFTG G there is an $\mathrm{LDT}^{\mathrm{R}}$-equivalent Σ-growing MCFTG G'. Moreover, $\theta(G') = \theta(G)$ and $\mu(G') = \mu(G)$.*

At the end of this section we consider an additional basic normal form for MCFTGs that generalizes one that is familiar from multiple context-free grammars (viz. condition (N3) of [87, Lemma 2.2]), and will be needed in Section 6.1. We say that the MCFTG G is *nonerasing* if $u_i \neq x_1$ for every rule $(A_1, \ldots, A_n) \to ((u_1, \ldots, u_n), \mathcal{L})$ and every $i \in [n]$. Note that in a grammar G, the tree u_i can only be equal to x_1 if $\mathrm{rk}(A_i) = 1$.

Lemma 40 *For every MCFTG G there is an $\mathrm{LDT}^{\mathrm{R}}$-equivalent nonerasing MCFTG G'. If the grammar G is Δ-lexicalized, then so is G'. Moreover, $\theta(G') = \theta(G)$ and $\mu(G') = \mu(G)$.*

Proof For a sequence $w = (a_1, \ldots, a_n)$ we denote, in this proof only, $[n]$ by $\mathrm{num}(w)$, and a_j by $w|_j$ for every $j \in \mathrm{num}(w)$. For every $\Psi \subseteq \mathrm{num}(w)$, we denote by $w|_\Psi$ the "scattered subsequence" $(a_{j_1}, \ldots, a_{j_m})$ of w, in which $\Psi = \{j_1, \ldots, j_m\}$ and $1 \leq j_1 < \cdots < j_m \leq n$. Intuitively, $w|_\Psi$ is obtained from w by selecting the j-th element of w for every $j \in \Psi$.

By Lemma 30 we may assume that all terminal rules of $G = (N, \mathcal{N}, \Sigma, S, R)$ are Σ-lexicalized. Moreover, we can assume that G has disjoint big nonterminals, as observed after Lemma 20. The set $\mathcal{N}'$ of big nonterminals of the new grammar $G' = (N, \mathcal{N}', \Sigma, S, R')$ consists of all $A|_\Psi$ such that $A \in \mathcal{N}$, $\Psi \subseteq \mathrm{num}(A)$, $\Psi \neq \emptyset$, and $\mathrm{rk}(A|_j) = 1$ for every $j \in \mathrm{num}(A) \setminus \Psi$. Intuitively, Ψ selects those nonterminals of A that do not generate x_1. Since all terminal rules of G are Σ-lexicalized, it is not possible that all nonterminals of A generate x_1. Note that $S = S|_{\{1\}}$ and that for every $A' \in \mathcal{N}'$ there are a unique $A \in \mathcal{N}$ and a unique $\Psi \subseteq \mathrm{num}(A)$ such that $A' = A|_\Psi$ because G has disjoint big nonterminals. Note also that $\mathrm{num}(A) = \mathrm{num}(u)$ for every rule $A \to (u, \mathcal{L})$ of G.

Let $\rho = A \to (u, \mathcal{L})$ be a rule of G with $\mathcal{L} = \{B_1, \ldots, B_k\} \subseteq \mathcal{N}$, and let $\Psi_1, \ldots, \Psi_k \subseteq \mathbb{N}$ such that $B_i|_{\Psi_i} \in \mathcal{N}'$ for every $i \in [k]$. Finally, let $u' = u[B_i|_j \leftarrow x_1 \mid i \in [k], j \notin \Psi_i]$, and

let $\Psi = \{j \in \mathrm{num}(A) \mid u'|_j \neq x_1\}$. Then R' contains the rule $\rho_{\Psi_1,\ldots,\Psi_k} = A|_\Psi \to (u'|_\Psi, \mathcal{L}')$ with $\mathcal{L}' = \{B_1|_{\Psi_1}, \ldots, B_k|_{\Psi_k}\}$ provided that $\Psi \neq \emptyset$. This concludes the definition of G'.

For every derivation tree $d \in L(G_{\mathrm{der}}, A)$ we define $\Psi(d) = \{j \in \mathrm{num}(A) \mid \mathrm{val}(d)|_j \neq x_1\}$. Then, as already observed before, we have $A|_{\Psi(d)} \in \mathcal{N}'$. It is straightforward to verify that if $d = \rho(d_1, \ldots, d_k)$, where ρ is the rule of the previous paragraph, then the left-hand side of the rule $\rho_{\Psi(d_1),\ldots,\Psi(d_k)}$ is $A|_{\Psi(d)}$ because $\mathrm{val}(d)|_j = x_1$ if and only if $u|_j = wx_1$ with $w \in \{B_i|_\ell \mid i \in [k],\, \ell \in \mathrm{num}(B_i),\, \mathrm{val}(d_i)|_\ell = x_1\}^*$.

For every derivation tree $d \in L(G_{\mathrm{der}}, A)$ there exists a derivation tree $d' \in L(G'_{\mathrm{der}}, A|_{\Psi(d)})$ such that $\mathrm{val}(d') = \mathrm{val}(d)|_{\Psi(d)}$. In fact, let $d = \rho(d_1, \ldots, d_k)$, and let $d'_i \in L(G'_{\mathrm{der}}, B_i|_{\Psi(d_i)})$ be a derivation tree such that $\mathrm{val}(d'_i) = \mathrm{val}(d_i)|_{\Psi(d_i)}$ for every $i \in [k]$, which exist by the induction hypotheses. By Lemma 4(2) we have $\mathrm{val}(d') = \mathrm{val}(d)|_{\Psi(d)}$ for $d' = \rho_{\Psi(d_1),\ldots,\Psi(d_k)}(d'_1, \ldots, d'_k)$. This shows that $L(G) \subseteq L(G')$. Clearly, $L_\Psi = \{d \in L(G_{\mathrm{der}}, A) \mid \Psi(d) = \Psi\}$ is a regular tree language for every Ψ. Thus, d' can be computed from d by the one-state LDT^{R}-transducer M with the rules

$$\langle q, \rho(y_1 \colon L_{\Psi_1}, \ldots, y_k \colon L_{\Psi_k})\rangle \to \rho_{\Psi_1,\ldots,\Psi_k}(\langle q, y_1\rangle, \ldots, \langle q, y_k\rangle) \ .$$

Vice versa, for every derivation tree $d' \in L(G'_{\mathrm{der}}, A|_\Psi)$ there exists a derivation tree $d \in L(G_{\mathrm{der}}, A)$ such that $M(d) = d'$ and $\Psi = \Psi(d)$, where A is uniquely determined by $A|_\Psi$ because G has disjoint big nonterminals. In fact, let $d' = \rho'(d'_1, \ldots, d'_k)$ with $d'_i \in L(G'_{\mathrm{der}}, B_i|_{\Psi_i})$. Then there exists a rule ρ as above such that $\rho' = \rho_{\Psi_1,\ldots,\Psi_k}$. Clearly, if $d_i \in L(G_{\mathrm{der}}, B_i)$ such that $M(d_i) = d'_i$ and $\Psi_i = \Psi(d_i)$, then $M(d) = d'$ and $\Psi = \Psi(d)$ for $d = \rho(d_1, \ldots, d_k)$. Thus $L(G') \subseteq L(G)$, and d can be computed by an LDT-transducer. ∎

5. Lexicalization

In this section, in Lemma 42, we present the main lexicalization step, in which we lexicalize all non-monic non-terminal rules. It generalizes the transformation of a context-free grammar into Operator Normal Form (see [46, Theorem 1.2] and [3, Theorem 3.5]). We assume that G is Δ-growing (see Theorem 37). Thus, all non-initial terminal rules are doubly Δ-lexicalized and all monic rules are Δ-lexicalized. In the following we will simply write 'lexicalized' to mean 'Δ-lexicalized'.

For a derivation tree $d \in L(G_{\mathrm{der}})$ and a position $r \in \mathrm{pos}(d)$ such that $d(r)$ is a non-lexicalized rule of rank at least 2, we say that the "source" of r is the first position q in a pre-order traversal of the second direct subtree of r (i.e., the subtree at $r2$) such that $d(q)$ is a doubly lexicalized rule. Clearly, since every terminal rule at the leaves of d is doubly lexicalized, such a position exists and can be found by only exploring the first children of each visited node; i.e., $q = r21^m$ for some $m \in \mathbb{N}_0$. The basic idea of the lexicalization construction is to remove one lexical symbol δ from the source q and transport it to the "target" r. Then $d(q)$ is still lexicalized, and $d(r)$ has become lexicalized. Note that different targets have different sources, which is a simple fact that is well known to be useful (cf. [76, Section 3] and [47, page 346]). The transportation of δ from the source node q to the target node r is the task of the non-lexicalized or singly lexicalized rules at the positions along the path from q to r. The required relabeling of the derivation tree can be realized deterministically by an LDT^{R}-transducer that uses its look-ahead at r to determine the node label $d(q)$. From the rewriting point of view (Section 3.3), it is a guess-and-verify process. We guess δ at position r and verify it at position q.

Example 41 As before, let $\Delta = \{\alpha, \beta, \tau, \nu\}$. Since the resulting grammar G in Example 36 can be lexicalized by simple substitution of rules (as discussed in Example 36), we consider another Δ-growing grammar, which is similar to the original grammar of Example 7, but has an additional non-lexicalized rule $A \to B(\gamma(A))$. Moreover, we replace the rule $\rho_4 = B(x_1) \to x_1$ by the two doubly lexicalized rules $B(x_1) \to \sigma(x_1, \alpha T_1(\beta\sigma(\alpha T_2, \gamma T_3)))$ and $B(x_1) \to \sigma(x_1, \sigma(\tau, \nu))$, which are taken from Example 36. The (big) nonterminal B' remains an alias of B. The resulting Δ-growing MCFTG, which we again call G, has the following rules (renamed with respect to Example 7):

$$\begin{array}{lll}
\rho_1 \colon\ S \to \alpha A & \rho_2 \colon\ A \to T_1(\sigma(B(T_2), T_3)) & \rho_3 \colon\ A \to B(\gamma A) \\
\rho_4 \colon\ B(x_1) \to \sigma(B(x_1), B'(A)) & \rho_5 \colon\ B(x_1) \to \sigma(x_1, \alpha T_1(\beta\sigma(\alpha T_2, \gamma T_3))) & \rho_6 \colon\ B(x_1) \to \sigma(x_1, \sigma(\tau, \nu)) \\
\rho'_4 \colon\ B'(x_1) \to \sigma(B(x_1), B'(A)) & \rho'_5 \colon\ B'(x_1) \to \sigma(x_1, \alpha T_1(\beta\sigma(\alpha T_2, \gamma T_3))) & \rho'_6 \colon\ B'(x_1) \to \sigma(x_1, \sigma(\tau, \nu)) \\
 & \rho_7 \colon\ T \to (\alpha T_1(\beta x_1),\, \alpha T_2,\, \gamma T_3) & \rho_8 \colon\ T \to (x_1, \tau, \nu)
\end{array}$$

with $T = (T_1(x_1), T_2, T_3)$. Rule ρ_1 is singly lexicalized, whereas rules ρ_2, ρ_3, ρ_4, and ρ'_4 are non-lexicalized. The remaining rules are doubly lexicalized. We will remove the lexical symbol β or τ from each doubly

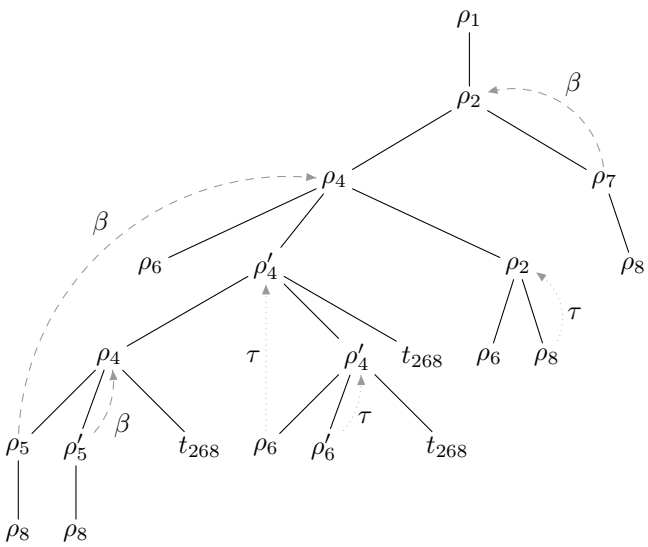


Figure 7: Derivation tree of $L(G_{\mathrm{der}})$ for the MCFTG G of Example 41 with indicated sources, targets, and transported lexical elements, where $t_{268} = \rho_2(\rho_6, \rho_8)$ with τ transported from ρ_8 to ρ_2.

lexicalized rule that labels a source and transport it to the target. For our derivation trees, we need to fix the order of the big nonterminals in the rules, so we let

$$\mathcal{L}(\rho_2) = \{B, (T_1, T_2, T_3)\} \qquad \mathcal{L}(\rho_3) = \{A, B\} \qquad \text{and} \qquad \mathcal{L}(\rho_4) = \mathcal{L}(\rho'_4) = \{B, B', A\} \ .$$

Figure 7 shows a derivation tree of $L(G_{\mathrm{der}})$ together with arrows indicating sources, corresponding targets, and transported lexical elements. A transportation of β is marked by a dashed arrow, whereas a transport of τ is marked by a dotted arrow. $\square$

We need some more terminology. Let Ω be a ranked alphabet (such as $N \cup \Sigma$) and let $X_\infty = X \setminus \{\square\}$; i.e., $X_\infty = \{x_1, x_2, \dots\}$. For a finite subset Z of X_∞, if $Z = \{x_{i_1}, \dots, x_{i_n}\}$ with $n \in \mathbb{N}_0$ and $i_1 < i_2 < \cdots < i_n$, then we define $\mathrm{seq}(Z) = x_{i_1} \cdots x_{i_n} \in X_\infty^*$, the sequence of variables in Z with increasing indices. A tree t in $T_\Omega(X)$ is *linear* if each variable occurs at most once in it; i.e., $|\mathrm{pos}_x(t)| \leq 1$ for every $x \in X$. For a linear tree $t \in T_\Omega(X)$, we denote by $\mathrm{var}(t)$ the set of variables x_i that occur in t; i.e., $\mathrm{var}(t) = \mathrm{occ}_{X_\infty}(t)$. If $\mathrm{seq}(\mathrm{var}(t)) = x_{i_1} \cdots x_{i_n}$, then we define $\mathrm{ren}(t) = t[x_{i_j} \leftarrow x_j \mid 1 \leq j \leq n]$, the *renumbering* of t, which is a pattern in $P_\Omega(X_n)$ if $\square$ does not occur in t. Note that $t = \mathrm{ren}(t)[x_j \leftarrow x_{i_j} \mid 1 \leq j \leq n]$. As an example, if $t = \sigma(x_4, \sigma(x_2, x_5))$ then $\mathrm{var}(t) = \{x_2, x_4, x_5\}$, $\mathrm{seq}(\mathrm{var}(t)) = x_2 x_4 x_5$, and $\mathrm{ren}(t) = \sigma(x_2, \sigma(x_1, x_3))$. We will use the easy fact that if h is a tree homomorphism over Ω and $t \in T_\Omega(X)$ is linear, then $\hat{h}(t)$ is linear and $\mathrm{var}(\hat{h}(t)) = \mathrm{var}(t)$ by Lemma 1(1), and $\mathrm{ren}(\hat{h}(t)) = \hat{h}(\mathrm{ren}(t))$ by Lemma 2.

To define contexts, we use the special variable $\square$. A *context* is a tree t with exactly one occurrence of $\square$; i.e., $|\mathrm{pos}_\square(t)| = 1$. For a linear context $t \in T_\Omega(X)$ we define $\mathrm{ren}_\square(t) = \mathrm{ren}(t)[\square \leftarrow x_{n+1}]$, where $n = |\mathrm{var}(t)|$. Note that $\mathrm{ren}_\square(t)$ is a pattern in $P_\Omega(X_{n+1})$. The above easy fact also holds for contexts: $\hat{h}(t)$ is a linear context and, by Lemma 2 again, $\mathrm{ren}_\square(\hat{h}(t)) = \hat{h}(\mathrm{ren}_\square(t))$.

For a tree $t \in T_\Omega(X_k)$ and a position $p \in \mathrm{pos}_\Omega(t)$, there exist a unique context $c \in T_\Omega(X_k \cup \{\square\})$ and a unique tree $u \in T_\Omega(X_k)$ such that $\mathrm{pos}_\square(c) = \{p\}$ and $t = c[\square \leftarrow u]$. The context c is called the *p-context* of t and denoted by $t|^p$, and the tree u is called the *subtree* of t at p and denoted by $t|_p$. If $p \in \mathrm{pos}_\omega(t)$ with $\mathrm{rk}(\omega) = m$, then $t = t|^p[\square \leftarrow \omega(t|_{p1}, \dots, t|_{pm})]$. Let h be a tree homomorphism over Ω. By Lemma 2, $\hat{h}(c[\square \leftarrow u]) = \hat{h}(c)[\square \leftarrow \hat{h}(u)]$. Thus, if $\mathrm{pos}_\square(\hat{h}(t|^p)) = \{\hat{p}\}$, then $\hat{h}(t|^p) = \hat{h}(t)|^{\hat{p}}$ and $\hat{h}(t|_p) = \hat{h}(t)|_{\hat{p}}$. Moreover, if $p \in \mathrm{pos}_\omega(t)$ and $h(\omega) = \mathrm{in}(\omega)$, then $\hat{p} \in \mathrm{pos}_\omega(\hat{h}(t))$ and $\hat{h}(t|_{pi}) = \hat{h}(t)|_{\hat{p}i}$ for every $i \in [m]$.

Lemma 42 *For every Δ-growing MCFTG G there is an* $\mathrm{LDT}^{\mathrm{R}}$*-equivalent Δ-lexicalized MCFTG G'.*

Proof Let $G = (N, \mathcal{N}, \Sigma, S, R)$ be a Δ-growing MCFTG. We can assume that all its terminal rules are doubly lexicalized because initial terminal rules can be removed from G and added after lexicalization. Moreover, for technical convenience, we assume that there is a subset Δ_{dl} of Δ such that (1) for every doubly lexicalized rule $A \to (u, \mathcal{L})$ there is a lexical symbol $\delta \in \Delta_{\mathrm{dl}}$ that occurs exactly once in u, and (2) for every singly lexicalized rule $A \to (u, \mathcal{L})$, the lexical symbol that occurs in u is not an element

of Δ_{dl}. This can be assumed because we could even assume that G is uniquely terminal labeled as defined after Lemma 21. In fact, as observed there, G has a cover (G_{u}, h) such that $G_{\mathrm{u}} = (N, \mathcal{N}, \Sigma_{\mathrm{u}}, S, R_{\mathrm{u}})$ is uniquely terminal labeled. If we let $\Delta_{\mathrm{u}} = \{\sigma \in \Sigma_{\mathrm{u}} \mid h(\sigma) \in \Delta\}$, then G_{u} is Δ_{u}-growing. Let G'_{u} be a Δ_{u}-lexicalized MCFTG that is LDT^{R}-equivalent to G_{u}, and let $G' = (G'_{\mathrm{u}})_h$; i.e., G' is the unique MCFTG such that (G'_{u}, h) is a cover of G'. Then G' is Δ-lexicalized. Moreover, G is LDT^{R}-$\hat{h}$-equivalent to G_{u} and G' is LDT^{R}-$\hat{h}$-equivalent to G'_{u}, by Lemma 21. Consequently, we can conclude that G and G' are LDT^{R}-equivalent. This shows that we could even assume that G is uniquely terminal labeled. However we do not do so, because we wish to illustrate the construction in this proof on the grammar G of Example 41, for which $\Delta_{\mathrm{dl}} = \{\beta, \tau\}$.

For every doubly lexicalized rule $\rho = A \to (u, \mathcal{L})$ of G, let $\mathrm{lex}(\rho) \in \Delta_{\mathrm{dl}}$ be a fixed lexical symbol that occurs exactly once in u. In the grammar G' to be constructed, this symbol will possibly be removed from u, leaving a rule that is still lexicalized.

We let

$$N_{\mathrm{new}} = \{\langle C, \delta, i, Z\rangle \mid C \in N,\ \delta \in \Delta_{\mathrm{dl}},\ 0 \le i \le \mathrm{rk}(\delta),\ Z \subseteq X_{\mathrm{rk}(C)}\}$$

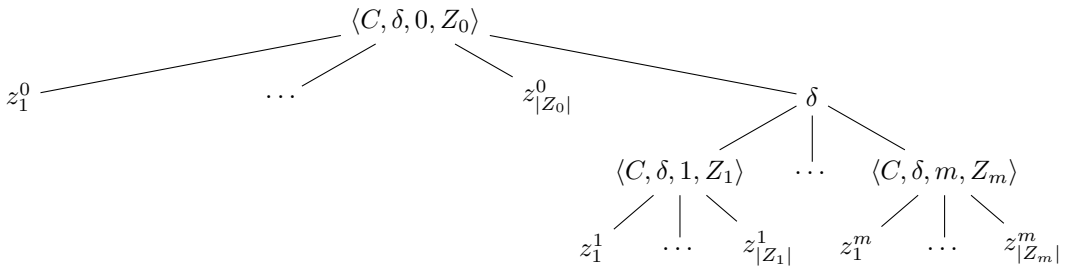
be a set of new nonterminals such that $\mathrm{rk}(\langle C, \delta, 0, Z\rangle) = |Z| + 1$ and $\mathrm{rk}(\langle C, \delta, i, Z\rangle) = |Z|$ for every $i \in [\mathrm{rk}(\delta)]$. The grammar G' will have the set of nonterminals $N' = N \cup N_{\mathrm{new}}$.

Let us provide some intuition for these new nonterminals. We first observe that for every derivation tree $d \in L(G_{\mathrm{der}}, A)$ there is a natural label-preserving bijection τ_d between the sets $\mathrm{pos}_\Sigma(\mathrm{val}(d))$ and $\bigcup_{q \in \mathrm{pos}(d)} \big(\{q\} \times \mathrm{pos}_\Sigma(\mathrm{rhs}(d(q)))\big)$; i.e., between the set of terminal positions of $\mathrm{val}(d)$ and the disjoint union of the sets of terminal positions of the right-hand sides of the rules that occur in d, cf. Lemma 10(1). For positions $q \in \mathrm{pos}(d)$ and $p \in \mathrm{pos}_\Sigma(\mathrm{rhs}(d(q)))$, let $\tau_d(q, p)$ be the corresponding position in $\mathrm{pos}_\Sigma(\mathrm{val}(d))$. Since τ_d is only needed in this paragraph, we do not give its straightforward, but tedious, definition. The existence of τ_d should be intuitively clear, and can be proved by induction on the structure of d; the induction step is based on the fact that for a tree homomorphism h over $N \cup \Sigma$ and a forest u, there is a natural label-preserving bijection between the sets $\mathrm{pos}_\Sigma(\hat{h}(u))$ and $\bigcup_{q \in \mathrm{pos}(u)} \big(\{q\} \times \mathrm{pos}_\Sigma(h(u(q)))\big)$, cf. Lemma 1(2). Now, roughly speaking, the intuition for the new nonterminals is the following. Consider a derivation tree $d \in L(G_{\mathrm{der}}, A)$, and let q be the shortest position of d of the form 1^m for some $m \in \mathbb{N}_0$ such that $\rho = d(q)$ is doubly lexicalized. Thus, q is a potential source for a target that has d as its second direct subtree (in some other derivation tree). Let $\mathrm{lex}(\rho) = \delta$, and let $p \in \mathrm{pos}_\Sigma(\mathrm{rhs}(\rho))$ be the unique δ-labeled position of the right-hand side of rule ρ. Moreover, suppose that the corresponding δ-labeled position $\tau_d(q, p)$ of $\mathrm{val}(d)$ belongs to the j-th tree t of the forest $\mathrm{val}(d)$ with $1 \le j \le |A|$; i.e., $\tau_d(q, p) = \#^{j-1}p'$ with $p' \in \mathrm{pos}(t)$. Let the nonterminal C be the j-th element of the big nonterminal A. Thus, C (as part of A) generates (in G) the terminal tree t. Then $\langle C, \delta, 0, Z_0\rangle$ generates (in G') the p'-context of t (with $\square$ at position p'), and $\langle C, \delta, i, Z_i\rangle$ generates the subtree of t at $p'i$ for every $i \in [\mathrm{rk}(\delta)]$. The sets Z_0 and Z_i consist of the variables that occur in that context and that subtree, so $Z_0 = \mathrm{var}(t|^{p'})$ and $Z_i = \mathrm{var}(t|_{p'i})$. To be more precise, $\langle C, \delta, 0, Z_0\rangle$ generates $\mathrm{ren}_\square(t|^{p'})$ and $\langle C, \delta, i, Z_i\rangle$ generates $\mathrm{ren}(t|_{p'i})$.

We now continue the formal proof. For a nonterminal $C \in N$, we say that the triple (C, δ, Z) is a *skeleton* of C if $\delta \in \Delta_{\mathrm{dl}}$ and $Z = (Z_0, Z_1, \ldots, Z_m)$, where $m = \mathrm{rk}(\delta)$ and $\{Z_0, Z_1, \ldots, Z_m\}$ is a partition of $X_{\mathrm{rk}(C)}$.[21] For such a skeleton, we will denote by $\mathrm{tree}(C, \delta, Z)$ the tree

$$\langle C, \delta, 0, Z_0\rangle\, \mathrm{seq}(Z_0)\ \delta(\langle C, \delta, 1, Z_1\rangle\, \mathrm{seq}(Z_1), \ldots, \langle C, \delta, m, Z_m\rangle\, \mathrm{seq}(Z_m))$$

of which we observe (for clearness sake) that it looks as follows:



[21] Recall from the beginning of Section 2 that we allow the empty set to be an element of a partition. Thus, we allow $Z_i = \emptyset$.

where $\mathrm{seq}(Z_i) = z_1^i \cdots z_{|Z_i|}^i$ for every $0 \le i \le m$. Note that $\mathrm{tree}(C,\delta,Z) \in P_{N_{\mathrm{new}}\cup\{\delta\}}(X_{\mathrm{rk}(C)})$. Moreover, we will denote $\mathrm{yd}_{N_{\mathrm{new}}}(\mathrm{tree}(C,\delta,Z))$ by $\mathrm{seq}(C,\delta,Z)$; i.e., $\mathrm{seq}(C,\delta,Z)$ is the sequence

$$\langle C,\delta,0,Z_0\rangle\langle C,\delta,1,Z_1\rangle\cdots\langle C,\delta,m,Z_m\rangle \ .$$

Obviously, the skeleton (C,δ,Z) can be reconstructed from $\mathrm{seq}(C,\delta,Z)$, and hence from $\mathrm{tree}(C,\delta,Z)$.

To motivate $\mathrm{tree}(C,\delta,Z)$ and $\mathrm{seq}(C,\delta,Z)$, we observe that for every pattern $t \in P_{N'\cup\Sigma}(X_{\mathrm{rk}(C)})$ and every δ-labeled position p' of t (i.e., $p' \in \mathrm{pos}_\delta(t)$), the pattern t can be decomposed as

$$t = \mathrm{tree}(C,\delta,Z)[\mathrm{seq}(C,\delta,Z) \leftarrow (t_0,t_1,\ldots,t_m)] \ ,$$

where $t_0 = \mathrm{ren}_\square(t|^{p'})$ is the renumbered p'-context and $Z_0 = \mathrm{var}(t|^{p'})$ is the set of its variables before renumbering, and moreover, for every $i \in [m]$, $t_i = \mathrm{ren}(t|_{p'i})$ is the renumbered subtree at $p'i$ and $Z_i = \mathrm{var}(t|_{p'i})$ is the set of its variables before renumbering. Intuitively, $\mathrm{tree}(C,\delta,Z)$ can be viewed as the "skeleton" of this decomposition, which was our reason to call (C,δ,Z) a skeleton of C.

We let $\mathcal{N}_{\mathrm{new}}$ be the new set of big nonterminals of the form $\beta\cdot\mathrm{seq}(C,\delta,Z)\cdot\gamma$, where $\beta C\gamma \in \mathcal{N}$ with $C \in N$ and $\beta,\gamma \in N^*$, and (C,δ,Z) is a skeleton of C. We now construct the new MCFTG $G' = (N',\mathcal{N}',\Sigma,S,R')$ with $N' = N \cup N_{\mathrm{new}}$ and $\mathcal{N}' = \mathcal{N}\cup\mathcal{N}_{\mathrm{new}}$. To define the set R' of rules of G', we first define an auxiliary MCFTG $G_+ = (N',\mathcal{N}',\Sigma,S,R\cup R_+)$ where R_+ is a set of new rules that, intuitively, realize the transport of a lexical symbol from a source to a target (but not yet its arrival at the target).

For every doubly lexicalized rule $\rho = A_1\cdots A_n \to ((u_1,\ldots,u_n),\mathcal{L})$ of G with $\mathcal{L} = \{B_1,\ldots,B_k\}$ (in that order), $A_i \in N$, and $u_i \in P_{N\cup\Sigma}(X_{\mathrm{rk}(A_i)})$, we define a skeleton $\kappa(\rho)$ and a new rule $\overline{\rho}$ in R_+ as follows. Let $\delta = \mathrm{lex}(\rho)$ and let $\#^{j-1}p$ be the unique δ-labeled position of $(u_1,\ldots,u_n)$, so $j \in [n]$ and $\mathrm{pos}_\delta(u_j) = \{p\}$. Moreover, let $u = u_j$, $\mathrm{rk}(\delta) = m$, and $Z = (Z_0,Z_1,\ldots,Z_m)$ with $Z_0 = \mathrm{var}(u|^p)$ and $Z_i = \mathrm{var}(u|_{pi})$ for every $i \in [m]$. Then we define $\kappa(\rho) = (A_j,\delta,Z)$. Note that $u \in P_{N\cup\Sigma}(X_{\mathrm{rk}(A_j)})$ and hence (A_j,δ,Z) is a skeleton of A_j. Additionally, we define the rule

$$\begin{aligned}\overline{\rho} = \quad & A_1\cdots A_{j-1}\cdot\mathrm{seq}(A_j,\delta,Z)\cdot A_{j+1}\cdots A_n \\ \to\ & ((u_1,\ldots,u_{j-1},v_0,v_1,\ldots,v_m,u_{j+1},\ldots,u_n),\ \mathcal{L}) \ ,\end{aligned}$$

where $v_0 = \mathrm{ren}_\square(u|^p)$ and $v_i = \mathrm{ren}(u|_{pi})$ for every $i \in [m]$ (and $\mathcal{L} = \{B_1,\ldots,B_k\}$, in the same order). Clearly, $\overline{\rho}$ is lexicalized because $|\mathrm{pos}_\Delta((u_1,\ldots,u_n))| \ge 2$ and $|\mathrm{pos}_\Delta((v_0,\ldots,v_m))| = |\mathrm{pos}_\Delta(u)| - 1$.

For every non-lexicalized or singly lexicalized rule $\rho = A_1\cdots A_n \to ((u_1,\ldots,u_n),\mathcal{L})$ of G with $\mathcal{L} = \{B_1,\ldots,B_k\}$ and $k \ge 1$, and for every skeleton (C,δ,W) such that $C \in \mathrm{occ}(B_1)$, we define a skeleton $\kappa(\rho,(C,\delta,W))$ and a new rule $\rho_{C,\delta,W}$ in R_+ as follows. Let $j \in [n]$ be the unique integer such that $C \in \mathrm{occ}_N(u_j)$, and let $u' = u_j[C \leftarrow \mathrm{tree}(C,\delta,W)]$. Moreover, let $\mathrm{rk}(\delta) = m$, $\mathrm{pos}_\delta(u') = \{p\}$,[22] and $Z = (Z_0,Z_1,\ldots,Z_m)$ with $Z_0 = \mathrm{var}(u'|^p)$ and $Z_i = \mathrm{var}(u'|_{pi})$ for every $i \in [m]$. Then we define $\kappa(\rho,(C,\delta,W)) = (A_j,\delta,Z)$. Let $B_1 = \beta C\gamma$ for some $\beta,\gamma \in N^*$, which are unique because B_1 is repetition-free. Then we define the rule

$$\begin{aligned}\rho_{C,\delta,W} = \quad & A_1\cdots A_{j-1}\cdot\mathrm{seq}(A_j,\delta,Z)\cdot A_{j+1}\cdots A_n \\ \to\ & ((u_1,\ldots,u_{j-1},v'_0,v'_1,\ldots,v'_m,u_{j+1},\ldots,u_n),\ \mathcal{L}') \ ,\end{aligned}$$

where $v'_0 = \mathrm{ren}_\square(u'|^p)$ and $v'_i = \mathrm{ren}(u'|_{pi})$ for every $i \in [m]$. Additionally, $\mathcal{L}' = \{B'_1,B_2,\ldots,B_k\}$ with $B'_1 = \beta\cdot\mathrm{seq}(C,\delta,W)\cdot\gamma$. Note that $\rho_{C,\delta,W}$ is non-lexicalized or singly lexicalized, respectively, because $|\mathrm{pos}_\Delta((v'_0,v'_1,\ldots,v'_m))| = |\mathrm{pos}_\Delta(u')| - 1 = |\mathrm{pos}_\Delta(u_j)|$.

These are all the rules of R_+. Thus, G_+ is the grammar obtained from G by adding all the above new rules $\overline{\rho}$ and $\rho_{C,\delta,W}$ to R. It is straightforward to check that from the rule $\overline{\rho}$ the original rule ρ can be reconstructed, and similarly, from $\rho_{C,\delta,W}$ we can reconstruct both ρ and (C,δ,W). Note that all terminal and all monic rules of G_+ are lexicalized.

We now define the set R' of rules of G'. First, R' contains all lexicalized rules of G_+. Second, we define rules that realize the arrival of a lexical symbol δ' at a target. Let $\rho = A \to (u,\mathcal{L})$ be a non-lexicalized rule of G_+ with $\mathcal{L} = \{B_1,\ldots,B_k\}$, where $k \ge 2$, $B_1 \in \mathcal{N}\cup\mathcal{N}_{\mathrm{new}}$, and $B_i \in \mathcal{N}$ for $2 \le i \le k$. For every skeleton (C',δ',Z) such that $C' \in \mathrm{occ}(B_2)$, we define the new rule $\langle\rho\rangle_{C',\delta',Z}$ in R' as follows.

[22] Note that by our second assumption on G, the symbol δ does not occur in u_j because $\delta \in \Delta_{\mathrm{dl}}$ and ρ is non-lexicalized or singly lexicalized.

Let $B_2 = \beta C'\gamma$ with $C' \in N$ and $\beta, \gamma \in N^*$, which are again unique because B_2 is repetition-free. Then $\langle\rho\rangle_{C',\delta',Z} = A \to (u', \mathcal{L}')$, where $u' = u[C' \leftarrow \mathrm{tree}(C', \delta', Z)]$ and $\mathcal{L}' = \{B_1, B_2', B_3, \ldots, B_k\}$ with $B_2' = \beta \cdot \mathrm{seq}(C', \delta', Z) \cdot \gamma$. Clearly, $\langle\rho\rangle_{C',\delta',Z}$ is lexicalized because δ' occurs in its right-hand side. It is easy to check that from the rule $\langle\rho\rangle_{C',\delta',Z}$ we can reconstruct both ρ and (C', δ', Z). Thus, R' consists of:

- all lexicalized rules ρ of G,
- all rules $\overline{\rho}$, where ρ is a doubly lexicalized rule of G,
- all rules $\rho_{C,\delta,W}$, where ρ is a singly lexicalized rule of G, and
- all rules $\langle\rho\rangle_{C',\delta',Z}$ and $\langle\rho_{C,\delta,W}\rangle_{C',\delta',Z}$, where ρ is a non-lexicalized rule of G.

This ends the construction of G'. It remains to show that G and G' are $\mathrm{LDT}^{\mathrm{R}}$-equivalent. We first show how to transform the derivation trees of G into those of G'. We start by defining a skeleton for every derivation tree of G.

For every derivation tree $d \in L(G_{\mathrm{der}}, A)$ we define a skeleton $\kappa(d) = (C, \delta, Z)$ with $C \in \mathrm{occ}(A)$ (and $\delta = \mathrm{lex}(\rho)$ for the label ρ of the shortest position of d of the form 1^m such that ρ is doubly lexicalized). The definition is by induction on the structure of $d = \rho(d_1, \ldots, d_k)$. If ρ is a doubly lexicalized rule (in particular if $k = 0$), then we define $\kappa(d) = \kappa(\rho)$ as defined above. Otherwise ρ is not doubly lexicalized (and so $k \geq 1$); let $\rho = A \to (u, \mathcal{L})$ with $\mathcal{L} = \{B_1, \ldots, B_k\}$. By the induction hypothesis we have $\kappa(d_1) = (C, \delta, W)$, where $C \in \mathrm{occ}(B_1)$. Then we define $\kappa(d) = \kappa(\rho, (C, \delta, W))$ as defined above. Clearly, for every skeleton (C, δ, Z), the set of derivation trees

$$L_{C,\delta,Z} = \{d \in \bigcup_{A \in \mathcal{N}} L(G_{\mathrm{der}}, A) \mid \kappa(d) = (C, \delta, Z)\}$$

is a regular tree language, which can be recognized by a deterministic bottom-up finite tree automaton using all skeletons as states.

For every derivation tree $d \in L(G_{\mathrm{der}}, A)$ we define two derivation trees $\mathrm{dtr}_1(d)$ and $\mathrm{dtr}_2(d)$ of G' with $\mathrm{dtr}_1(d) \in L(G'_{\mathrm{der}}, A)$ and $\mathrm{dtr}_2(d) \in L(G'_{\mathrm{der}}, \beta \cdot \mathrm{seq}(C, \delta, Z) \cdot \gamma)$, where $\kappa(d) = (C, \delta, Z)$ and $A = \beta C \gamma$ with $\beta, \gamma \in N^*$. These two derivation trees are relabelings of d. They are defined by induction on the structure of $d = \rho(d_1, \ldots, d_k)$.

- If ρ is a doubly lexicalized rule (in particular, if $k = 0$), then we define

$$\begin{aligned}\mathrm{dtr}_1(d) &= \rho(\mathrm{dtr}_1(d_1), \ldots, \mathrm{dtr}_1(d_k))\\ \mathrm{dtr}_2(d) &= \overline{\rho}(\mathrm{dtr}_1(d_1), \ldots, \mathrm{dtr}_1(d_k))\ .\end{aligned}$$

- Now let $\rho = A \to (u, \mathcal{L})$ be a rule with $\mathcal{L} = \{B_1, \ldots, B_k\}$ that is not doubly lexicalized (and hence $k \geq 1$). Moreover, let $\kappa(d_1) = (C, \delta, W)$, where $C \in \mathrm{occ}(B_1)$.
  - If ρ is singly lexicalized, then we define

$$\begin{aligned}\mathrm{dtr}_1(d) &= \rho(\mathrm{dtr}_1(d_1), \ldots, \mathrm{dtr}_1(d_k))\\ \mathrm{dtr}_2(d) &= \rho_{C,\delta,W}(\mathrm{dtr}_2(d_1), \mathrm{dtr}_1(d_2), \ldots, \mathrm{dtr}_1(d_k))\ .\end{aligned}$$

  - If ρ is non-lexicalized, and thus $k \geq 2$, then we let $\kappa(d_2) = (C', \delta', Z)$, where $C' \in \mathrm{occ}(B_2)$, and we define

$$\begin{aligned}\mathrm{dtr}_1(d) &= \langle\rho\rangle_{C',\delta',Z}(\mathrm{dtr}_1(d_1), \mathrm{dtr}_2(d_2), \mathrm{dtr}_1(d_3), \ldots, \mathrm{dtr}_1(d_k))\\ \mathrm{dtr}_2(d) &= \langle\rho_{C,\delta,W}\rangle_{C',\delta',Z}(\mathrm{dtr}_2(d_1), \mathrm{dtr}_2(d_2), \mathrm{dtr}_1(d_3), \ldots, \mathrm{dtr}_1(d_k))\ .\end{aligned}$$

Clearly, there is an $\mathrm{LDT}^{\mathrm{R}}$-transducer M that transforms $d \in L(G_{\mathrm{der}})$ into $\mathrm{dtr}_1(d) \in L(G'_{\mathrm{der}})$. It has states q_1 and q_2 with initial state q_1, and it uses the regular tree languages $L_{C,\delta,Z}$ as look-ahead. It has the following rules, corresponding directly to the above definitions, where $\langle q_1, y_i \cdots y_k\rangle$ abbreviates $\langle q_1, y_i\rangle, \ldots, \langle q_1, y_k\rangle$ for $i \in [k]$:

- for every doubly lexicalized rule ρ

$$\begin{aligned}\langle q_1,\ \rho(y_1, \ldots, y_k)\rangle &\to \rho(\langle q_1, y_1 \cdots y_k\rangle)\\ \langle q_2,\ \rho(y_1, \ldots, y_k)\rangle &\to \overline{\rho}(\langle q_1, y_1 \cdots y_k\rangle)\end{aligned}$$

- for every singly lexicalized rule ρ and every skeleton (C, δ, W)

$$\begin{aligned}\langle q_1,\ \rho(y_1, \ldots, y_k)\rangle &\to \rho(\langle q_1, y_1 \cdots y_k\rangle)\\ \langle q_2,\ \rho(y_1 \colon L_{C,\delta,W}, y_2, \ldots, y_k)\rangle &\to \rho_{C,\delta,W}(\langle q_2, y_1\rangle, \langle q_1, y_2 \cdots y_k\rangle)\end{aligned}$$

- and for every non-lexicalized rule ρ and all skeletons (C', δ', Z) and (C, δ, W)

$$\langle q_1,\, \rho(y_1, y_2 \colon L_{C',\delta',Z}, y_3, \ldots, y_k)\rangle \to \langle \rho\rangle_{C',\delta',Z}(\langle q_1, y_1\rangle, \langle q_2, y_2\rangle, \langle q_1, y_3 \cdots y_k\rangle)$$
$$\langle q_2,\, \rho(y_1 \colon L_{C,\delta,W}, y_2 \colon L_{C',\delta',Z}, y_3, \ldots, y_k)\rangle \to \langle \rho_{C,\delta,W}\rangle_{C',\delta',Z}(\langle q_2, y_1\rangle, \langle q_2, y_2\rangle, \langle q_1, y_3 \cdots y_k\rangle) \ .$$

We will prove below that d and $\mathrm{dtr}_1(d)$ have the same value. However, to express the relationship between $\mathrm{val}(d)$ and $\mathrm{val}(\mathrm{dtr}_2(d))$, we need the following definition. Let $A \in \mathcal{N}$ be a big nonterminal and (C, δ, Z) be a skeleton such that $A = \beta C \gamma$ for some $\beta, \gamma \in N^*$. Moreover, let s and s' be forests in $P_\Sigma(X)^+$ such that $\mathrm{rk}(s) = \mathrm{rk}(A)$ and $s = \zeta t \eta$ for some $\zeta, \eta \in P_\Sigma(X)^*$ with $|\zeta| = |\beta|$ and $t \in P_\Sigma(X_{\mathrm{rk}(C)})$. We note that β, γ, ζ, t, and η are unique given A, C, and s. We say that s' *decomposes* s *for* A *and* (C, δ, Z) if there exists a position $p' \in \mathrm{pos}_\delta(t)$ such that $s' = \zeta \cdot (t_0, t_1, \ldots, t_m) \cdot \eta$, where $m = \mathrm{rk}(\delta)$, $t_0 = \mathrm{ren}_\square(t|^{p'})$, $Z_0 = \mathrm{var}(t|^{p'})$, and $t_i = \mathrm{ren}(t|_{p'i})$ and $Z_i = \mathrm{var}(t|_{p'i})$ for every $i \in [m]$.

We now prove by induction on the structure of $d \in L(G_{\mathrm{der}}, A)$ that

(i) $\mathrm{val}(\mathrm{dtr}_1(d)) = \mathrm{val}(d)$ and

(ii) $\mathrm{val}(\mathrm{dtr}_2(d))$ decomposes $\mathrm{val}(d)$ for A and $\kappa(d)$.

Let $d = \rho(d_1, \ldots, d_k)$ and suppose that (i) and (ii) hold for $d_1, \ldots, d_k$.

- We first consider the case where ρ is *doubly lexicalized*. Since (i) is obvious from the definition of 'val' and by the induction hypotheses, it remains to prove (ii). Let ρ be as in the definition of $\overline{\rho}$, and let us adopt the terminology in that definition. Abbreviating $[B_i \leftarrow \mathrm{val}(d_i) \mid 1 \le i \le k]$ by $[f]$, we obtain that

$$\mathrm{val}(d) = (u_1, \ldots, u_{j-1}, u, u_{j+1}, \ldots, u_n)[f] = \zeta t \eta$$
$$\mathrm{val}(\mathrm{dtr}_2(d)) = (u_1, \ldots, u_{j-1}, v_0, v_1, \ldots, v_m, u_{j+1}, \ldots, u_n)[f] = \zeta \cdot (t_0, t_1, \ldots, t_m) \cdot \eta \ ,$$

  where the first equality in the second line uses the induction hypotheses and where we define $\zeta = (u_1, \ldots, u_{j-1})[f]$, $t = u[f]$, $\eta = (u_{j+1}, \ldots, u_n)[f]$, and $t_i = v_i[f]$ for every $0 \le i \le m$. We know that $v_0 = \mathrm{ren}_\square(u|^p)$, $Z_0 = \mathrm{var}(u|^p)$, and $v_i = \mathrm{ren}(u|_{pi})$ and $Z_i = \mathrm{var}(u|_{pi})$ for every $i \in [m]$. It remains to show that a position $p' \in \mathrm{pos}_\delta(t)$ exists with $t_0 = \mathrm{ren}_\square(t|^{p'})$, $Z_0 = \mathrm{var}(t|^{p'})$, and $t_i = \mathrm{ren}(t|_{p'i})$ and $Z_i = \mathrm{var}(t|_{p'i})$ for every $i \in [m]$. We select the unique position $p' \in \mathrm{pos}_\square((u|^p)[f])$. Then, using the easy facts that are stated before this lemma (for the tree homomorphism corresponding to $[f]$), we obtain that $(u|^p)[f] = u[f]|^{p'} = t|^{p'}$ with $p' \in \mathrm{pos}_\delta(t)$, and $(u|_{pi})[f] = u[f]|_{p'i} = t|_{p'i}$, and so

$$t_0 = v_0[f] = \mathrm{ren}_\square(u|^p)[f] = \mathrm{ren}_\square(u|^p[f]) = \mathrm{ren}_\square(t|^{p'})$$
$$Z_0 = \mathrm{var}(u|^p) = \mathrm{var}(u|^p[f]) = \mathrm{var}(t|^{p'}) \ ,$$

  and similarly for $t_i = v_i[f]$ and Z_i for every $i \in [m]$.
- Next we consider the case where ρ is *non-lexicalized*, and we prove (i). Let ρ be as in the definition of $\langle \rho\rangle_{C',\delta',Z}$ with $\kappa(d_2) = (C', \delta', Z)$, where $C' \in \mathrm{occ}(B_2)$, and let us adopt the terminology found there. By definition, $\mathrm{dtr}_1(d) = \langle \rho\rangle_{C',\delta',Z}(\mathrm{dtr}_1(d_1), \mathrm{dtr}_2(d_2), \mathrm{dtr}_1(d_3), \ldots, \mathrm{dtr}_1(d_k))$. Hence,

$$\mathrm{val}(\mathrm{dtr}_1(d)) = u[C' \leftarrow \mathrm{tree}(C', \delta', Z)][B_1 B_2' B_3 \cdots B_k \leftarrow \mathrm{val}(d_1)\, \mathrm{val}(\mathrm{dtr}_2(d_2))\, \mathrm{val}(d_3) \cdots \mathrm{val}(d_k)] \ .$$

  We know that $B_2 = \beta C' \gamma$ and $B_2' = \beta \cdot \mathrm{seq}(C', \delta', Z) \cdot \gamma$. Let $\mathrm{val}(d_2) = \zeta t \eta$ with $|\zeta| = |\beta|$. By (ii) for d_2, there exists $p' \in \mathrm{pos}_\delta(t)$ such that $\mathrm{val}(\mathrm{dtr}_2(d_2)) = \zeta \cdot (t_0, t_1, \ldots, t_m) \cdot \eta$, where $m = \mathrm{rk}(\delta)$, $t_0 = \mathrm{ren}_\square(t|^{p'})$, $Z_0 = \mathrm{var}(t|^{p'})$, and $t_i = \mathrm{ren}(t|_{p'i})$ and $Z_i = \mathrm{var}(t|_{p'i})$ for every $i \in [m]$. By Lemmas 4(2) and 4(4), we now obtain that

$$\mathrm{val}(\mathrm{dtr}_1(d)) = u\big[C' \leftarrow \mathrm{tree}(C', \delta', Z)[\mathrm{seq}(C', \delta', Z) \leftarrow (t_0, t_1, \ldots, t_m)]\big]\,[g] \ ,$$

  where $[g] = [B_1 \cdot \beta\gamma \cdot B_3 \cdots B_k \leftarrow \mathrm{val}(d_1) \cdot \zeta\eta \cdot \mathrm{val}(d_3) \cdots \mathrm{val}(d_k)]$. As observed earlier (in the paragraph after the definition of 'tree' and 'seq'),

$$\mathrm{tree}(C', \delta', Z)[\mathrm{seq}(C', \delta', Z) \leftarrow (t_0, t_1, \ldots, t_m)] = t$$

  and so, again by Lemmas 4(2) and 4(4),

$$\mathrm{val}(\mathrm{dtr}_1(d)) = u[C' \leftarrow t]\,[g] = u[B_1 \cdot \beta C' \gamma \cdot B_3 \cdots B_k \leftarrow \mathrm{val}(d_1) \cdot \zeta t \eta \cdot \mathrm{val}(d_3) \cdots \mathrm{val}(d_k)] \ ,$$

  which equals $u[B_i \leftarrow \mathrm{val}(d_i) \mid 1 \le i \le k] = \mathrm{val}(d)$.

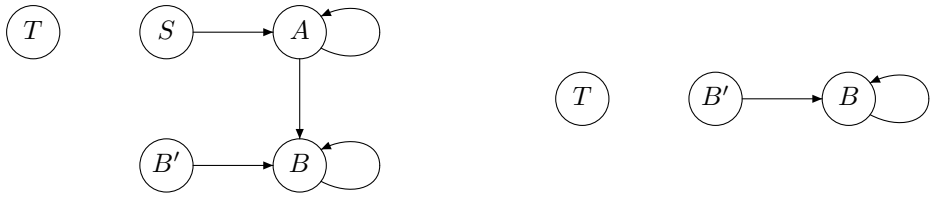


Figure 8: The graphs g [left] and g_{red} [right] constructed in Example 43.

- Next we consider the case where the rule ρ is *singly lexicalized*. Again, (i) is obvious, so it remains to prove (ii). Let ρ be as in the definition of $\rho_{C,\delta,W}$, and let us adopt the terminology there. Note that $\rho = A \to ((u_1, \ldots, u_n), \mathcal{L})$ with $\mathcal{L} = \{B_1, \ldots, B_k\}$ and $B_1 = \beta C \gamma$. Consider the auxiliary new rule $\rho' = A \to ((u_1, \ldots, u_{j-1}, u', u_{j+1}, \ldots, u_n), \mathcal{L}')$ with $\mathcal{L}' = \{B'_1, B_2, \ldots, B_k\}$ and $B'_1 = \beta \cdot \text{seq}(C, \delta, W) \cdot \gamma$. This rule ρ' is analogous to the rule $\langle \rho \rangle_{C,\delta,W}$, except that C occurs in B_1 instead of B_2 (and ρ is singly lexicalized instead of non-lexicalized). However, we can prove $\text{val}(d') = \text{val}(d)$ exactly as in the previous case, where $d' = \rho'(\text{dtr}_2(d_1), \text{dtr}_1(d_2), \ldots, \text{dtr}_1(d_k))$. Also, the rule $\rho_{C,\delta,W}$ is analogous to the rule $\overline{\rho'}$, if we define $\text{lex}(\rho') = \delta$. In the first (doubly lexicalized) case we have shown that the value of $\overline{\rho}(\text{dtr}_1(d_1), \ldots, \text{dtr}_1(d_k))$ decomposes the value of $d = \rho(d_1, \ldots, d_k)$ for A and $\kappa(d)$ under the assumption that $\text{dtr}_1(d_i)$ and d_i have the same value. In exactly the same way we can prove here that the value of $\rho_{C,\delta,W}(\text{dtr}_2(d_1), \text{dtr}_1(d_2), \ldots, \text{dtr}_1(d_k))$ decomposes the value of $d' = \rho'(\text{dtr}_2(d_1), \text{dtr}_1(d_2), \ldots, \text{dtr}_1(d_k))$ for A and $\kappa(d')$. In other words, $\text{val}(\text{dtr}_2(d))$ decomposes $\text{val}(d)$ for A and $\kappa(d')$. Since $\kappa(d') = \kappa(\rho') = \kappa(\rho, (C, \delta, W))$, which in turn equals $\kappa(d)$, this proves (ii).
- It remains to prove (ii) in the case where the rule ρ is *non-lexicalized*. We now apply the argument that we used to prove (i) to the rule $\rho_{C,\delta,W}$ instead of ρ. For $\rho_{C,\delta,W}$ we obtain from the previous case (even though ρ is a non-lexicalized rather than a singly lexicalized rule) that the value of

$$\rho_{C,\delta,W}(\text{dtr}_2(d_1), \text{dtr}_1(d_2), \ldots, \text{dtr}_1(d_k))$$

decomposes $\text{val}(d)$ for A and $\kappa(d)$. From the argument for (i) we obtain that the value of

$$\langle \rho_{C,\delta,W} \rangle_{C',\delta',Z}(\text{dtr}_2(d_1), \text{dtr}_2(d_2), \text{dtr}_1(d_3), \ldots, \text{dtr}_1(d_k))$$

equals the value of $\rho_{C,\delta,W}(\text{dtr}_2(d_1), \text{dtr}_1(d_2), \ldots, \text{dtr}_1(d_k))$, hence $\text{val}(\text{dtr}_2(d))$ decomposes $\text{val}(d)$ for A and $\kappa(d)$.

This concludes the proof that $L(G) \subseteq L(G')$. To prove the converse $L(G') \subseteq L(G)$, it is straightforward to check that, vice versa, (i) for every derivation tree $d' \in L(G'_{\text{der}}, A)$ there is a derivation tree $d \in L(G_{\text{der}}, A)$ with $\text{dtr}_1(d) = d'$, and (ii) for every derivation tree $d' \in L(G'_{\text{der}}, \beta \cdot \text{seq}(C, \delta, Z) \cdot \gamma)$ there is a derivation tree $d \in L(G_{\text{der}}, \beta C \gamma)$ with $\text{dtr}_2(d) = d'$ and $\kappa(d) = (C, \delta, Z)$. To be precise, in both cases d can be obtained from d' by changing every label $\overline{\rho}$, $\rho_{C,\delta,W}$, $\langle \rho \rangle_{C',\delta',Z}$, and $\langle \rho_{C,\delta,W} \rangle_{C',\delta',Z}$ into just ρ. Thus, it is obvious that d can be computed from d' by an LDT-transducer. Hence G and G' are LDT^{R}-equivalent.

Finally, we present a procedure that directly constructs the reduced version of G' provided that G is reduced. For a rule $\rho = A \to (u, \mathcal{L})$ with $\mathcal{L} = \{B_1, \ldots, B_k\}$, we define $\text{bign}_i(\rho) = B_i$ for $i \in [k]$ and $\text{bign}_0(\rho) = A$.

- Construct the set Target $\subseteq \mathcal{N}$ of all $\text{bign}_2(\rho)$, where ρ is a non-lexicalized rule.
- Construct the directed graph g with set $\mathcal{N}$ of nodes and with edges $\text{bign}_0(\rho) \to \text{bign}_1(\rho)$ for all non-lexicalized and singly lexicalized rules ρ, and let g_{red} be obtained from g by removing all nodes (and all incident edges) that are not reachable from a node in Target.
- Let Skel be a variable set of skeletons, which is initialized to $\emptyset$.
- Compute all rules $\overline{\rho}$ such that $\text{bign}_0(\rho)$ is a node of g_{red}, and add $\kappa(\rho)$ to Skel.
- Repeat the following subitem until Skel does not change any more:
  - compute all rules $\rho_{C,\delta,W}$ such that (C, δ, W) is in Skel and the edge $\text{bign}_0(\rho) \to \text{bign}_1(\rho)$ is in g_{red}, and add $\kappa(\rho, (C, \delta, W))$ to Skel.
- Finally, compute all rules $\langle \rho \rangle_{C',\delta',Z}$ such that (C', δ', Z) is in Skel, for the rules ρ obtained so far.

We leave the correctness of this procedure to the reader. ∎

Example 43 Let us lexicalize the new grammar G of Example 41, according to the construction in the proof of Lemma 42. We immediately construct the reduced version of G' with the procedure presented at the end of the proof of that lemma. Note that G satisfies the assumptions mentioned in the beginning of the proof for $\Delta_{\mathrm{dl}} = \{\beta, \tau\}$. For the doubly lexicalized rules ρ of G; i.e., for the rules

$$\begin{array}{llll} \rho_5\colon & B(x_1) \to \sigma(x_1, \alpha T_1(\underline{\beta}\sigma(\alpha T_2, \gamma T_3))) & \rho_6\colon & B(x_1) \to \sigma(x_1, \sigma(\underline{\tau}, \nu)) \\ \rho_7\colon & T \to (\alpha T_1(\underline{\beta} x_1),\, \alpha T_2,\, \gamma T_3) & \rho_8\colon & T \to (x_1, \underline{\tau}, \nu) \end{array}$$

(and the rules ρ_5' and ρ_6') we define $\mathrm{lex}(\rho) = \beta$ if β occurs in ρ, and $\mathrm{lex}(\rho) = \tau$ otherwise. We marked the lexical element in the rules by underlining it. We obtain that $\mathrm{Target} = \{T, B, B'\}$, where $T = (T_1, T_2, T_3)$. The graphs g and g_{red} are displayed in Figure 8. Since all doubly lexicalized rules ρ have their left-hand side in g_{red}, we construct the new rule $\overline{\rho}$ for each of them. We will use the following abbreviations for the new nonterminals

$$\begin{array}{llll} B_{\beta,0} = \langle B, \beta, 0, \{x_1\}\rangle & B_{\beta,1} = \langle B, \beta, 1, \emptyset\rangle & B_\tau = \langle B, \tau, 0, \{x_1\}\rangle & \text{of rank 2, 0, and 2, resp.} \\ T_{1,\beta,0} = \langle T_1, \beta, 0, \emptyset\rangle & T_{1,\beta,1} = \langle T_1, \beta, 1, \{x_1\}\rangle & T_{2,\tau} = \langle T_2, \tau, 0, \emptyset\rangle & \text{all of rank 1.} \end{array}$$

Then we obtain the new rules $\overline{\rho}_5$ and $\overline{\rho}_6$ on the first line and the rules $\overline{\rho}_7$ and $\overline{\rho}_8$ on the second line:

$$\begin{array}{rr} (B_{\beta,0}(x_1, x_2), B_{\beta,1}) \to (\sigma(x_1, \alpha T_1(x_2)),\, \sigma(\alpha T_2, \gamma T_3)) & B_\tau(x_1, x_2) \to \sigma(x_1, \sigma(x_2, \nu)) \\ (T_{1,\beta,0}(x_1), T_{1,\beta,1}(x_1), T_2, T_3) \to (\alpha T_1(x_1),\, x_1,\, \alpha T_2,\, \gamma T_3) & (T_1(x_1), T_{2,\tau}(x_1), T_3) \to (x_1, x_1, \nu)\ . \end{array}$$

The construction of the first new rule is illustrated in the top box of Figure 9. The rules $\overline{\rho}_5'$ and $\overline{\rho}_6'$ are obtained from $\overline{\rho}_5$ and $\overline{\rho}_6$ by changing every B into B'. Let $Z_\beta = (\{x_1\}, \emptyset)$, $Z_\tau = (\{x_1\})$, $Z_\beta' = (\emptyset, \{x_1\})$, and $Z_\tau' = (\emptyset)$. Then

$$\mathrm{Skel} = \{(B, \beta, Z_\beta),\, (B', \beta, Z_\beta),\, (B, \tau, Z_\tau),\, (B', \tau, Z_\tau),\, (T_1, \beta, Z_\beta'),\, (T_2, \tau, Z_\tau')\}\ .$$

The only non-lexicalized or singly lexicalized rules ρ with $\mathrm{bign}_0(\rho) \to \mathrm{bign}_1(\rho)$ in g_{red} are the rules $\rho_4 = B(x_1) \to \sigma(B(x_1), B'(A))$ and the corresponding rule ρ_4' with left-hand side $B'(x_1)$. Since its first link is the nonterminal B, we construct the new rules $\rho_{C,\delta,W}$ for $(C, \delta, W) \in \{(B, \beta, Z_\beta), (B, \tau, Z_\tau)\} \subseteq \mathrm{Skel}$ and $\rho \in \{\rho_4, \rho_4'\}$. For the right-hand side u of ρ_4 (and ρ_4') we get

$$\begin{aligned} u[B \leftarrow \mathrm{tree}(B, \beta, Z_\beta)] &= u[B \leftarrow B_{\beta,0}(x_1, \beta(B_{\beta,1}))] = \sigma(B_{\beta,0}(x_1, \beta(B_{\beta,1})), B'(A)) \\ u[B \leftarrow \mathrm{tree}(B, \tau, Z_\tau)] &= u[B \leftarrow B_\tau(x_1, \tau)] = \sigma(B_\tau(x_1, \tau), B'(A))\ , \end{aligned}$$

and consequently we obtain the rules

$$\begin{aligned} (\rho_4)_{B,\beta,Z_\beta} &= (B_{\beta,0}(x_1, x_2), B_{\beta,1}) \to (\sigma(B_{\beta,0}(x_1, x_2), B'(A)),\, B_{\beta,1}) \\ (\rho_4)_{B,\tau,Z_\tau} &= \qquad B_\tau(x_1, x_2) \to \sigma(B_\tau(x_1, x_2), B'(A))\ , \end{aligned}$$

and similar rules for ρ_4'. The construction of the first rule is illustrated in the bottom box of Figure 9. Clearly, the set Skel does not change, so we do not have to repeat this step. In the final step we lexicalize the non-lexicalized (old and new) rules by substituting $\mathrm{tree}(C', \delta', Z)$ for a nonterminal C' of the second link of each rule. From $\rho_2 = A \to T_1(\sigma(B(T_2), T_3))$ we obtain the following two new rules, by substituting $\mathrm{tree}(T_1, \beta, Z_\beta') = T_{1,\beta,0}(\beta\, T_{1,\beta,1}(x_1))$ and $\mathrm{tree}(T_2, \tau, Z_\tau') = T_{2,\tau}(\tau)$ for T_1 and T_2 respectively:

$$\begin{aligned} \langle\rho_2\rangle_{T_1,\beta,Z_\beta'} &= A \to T_{1,\beta,0}(\beta\, T_{1,\beta,1}(\sigma(B(T_2), T_3))) \\ \langle\rho_2\rangle_{T_2,\tau,Z_\tau'} &= A \to T_1(\sigma(B(T_{2,\tau}(\tau)), T_3))\ . \end{aligned}$$

Moreover, from $\rho_3 = A \to B(\gamma A)$ and $\rho_4 = B(x_1) \to \sigma(B(x_1), B'(A))$ we obtain the new rules

$$\begin{aligned} \langle\rho_3\rangle_{B,\beta,Z_\beta} &= \qquad A \to B_{\beta,0}(\gamma A, \beta B_{\beta,1}) \\ \langle\rho_3\rangle_{B,\tau,Z_\tau} &= \qquad A \to B_\tau(\gamma A, \tau) \\ \langle\rho_4\rangle_{B',\beta,Z_\beta} &= B(x_1) \to \sigma(B(x_1), B'_{\beta,0}(A, \beta B'_{\beta,1})) \\ \langle\rho_4\rangle_{B',\tau,Z_\tau} &= B(x_1) \to \sigma(B(x_1), B'_\tau(A, \tau)) \end{aligned}$$

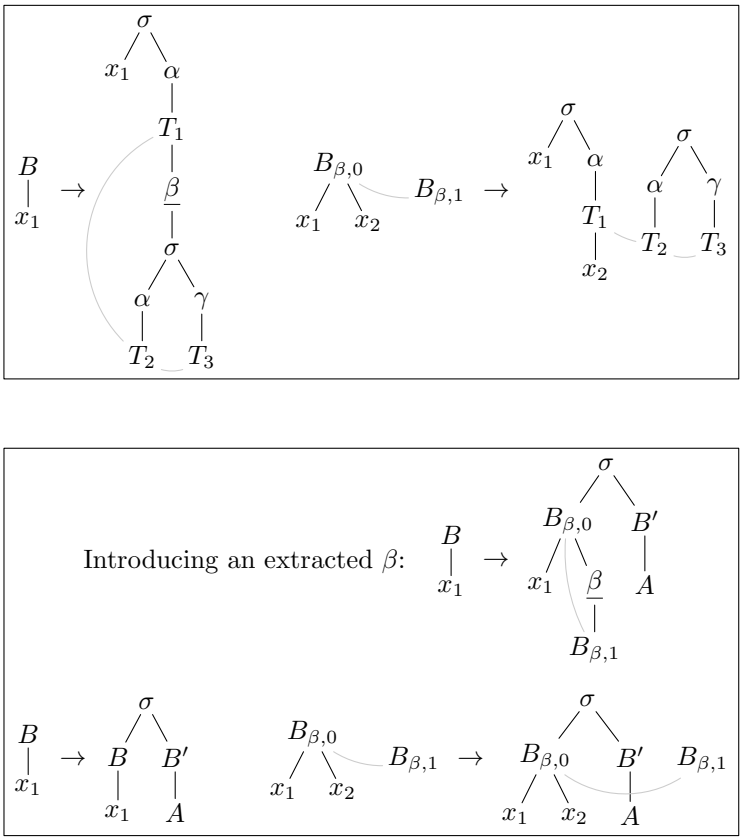


Figure 9: Illustration of the construction of the rule $\overline{\rho}_5$ extracting the underlined β [top box] by splitting the right-hand side into the parts above and below the extracted symbol. In the construction of the rule $(\rho_4)_{B,\beta,Z_\beta}$ [bottom box] we first introduce the lexical element β (replacing B) and the corresponding nonterminals [top rule] and then extract it again to obtain the final rule displayed at the bottom right.

and from the rules $(\rho_4)_{B,\beta,Z_\beta}$ and $(\rho_4)_{B,\tau,Z_\tau}$ we obtain

$$\begin{aligned}
\langle(\rho_4)_{B,\beta,Z_\beta}\rangle_{B',\beta,Z_\beta} &= (B_{\beta,0}(x_1,x_2), B_{\beta,1}) \to (\sigma(B_{\beta,0}(x_1,x_2), B'_{\beta,0}(A, \beta B'_{\beta,1})),\ B_{\beta,1}) \\
\langle(\rho_4)_{B,\beta,Z_\beta}\rangle_{B',\tau,Z_\tau} &= (B_{\beta,0}(x_1,x_2), B_{\beta,1}) \to (\sigma(B_{\beta,0}(x_1,x_2), B'_{\tau}(A, \tau)),\ B_{\beta,1}) \\
\langle(\rho_4)_{B,\tau,Z_\tau}\rangle_{B',\beta,Z_\beta} &= \qquad B_\tau(x_1,x_2) \to \sigma(B_\tau(x_1,x_2), B'_{\beta,0}(A, \beta B'_{\beta,1})) \\
\langle(\rho_4)_{B,\tau,Z_\tau}\rangle_{B',\tau,Z_\tau} &= \qquad B_\tau(x_1,x_2) \to \sigma(B_\tau(x_1,x_2), B'_{\tau}(A, \tau))
\end{aligned}$$

and similar rules for ρ'_4. The (reduced) lexicalized grammar G' has the rules

- ρ_1, ρ_5, ρ_6, ρ_7, ρ_8, $\overline{\rho}_5$, $\overline{\rho}_6$, $\overline{\rho}_7$, $\overline{\rho}_8$,
- $\langle\rho_2\rangle_{T_1,\beta,Z'_\beta}$, $\langle\rho_2\rangle_{T_2,\tau,Z'_\tau}$, $\langle\rho_3\rangle_{B,\beta,Z_\beta}$, $\langle\rho_3\rangle_{B,\tau,Z_\tau}$, $\langle\rho_4\rangle_{B',\beta,Z_\beta}$, $\langle\rho_4\rangle_{B',\tau,Z_\tau}$,
- $\langle(\rho_4)_{B,\beta,Z_\beta}\rangle_{B',\beta,Z_\beta}$, $\langle(\rho_4)_{B,\beta,Z_\beta}\rangle_{B',\tau,Z_\tau}$, $\langle(\rho_4)_{B,\tau,Z_\tau}\rangle_{B',\beta,Z_\beta}$, $\langle(\rho_4)_{B,\tau,Z_\tau}\rangle_{B',\tau,Z_\tau}$,

and the corresponding rules for ρ'_4, ρ'_5, and ρ'_6. Note that in all these rules, as in the grammar G of Example 7, there is only one possibility for the set of links $\mathcal{L}$. Note also that the left-hand sides of the primed rules are aliases of the left-hand sides of the nonprimed ones. We finally observe that rules $\langle\rho_2\rangle_{T_1,\beta,Z'_\beta}$ and $\overline{\rho}_7$ can be replaced by one rule $A \to \alpha T_1(\beta\sigma(B(\alpha T_2), \gamma T_3))$, and similarly $\langle\rho_2\rangle_{T_2,\tau,Z'_\tau}$ and $\overline{\rho}_8$ can be replaced by $A \to \sigma(B(\tau), \nu)$. In fact, these rules could have been obtained directly in the beginning as observed in Example 36. After this replacement, and disregarding the primed rules for aliases, the resulting lexicalized grammar has 17 rules.

Consider in the derivation tree d of Figure 7 the path from the root to the left-most leaf with label ρ_8. The sequence of node labels along this path is $(\rho_1, \rho_2, \rho_4, \rho'_4, \rho_4, \rho_5, \rho_8)$. In the derivation tree $\mathrm{dtr}_1(d)$ of G'_2 these nodes are relabeled to $(\rho_1, \langle\rho_2\rangle_{T_1,\beta,Z'_\beta}, \langle\rho_4\rangle_{B',\beta,Z_\beta}, \langle(\rho'_4)_{B,\beta,Z_\beta}\rangle_{B',\tau,Z_\tau}, \langle(\rho_4)_{B,\beta,Z_\beta}\rangle_{B',\beta,Z_\beta}, \overline{\rho}_5, \rho_8)$. $\square$

We now state the main theorem of this paper.

Theorem 44 *It is decidable for the MCFTG G whether or not G has finite Δ-ambiguity, and if so, there is a Δ-lexicalized MCFTG G' that is* $\mathrm{LDT}^{\mathrm{R}}$*-equivalent to G. Moreover, G' can be chosen such that $\theta(G') = \theta(G) + 1$ and $\mu(G') = \mu(G) + mrk_\Delta$.*[23]

Proof The first statement is immediate from Theorem 37 and Lemma 42. Since Theorem 37 preserves $\theta(G)$ and $\mu(G)$, it suffices to check that the construction in the proof of Lemma 42 satisfies the second statement. ∎

Note that if $\Delta \subseteq \Sigma^{(0)}$, then G' has the same multiplicity as G. Thus, as a corollary we obtain (a more specific version of) the main result of [70].

Corollary 45 *If we have $\Delta \subseteq \Sigma^{(0)}$, then Theorem 44 holds for spCFTG instead of MCFTG.*

Since every MCFTG has finite $(\Sigma^{(0)} \cup \Sigma^{(1)})$-ambiguity, we also obtain the following special case of Theorem 44.

Corollary 46 *For every MCFTG G there is an* $\mathrm{LDT}^{\mathrm{R}}$*-equivalent Σ-lexicalized MCFTG G' such that $\theta(G') = \theta(G) + 1$ and $\mu(G') = \mu(G) + 1$.*

It should be clear that Theorems 37 and 44 can be combined. If G has finite Δ-ambiguity, then there is an $\mathrm{LDT}^{\mathrm{R}}$-equivalent Δ-growing Δ-lexicalized MCFTG. Since every Δ-lexicalized MCFTG is almost Δ-growing, it suffices to apply once more the construction in the proof of Theorem 37 to the Δ-lexicalized MCFTG G' of Theorem 44.

It should even be clear that, by combining rules in a standard way, we can now ensure that every rule contains at least n lexical symbols for any $n \in \mathbb{N}$. This will be used in Section 6.3. Unfortunately, such a combination of rules cannot be realized by an $\mathrm{LDT}^{\mathrm{R}}$-transducer.[24] For every $n \geq 1$, let us say that a rule $A \to (u, \mathcal{L})$ of an MCFTG G is *n-Δ-lexicalized* if $|\mathrm{pos}_\Delta(u)| \geq n$, and that G is n-Δ-lexicalized if all its proper rules are n-Δ-lexicalized.

Lemma 47 *For every Δ-lexicalized MCFTG G and every $n \geq 1$ there is an n-Δ-lexicalized MCFTG G' such that $\theta(G') = \theta(G)$ and $\mu(G') = \mu(G)$.*

Proof The proof is by induction on n. For the induction step, let G be an n-Δ-lexicalized MCFTG. We may assume that all non-initial terminal rules of G are $(n+1)$-Δ-lexicalized because otherwise we can apply once more the construction in the proof of Theorem 37 for $\mathcal{F} = \{t \in P_\Sigma(X)^+ \mid n = |\mathrm{pos}_\Delta(t)|\}$. Moreover, we may assume that every big nonterminal $A \neq S$ has an alias $\bar{A}$ such that A and $\bar{A}$ do not occur together in any right-hand side of a rule. This can be achieved by introducing a new symbol $\bar{C}$ for every nonterminal C, and letting $\bar{A} = (\bar{A}_1, \ldots, \bar{A}_n)$ be an alias of $A = (A_1, \ldots, A_n)$.

Now let $G = (N, \mathcal{N}, \Sigma, S, R)$. We construct $G' = (N, \mathcal{N}, \Sigma, S, R')$, where R' is defined as follows. Let $\rho = A \to (u, \mathcal{L})$ be a rule in R with $\mathcal{L} = \{B_1, \ldots, B_k\}$ and $k \geq 1$, and let $\rho' = B_1 \to (u', \mathcal{L}')$ be a rule in R with left-hand side B_1 and $\mathcal{L}' = \{B'_1, \ldots, B'_\ell\}$. Let $u'' = u'[B'_i \leftarrow \mathrm{in}(\bar{B}'_i) \mid 1 \leq i \leq \ell]$. Then R' contains the rule $\langle \rho, \rho' \rangle = A \to (u[A_1 \leftarrow u''], \mathcal{L}'')$, where $\mathcal{L}'' = \{\bar{B}'_1, \ldots, \bar{B}'_\ell, B_2, \ldots, B_k\}$. Moreover, R' contains all terminal rules of R. Obviously, G' is $(n+1)$-Δ-lexicalized.

It is straightforward to prove that the derivation trees of G' are obtained from those of G by the value-preserving mapping M such that if $d = \rho(\rho'(d'_1, \ldots, d'_\ell), d_2, \ldots, d_k)$ then

$$M(d) = \langle \rho, \rho' \rangle(M(d'_1), \ldots, M(d'_\ell), M(d_2), \ldots, M(d_k)),$$

and if $d = \rho$ where ρ is a terminal rule then $M(d) = d$. Vice versa, the derivation trees of G are obtained from those of G' by the value-preserving tree homomorphism M' such that

$$M'(\langle \rho, \rho' \rangle) = \rho(\rho'(x_1, \ldots, x_\ell), x_{\ell+1}, \ldots, x_{\ell+k-1})$$

and $M'(\rho) = \rho$ for every terminal rule ρ. That proves that $L(G') = L(G)$. ∎

[23] Recall that mrk_Δ is the maximal rank of the symbols in Δ.

[24] It *can* be realized by a *finite-copying* deterministic top-down tree transducer with regular look-ahead.

6. MCFTG and MC-TAG

In this section we show that MC-TAGs have ("almost") the same tree generating power as MCFTGs. It is shown in [61] that non-strict tree adjoining grammars (nsTAGs) have the same tree generating power as monadic spCFTGs, where an spCFTG G is monadic if $\theta(G) \leq 1$; i.e., all its nonterminals have rank 1 or 0. In the first subsection we prove that MCFTGs have the same tree generating power as non-strict set-local multi-component tree adjoining grammars (nsMC-TAGs), generalizing the result of [61]. To avoid the introduction of the formal machinery that is needed to define nsMC-TAGs in the usual way, we define them to be "footed" MCFTGs, similar to the footed spCFTGs from [61]. As shown in [61, Section 4] for nsTAGs, the translation from one definition to the other is straightforward. In the second subsection we prove that MCFTGs have the same tree generating power as (strict) set-local multi-component tree adjoining grammars (MC-TAGs), where we define MC-TAGs as a special type of footed MCFTGs. The last result implies that MC-TAGs can be (strongly) lexicalized. It also implies, as shown in the third subsection, that MCFTGs have the same tree generating power as monadic MCFTGs (i.e., MCFTGs of width at most 1), which is essentially the same result as in [1, Theorem 3].[25] These results can be viewed as additional normal forms for MCFTGs.

Roughly speaking, the transformation of an MCFTG into an MC-TAG will be realized by decomposing each tree u_i in the right-hand side of a rule $A \to (u, \mathcal{L})$ with $A = (A_1, \ldots, A_n)$ and $u = (u_1, \ldots, u_n)$ into a bounded number of parts, to replace u_i in u by the sequence of these parts, and to replace A_i in A by a corresponding sequence of new nonterminals that simultaneously generate these parts. This is similar to the construction in the proof of Lemma 42 where, however, just one u_i was decomposed into parts.

6.1. Footed MCFTGs

Tree adjoining grammars (TAGs) are closely related to "footed" (simple) context-free tree grammars as shown in [61, Section 4]. An spCFTG is footed if for every rule $A(x_1, \ldots, x_k) \to u$ with $k \geq 1$ there is a node of u with exactly k children, which are labeled $x_1, \ldots, x_k$ from left to right. In other words, the arguments of A are passed in the same order to one node of u. In this section we generalize this notion to MCFTGs and prove that for every MCFTG there is an equivalent footed MCFTG.

Definition 48 Let $G = (N, \mathcal{N}, \Sigma, S, R)$ be an MCFTG. A pattern $t \in P_{N\cup\Sigma}(X_k)$ with $k \in \mathbb{N}_0$ is *footed* if either $k = 0$, or $k \geq 1$ and there exists a position $p \in \mathrm{pos}_{N\cup\Sigma}(t)$, called the *foot node* of t, such that $\mathrm{rk}(t(p)) = k$ and $t(pi) = x_i$ for every $i \in [k]$. A rule $\rho = A \to ((u_1, \ldots, u_n), \mathcal{L}) \in R$ is footed if u_j is footed for every $j \in [n]$. The MCFTG G is footed if every rule $\rho \in R$ is footed. □

Note that, by definition and for technical convenience, every tree $t \in T_{N\cup\Sigma} = P_{N\cup\Sigma}(X_0)$ is footed. The foot node of a footed pattern $t \in P_{N\cup\Sigma}(X_k)$ with $k \geq 1$ is obviously unique. If p is the foot node of t, then $t|_p = \mathrm{in}(t(p))$. It is straightforward to show, for a footed MCFTG G, that if $(t_1, \ldots, t_n) \in L(G, A)$, then t_j is footed for every $j \in [n]$. Assuming that G is reduced, this implies that $\theta(G) \leq mrk_\Sigma$. Moreover, G is permutation-free and nonerasing (cf. Lemmas 22 and 40).

Based on the close relationship between non-strict TAGs and footed context-free tree grammars as shown in [61, Section 4], we define a *non-strict tree adjoining grammar* (in short, nsTAG) to be a footed spCFTG, and similarly we define a *non-strict (set-local) multi-component TAG* (in short, nsMC-TAG) to be a footed MCFTG. This definition will be motivated after we have proved that for every MCFTG there is an equivalent footed MCFTG, which shows that MCFTGs and nsMC-TAGs have the same tree generating power.

It is shown in [61, Proposition 3] that every monadic nonerasing spCFTG can be transformed into an equivalent footed spCFTG. However, the proof of that proposition is not entirely correct, which can be seen from the following example. Consider the spCFTG G with rules $S \to A(e)$, $A(x_1) \to \sigma(A(x_1))$, and $A(x_1) \to \tau(a, x_1, b)$. Clearly, the last rule is not footed. In the proof of [61, Proposition 3] this grammar is transformed into the equivalent spCFTG G' with rules $S \to A(e)$, $S \to A'(T_1, e, T_3)$, $A(x_1) \to \sigma(A(x_1))$, $A(x_1) \to \sigma(A'(T_1, x_1, T_3))$, $A'(x_1, x_2, x_3) \to \tau(x_1, x_2, x_3)$, $T_1 \to a$, and $T_3 \to b$. However, the rule $A(x_1) \to \sigma(A'(T_1, x_1, T_3))$ is not footed, which is due to the fact that the foot node of the right-hand side $\sigma(A(x_1))$ of the second rule of G has a nonterminal label. The solution to this

[25] It is shown in [1, Theorem 3] that multi-parameter STTs (streaming tree transducers) have the same power as one-parameter STTs. Multi-parameter STTs are closely related to finite-copying macro tree transducers (cf. [1, Section 4.2]), and hence to MCFTGs as will be shown in Section 8. The number of parameters of the STT corresponds to the width of the MCFTG.

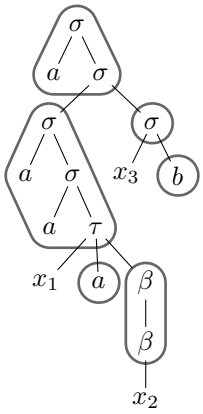


Figure 10: Decomposition into footed patterns.

problem is to introduce the nonterminals T_1 and T_3 in the first step of each derivation rather than in the last step. Thus, the footed spCFTG G'' with rules $S \to A'(T_1, e, T_3)$, $A'(x_1, x_2, x_3) \to \sigma(A'(x_1, x_2, x_3))$, $A'(x_1, x_2, x_3) \to \tau(x_1, x_2, x_3)$, $T_1 \to a$, and $T_3 \to b$ is equivalent to G. It is not difficult to repair the proof of [61, Proposition 3], but the construction becomes more complicated. We generalize that construction in the proof of the next theorem (without preserving the multiplicity, however). Since MRTGs are trivially footed, we restrict ourselves to MCFTGs G with $\theta(G) \geq 1$.

Theorem 49 *For every MCFTG G with $\theta(G) \geq 1$ there is an $\mathrm{LDT}^{\mathrm{R}}$-equivalent footed MCFTG G' such that $\mu(G') \leq \mu(G) \cdot mrk_\Sigma \cdot (2 \cdot \theta(G) - 1)$, where Σ is the terminal alphabet of G. Moreover, if G is Δ-lexicalized, then so is G'.*

Proof The basic idea of this proof is that, for any ranked alphabet Ω, every tree $u \in T_\Omega(X)$ with $u \notin X$ and $\mathrm{pos}_X(u) \neq \emptyset$ can be decomposed into at most $mrk_\Omega \cdot (2k-1)$ footed patterns, where $k = |\mathrm{pos}_X(u)|$. This can be understood as follows. Clearly, there are a unique $m \geq 1$, a unique footed pattern $u_\varepsilon \in P_\Omega(X_m)$, and unique trees $u_1, \ldots, u_m \in T_\Omega(X)$ such that $u = u_\varepsilon[x_i \leftarrow u_i \mid 1 \leq i \leq m]$ and $|\mathrm{pos}_X(u_i)| < |\mathrm{pos}_X(u)|$ for every $i \in [m]$ with $u_i \notin X$. In fact, the foot node of u_ε is the position p which, in u, is the least common Ω-labeled ancestor of the nodes in $\mathrm{pos}_X(u)$; i.e., the longest position such that $u(p) \in \Omega$ and $|\mathrm{pos}_X(u|_p)| = |\mathrm{pos}_X(u)|$. Note that the requirement $u(p) \in \Omega$ is only needed when $|\mathrm{pos}_X(u)| = 1$. Thus, we have decomposed u as $u_\varepsilon[x_i \leftarrow u_i \mid 1 \leq i \leq m]$ where u_ε is a footed pattern. For every $i \in [m]$ with $u_i \notin X$, either $u_i \in T_\Omega$ and so u_i is a footed pattern of rank 0, or $\mathrm{pos}_X(u_i) \neq \emptyset$ in which case u_i can be decomposed further. It should also be clear that, in this inductive process, there are at most $2k-1$ such foot node positions p. The factor mrk_Ω is due to the footed patterns of rank 0. As an example, consider the ranked alphabet $\Omega = \{\tau^{(3)}, \sigma^{(2)}, \beta^{(1)}, a^{(0)}, b^{(0)}\}$ and the tree $u = \sigma(a, \sigma(v, w))$ with $v = \sigma(a, \sigma(a, \tau(x_1, a, \beta(\beta(x_2)))))$ and $w = \sigma(x_3, b)$. For readability, let us use the notation $t_0[t_1, \ldots, t_n]$ for $t_0[x_i \leftarrow t_i \mid 1 \leq i \leq n]$. Then we obtain the decomposition $u = u_\varepsilon[u_1[x_1, u_{12}, u_{13}[x_2]], u_2[x_3, u_{22}]]$, illustrated in Figure 10, of u with the footed patterns $u_\varepsilon = \sigma(a, \sigma(x_1, x_2))$, $u_1 = \sigma(a, \sigma(a, \tau(x_1, x_2, x_3)))$, $u_{12} = a$, $u_{13} = \beta(\beta(x_1))$, $u_2 = \sigma(x_1, x_2)$, and $u_{22} = b$. Using new symbols C_p^m of rank m, with $p \in \mathbb{N}^*$, we can also express this as $u = K[\gamma]$ where K is the tree $C_\varepsilon^2(C_1^3(x_1, C_{12}^0, C_{13}^1(x_2)), C_2^2(x_3, C_{22}^0))$, which can be viewed as the skeleton of the decomposition, and γ is the second-order substitution such that $\gamma(C_p^m) = u_p$. A formal version of this decomposition is formulated below and applied to (a variant of) the trees in the right-hand sides of the rules of G. We note here that this decomposition is closely related to the one used in [67, Section 6] to turn a "straight-line" spCFTG into a monadic one.

Let $G = (N, \mathcal{N}, \Sigma, S, R)$ be an MCFTG with $\theta(G) \geq 1$. Then $mrk_\Sigma \cdot (2 \cdot \theta(G) - 1) \geq 1$, because $mrk_\Sigma \geq 1$ by Definition 5. By Lemmas 22 and 40 we may assume that G is permutation-free and nonerasing.[26] This means that if $(A_1, \ldots, A_n) \to ((u_1, \ldots, u_n), \mathcal{L})$ is a rule in R, then the pattern u_i is in $PF_{N \cup \Sigma}(X_{\mathrm{rk}(A_i)}) \setminus X$ for every $i \in [n]$.[27]

[26] First apply Lemma 40 and then Lemma 22. It is easy to check that Lemma 22 preserves the nonerasing and Δ-lexicalized properties.

[27] Recall from Lemma 22 that $PF_\Omega(X)$ denotes the set of permutation-free patterns over the ranked alphabet Ω. The requirement that $u_i \notin X$ is only relevant when $\mathrm{rk}(A_i) = 1$, meaning that $u_i \neq x_1$.

We define $G' = (N', \mathcal{N}', \Sigma, S', R')$. The set N' of nonterminals consists of all triples $\langle C, m, p\rangle$ with $C \in N$, $0 \le m \le mrk_\Sigma$, and $p \in \mathbb{N}^*$ such that $|p| \le \theta(G)$. The rank of $\langle C, m, p\rangle$ is m. The initial nonterminal is $S' = \langle S, 0, \varepsilon\rangle$. For every nonterminal $C \in N$, a *skeleton* of C is a permutation-free pattern $K \in PF_{N'}(X_{\mathrm{rk}(C)}) \setminus X$ such that[28]

(1) for every $p \in \mathrm{pos}_{N'}(K)$ there exists $0 \le m \le mrk_\Sigma$ such that $K(p) = \langle C, m, p\rangle$, and

(2) for every $p \in \mathbb{N}^*$ and $i \in \mathbb{N}$, if $pi \in \mathrm{pos}_{N'}(K)$ then $|\mathrm{pos}_X(K|_{pi})| < |\mathrm{pos}_X(K|_p)|$.

For such a skeleton K, we define $\mathrm{seq}(K) = \mathrm{yd}_{N'}(K)$, which is an element of $(N')^+$.[29] There are only finitely many skeletons of C. In fact, it is easy to show that $|\mathrm{pos}_{N'}(K)| \le mrk_\Sigma \cdot (2k-1)$ for every skeleton K of C, if $k = \mathrm{rk}(C) \ge 1$. Additionally, if $\mathrm{rk}(C) = 0$, then the only skeleton of C is $\langle C, 0, \varepsilon\rangle$. Note that K can be reconstructed from $\mathrm{seq}(K)$ because K is permutation-free. In the example above, the tree K is a skeleton of C, provided that C^m_p denotes $\langle C, m, p\rangle$, and $\mathrm{seq}(K) = (C^2_\varepsilon, C^3_1, C^0_{12}, C^1_{13}, C^2_2, C^0_{22})$.

We will apply the above basic idea to a pattern $u \in PF_{N'\cup\Sigma}(X_{\mathrm{rk}(C)}) \setminus X$. This leads to a decomposition of u that can be represented by a skeleton K of C and a substitution function γ such that $u = K[\gamma]$. This is formalized as follows. Let K be a skeleton of $C \in N$. A substitution function γ for $\mathrm{occ}_{N'}(K)$ is *footed* if, for every $C' \in \mathrm{occ}_{N'}(K)$, the pattern $\gamma(C') \in P_{N'\cup\Sigma}(X)$ is footed. We say that the pair $\langle K, \gamma\rangle$ is a *footed C-decomposition* of the tree $K[\gamma]$.

Basic fact. Every pattern u as above has a footed C-decomposition $\mathrm{dec}_C(u)$.[30] More precisely, for every $C \in N$ and every $u \in PF_{N'\cup\Sigma}(X_{\mathrm{rk}(C)}) \setminus X$ there is a pair $\mathrm{dec}_C(u) = \langle K, \gamma\rangle$ such that K is a skeleton of C, γ is a footed substitution function for $\mathrm{occ}_{N'}(K)$, and $K[\gamma] = u$.

Proof of the basic fact. To prove this by induction, we prove it for arbitrary $u \in T_{N'\cup\Sigma}(X)$ and we allow K to be an element of $T_{N'}(X)$ such that $\mathrm{yd}_X(K) = \mathrm{yd}_X(u)$. Obviously, if $K[\gamma] = u$ and u is a k-ary permutation-free pattern $\ne x_1$, then so is K.

If $u = x \in X$, then $\mathrm{dec}_C(u) = \langle K, \gamma\rangle$ with $K = x$ and γ is the empty function. If $u \in T_{N'\cup\Sigma}$, then $\mathrm{dec}_C(u) = \langle K, \gamma\rangle$ with $K = \langle C, 0, \varepsilon\rangle$ and $\gamma(\langle C, 0, \varepsilon\rangle) = u$. Now suppose that $u \notin X$ and $\mathrm{pos}_X(u) \ne \emptyset$. We proceed by induction on $|\mathrm{pos}_X(u)|$. Let the footed pattern u_ε in $P_{N'\cup\Sigma}(X_m)$ and the trees $u_1, \ldots, u_m$ in $T_{N'\cup\Sigma}(X)$ be as in the basic idea above, and let, by the induction hypotheses or by the previous two basic cases, $\mathrm{dec}_C(u_i) = \langle K_i, \gamma_i\rangle$ for every $i \in [m]$. Then $\mathrm{dec}_C(u) = \langle K, \gamma\rangle$, where K and γ are defined as follows. For every $i \in [m]$ let K'_i be obtained from K_i by changing every label $\langle C, m', p\rangle$ into $\langle C, m', ip\rangle$. Then K is the tree $K = \langle C, m, \varepsilon\rangle(K'_1, \ldots, K'_m)$. Moreover, the substitution function γ is defined by $\gamma(\langle C, m, \varepsilon\rangle) = u_\varepsilon$ and $\gamma(\langle C, m', ip\rangle) = \gamma_i(\langle C, m', p\rangle)$ for every $i \in [m]$ and every $\langle C, m', ip\rangle \in \mathrm{occ}_{N'}(K'_i)$. It is straightforward to verify that K and γ satisfy the requirements, which completes the proof of the basic fact.

We define the set $\mathcal{N}'$ of big nonterminals to consist of all sequences $\mathrm{seq}(K_1)\cdots\mathrm{seq}(K_n)$ for which there exists $(A_1, \ldots, A_n) \in \mathcal{N}$ such that K_j is a skeleton of A_j for every $j \in [n]$. A *skeleton function* for $A \in \mathcal{N}$ is a substitution function κ for $\mathrm{occ}(A)$ that assigns a skeleton $\kappa(C)$ of C to every nonterminal $C \in \mathrm{occ}(A)$. The string homomorphism h_κ from $\mathrm{occ}(A)$ to N' is defined by $h_\kappa(C) = \mathrm{seq}(\kappa(C))$ for every $C \in \mathrm{occ}(A)$. Note that $\mathcal{N}'$ is the set of all $h^*_\kappa(A)$, where $A \in \mathcal{N}$ and κ is a skeleton function for A.

We now define the set R' of rules. Let $\rho = A \to (u, \mathcal{L})$ be a rule in R such that $A = (A_1, \ldots, A_n)$, $u = (u_1, \ldots, u_n)$, and $\mathcal{L} = \{B_1, \ldots, B_k\}$. Moreover, let $\overline{\kappa} = (\kappa_1, \ldots, \kappa_k)$, where κ_i is a skeleton function for B_i for every $i \in [k]$. Intuitively, $\overline{\kappa}$ guesses for every nonterminal C that occurs in $B_1, \ldots, B_k$ the skeleton of a footed C-decomposition of the tree generated by C. Let f be the substitution function for $\mathrm{occ}_N(u)$ such that $f = \bigcup_{i\in[k]} \kappa_i$; i.e., $f(C) = \kappa_i(C)$ if $C \in \mathrm{occ}(B_i)$. It should be clear that $u_j[f] \in PF_{N'\cup\Sigma}(X_{\mathrm{rk}(A_j)}) \setminus X$ for every $j \in [n]$. For every $j \in [n]$, let $u'_j = u_j[f]$, let $\mathrm{dec}_{A_j}(u'_j) = \langle K_j, \gamma_j\rangle$ (the footed A_j-decomposition of u'_j according to the above basic fact), and let $v'_j = \gamma^*_j(\mathrm{seq}(K_j))$.[31] Then R' contains the rule

$$\langle \rho, \overline{\kappa}\rangle = \mathrm{seq}(K_1)\cdots\mathrm{seq}(K_n) \to (v'_1 \cdots v'_n, \mathcal{L}')$$

with $\mathcal{L}' = \{h^*_{\kappa_1}(B_1), \ldots, h^*_{\kappa_k}(B_k)\}$. We also define the skeleton function $\kappa_{\rho,\overline{\kappa}}$ for A by $\kappa_{\rho,\overline{\kappa}}(A_j) = K_j$ for every $j \in [n]$. Intuitively, K_j is the skeleton of a footed A_j-decomposition of the tree generated by A_j, resulting from the skeletons guessed by $\overline{\kappa}$. Note that the left-hand side of the rule $\langle \rho, \overline{\kappa}\rangle$ is $h^*_{\kappa_{\rho,\overline{\kappa}}}(A)$. This

[28] We usually do not denote trees with a capital, but k is already used for natural numbers.

[29] Recall the definition of $\mathrm{yd}_{N'}$ from the paragraph on homomorphisms in Section 2.1.

[30] The decomposition is even unique, but that will not be needed.

[31] Thus, if $\mathrm{seq}(K_j) = (C'_1, \ldots, C'_\ell)$ with $C'_1, \ldots, C'_\ell \in N'$, then $v'_j = (\gamma_j(C'_1), \ldots, \gamma_j(C'_\ell))$.

concludes the definition of G'. It should be clear that G' is footed. Moreover, since the right-hand sides of the rules ρ and $\langle \rho, \overline{\kappa} \rangle$ contain the same terminal symbols, G' is Δ-lexicalized if G is Δ-lexicalized. It remains to prove the correctness of G'.

For every derivation tree $d \in L(G_{\mathrm{der}}, A)$ we define a skeleton function κ_d for A and a derivation tree $q(d) \in L(G'_{\mathrm{der}}, h^*_{\kappa_d}(A))$ inductively as follows. If $d = \rho(d_1, \ldots, d_k)$ with the rule ρ as above, then we define $\kappa_d = \kappa_{\rho,\overline{\kappa}}$ and $q(d) = \langle \rho, \overline{\kappa} \rangle(q(d_1), \ldots, q(d_k))$, where $\overline{\kappa} = (\kappa_{d_1}, \ldots, \kappa_{d_k})$. We now claim the following.

Claim: For every $A = (A_1, \ldots, A_n) \in \mathcal{N}$ and every $d \in L(G_{\mathrm{der}}, A)$, if $\mathrm{val}(d) = (t_1, \ldots, t_n)$ then $K_j[h^*_{\kappa_d}(A) \leftarrow \mathrm{val}(q(d))] = t_j$, where $K_j = \kappa_d(A_j)$, for every $j \in [n]$.

Proof of Claim: Assume that $d = \rho(d_1, \ldots, d_k)$ as above, and that the claim holds for d_i for every $i \in [k]$. Let g be the substitution function for $\mathrm{occ}_N(u)$ such that $g(C)$ is the m-th element of $\mathrm{val}(d_i)$ if C is the m-th element of B_i. So, $\mathrm{val}(d) = u[B_i \leftarrow \mathrm{val}(d_i) \mid 1 \leq i \leq k] = u[g]$, and hence $u_j[g] = t_j$ for every $j \in [n]$. We write $[g']$ for the substitution $[h^*_{\kappa_{d_i}}(B_i) \leftarrow \mathrm{val}(q(d_i)) \mid 1 \leq i \leq k]$. Consequently, $\mathrm{val}(q(d)) = u'[g']$, where $u' = v'_1 \cdots v'_n$. We first show that $u_j[f][g'] = u_j[g]$ for every $j \in [n]$. By Lemma 4(4) it suffices to show that $f(C)[g'] = g(C)$ for every $C \in \mathrm{occ}_N(u)$. For every $C \in \mathrm{occ}(B_i)$ we obtain that $f(C)[g'] = \kappa_{d_i}(C)[h^*_{\kappa_{d_i}}(B_i) \leftarrow \mathrm{val}(q(d_i))]$, which equals $g(C)$ by the induction hypotheses. Now let $j \in [n]$. Then

$$K_j[h^*_{\kappa_d}(A) \leftarrow \mathrm{val}(q(d))] = K_j\big[\mathrm{seq}(K_j) \leftarrow v'_j[g']\big] = K_j\big[\mathrm{seq}(K_j) \leftarrow \gamma^*_j(\mathrm{seq}(K_j))[g']\big] \ .$$

By Lemma 4(4) this equals $K_j[\gamma_j][g']$. Since $\mathrm{dec}_{A_j}(u'_j) = \langle K_j, \gamma_j \rangle$, we obtain that

$$K_j[\gamma_j][g'] = u'_j[g'] = u_j[f][g'] = u_j[g] = t_j \ .$$

This proves the claim. Note that it provides a footed A_j-decomposition of t_j (in fact, the unique one).

In the case where $A = S$ we obtain that $\kappa_d(S) = \langle S, 0, \varepsilon \rangle$. Thus, $\mathrm{val}(q(d)) = \mathrm{val}(d)$ by the claim. Hence $L(G) \subseteq L(G')$. Clearly, for every skeleton function κ, the set L_κ of all derivation trees d with $\kappa_d = \kappa$ is a regular tree language, which can be computed by a deterministic bottom-up finite tree automaton that uses all skeleton functions as states. The $\mathrm{LDT}^{\mathrm{R}}$-transducer M that computes $q(d)$ from d has one state q, and it has the rules

$$\langle q, \rho(y_1 \colon L_{\kappa_1}, \ldots, y_k \colon L_{\kappa_k}) \rangle \to \langle \rho, \overline{\kappa} \rangle(\langle q, y_1 \rangle, \ldots, \langle q, y_k \rangle) \ ,$$

where $\overline{\kappa} = (\kappa_1, \ldots, \kappa_k)$. In the other direction, every derivation tree $d' \in L(G'_{\mathrm{der}})$ can be turned into a derivation tree $d = M'(d')$ in $L(G_{\mathrm{der}})$ by changing every label $\langle \rho, \overline{\kappa} \rangle$ into just ρ, and it is straightforward to show that $q(d) = d'$. This shows that $L(G') \subseteq L(G)$, and hence the correctness of G'. ∎

Example 50 Let $\Sigma = \{\tau^{(3)}, \ell^{(1)}, r^{(1)}, a^{(0)}, b^{(0)}, e^{(0)}\}$. Intuitively ℓ stands for a left parenthesis and r for a right parenthesis. We consider the footed spCFTG $G_1 = (N_1, \Sigma, S, R_1)$ with the set of nonterminals $N_1 = \{S, A, A'\}$, of which A has rank 1 and A' is an alias of A, and the rules

$$S \to \ell A(A'(re)) \qquad A(x_1) \to \ell A(A'(rx_1)) \qquad \text{and} \qquad A(x_1) \to \ell\tau(a, b, rx_1) \ ,$$

where we have omitted the rules with left-hand side $A'(x_1)$. Let $\Delta = \{a, b\}$. Since G_1 is Δ-growing, it has finite Δ-ambiguity. However, as we will show in Remark 53, there is no Δ-lexicalized footed spCFTG G with $L(G) = L(G_1)$. The basic reason for this is that the set $\{\mathrm{yd}_{\{\ell,r\}}(t) \mid t \in L(G_1)\} \subseteq \{\ell, r\}^*$ consists of all balanced strings of parentheses ℓ and r. In fact, G_1 is a straightforward variant of the TAG of [65], for which there is no (strongly) equivalent Δ-lexicalized TAG. Note that we defined nsTAGs to be footed spCFTGs. We will also show in Remark 53 that there is no Δ-lexicalized spCFTG G with $\theta(G) \leq 1$ that is equivalent to G_1.

From Corollary 45, we obtain a Δ-lexicalized spCFTG G_2 with $\theta(G_2) = 2$ that is equivalent to G_1. It has the new nonterminals $B = \langle A, b, 0, X_1 \rangle$ and $B' = \langle A', b, 0, X_1 \rangle$, where $\mathrm{rk}(B) = 2$ and B' is an alias of B. For the sake of readability we interchanged the two arguments of B (and those of B'), and similarly we used B instead of B' in the first two rules, so that A' has become superfluous. Its rules are

$$\rho_1 \colon\ S \to \ell A(B(b, re)) \qquad \rho_2 \colon\ A(x_1) \to \ell A(B(b, rx_1)) \qquad \rho_3 \colon\ A(x_1) \to \ell\tau(a, b, rx_1)$$
$$\rho_4 \colon\ B(x_1, x_2) \to \ell B(x_1, B'(b, rx_2)) \qquad \rho_5 \colon\ B(x_1, x_2) \to \ell\tau(a, x_1, rx_2) \ ,$$

plus the rules ρ_4' and ρ_5' for the alias B' of B. Clearly, the tree $B(b, x_1)$ generates the same terminal trees as $A(x_1)$. More precisely, if $A(x_1)$ generates the tree $\ell^n\tau(a, b, wx_1)$, where $n \in \mathbb{N}$ and $w \in \Sigma^*$, then $B(x_1, x_2)$ generates $\ell^n\tau(a, x_1, wx_2)$.

Rules ρ_4 and ρ_5 are not footed. We now turn G_2 into an equivalent Δ-lexicalized footed MCFTG G_2' using the construction in the proof of Theorem 49. For rule $\rho_5 = B(x_1, x_2) \to u_5$ and $\overline{\kappa} = \varepsilon$, we obtain the footed B-decomposition $\mathrm{dec}_B(u_5) = \langle K_5, \gamma_5\rangle$ such that $K_5 = B_0(B_1, x_1, B_3(x_2))$, where $B_0 = \langle B, 3, \varepsilon\rangle$, $B_1 = \langle B, 0, 1\rangle$, and $B_3 = \langle B, 1, 3\rangle$, and γ_5 is defined as follows: $\gamma_5(B_0) = \ell\tau(x_1, x_2, x_3)$, $\gamma_5(B_1) = a$, and $\gamma_5(B_3) = rx_1$. The resulting rule $\tilde{\rho}_5 = \langle\rho_5, \varepsilon\rangle$ is

$$\tilde{\rho}_5\colon \quad (B_0(x_1, x_2, x_3), B_1, B_3(x_1)) \to (\ell\tau(x_1, x_2, x_3), a, rx_1)$$

with left-hand side $\mathrm{seq}(K_5) = (B_0, B_1, B_3)$, and the corresponding skeleton function for B is $\kappa_5 = \kappa_{\rho_5,\varepsilon}$ such that $\kappa_5(B) = K_5$. The construction of this rule is illustrated in the first part of Figure 11. Of course we obtain similar primed results for B'. Taking $\overline{\kappa} = (\kappa_5, \kappa_5')$ and substituting K_5 for B and K_5' for B' in the right-hand side $u_4 = \ell B(x_1, B'(b, rx_2))$ of rule ρ_4, we obtain $u_4' = \ell B_0(B_1, x_1, B_3(B_0'(B_1', b, B_3'(rx_2))))$ which has the footed B-decomposition $\mathrm{dec}_B(u_4') = \langle K_4, \gamma_4\rangle$ where $K_4 = K_5$, $\gamma_4(B_0) = \ell B_0(x_1, x_2, x_3)$, $\gamma_4(B_1) = B_1$, and $\gamma_4(B_3) = B_3(B_0'(B_1', b, B_3'(rx_1)))$. The resulting rule $\tilde{\rho}_4 = \langle\rho_4, (\kappa_5, \kappa_5')\rangle$ is

$$\tilde{\rho}_4\colon \quad (B_0(x_1, x_2, x_3), B_1, B_3(x_1)) \to (\ell B_0(x_1, x_2, x_3),\ B_1,\ B_3(B_0'(B_1', b, B_3'(rx_1)))) \ .$$

Since the skeleton function $\kappa_{\rho_4,(\kappa_5,\kappa_5')}$ for B is again κ_5, these are all the necessary rules of G_2' with left-hand side (B_0, B_1, B_3), and similarly for (B_0', B_1', B_3'). The decomposition $\mathrm{dec}_A(u_3) = \langle K_3, \gamma_3\rangle$ of $u_3 = \ell\tau(a, b, rx_1)$ is simply $K_3 = \langle A, 1, \varepsilon\rangle(x_1)$ and $\gamma_3(\langle A, 1, \varepsilon\rangle) = u_3$. Thus, identifying $\langle A, 1, \varepsilon\rangle$ with A, grammar G_2' has the rule $\tilde{\rho}_3 = \rho_3$. Substituting K_3 for A and K_5 for B in the right-hand side u_2 of ρ_2 we obtain the tree $u_2' = \ell A(B_0(B_1, b, B_3(rx_1)))$ which, just as u_3, decomposes into itself. Thus, G_2' has the rule $\tilde{\rho}_2 = A(x_1) \to u_2'$. The construction of this rule is illustrated in the second part of Figure 11. Finally, by a similar process (identifying $\langle S, 0, \varepsilon\rangle$ with S), we obtain the rule $\tilde{\rho}_1 = S \to u_2'[x_1 \leftarrow e]$. Summarizing, G_2' has the nonterminals $\{S, A, B_0, B_0', B_1, B_1', B_3, B_3'\}$ and the big nonterminals $\{S, A, (B_0, B_1, B_3), (B_0', B_1', B_3')\}$. Its rules (apart from those for the alias (B_0', B_1', B_3')) are

$$\begin{aligned}
\tilde{\rho}_1\colon &\quad S \to \ell A(B_0(B_1, b, B_3(re))) \\
\tilde{\rho}_2\colon &\quad A(x_1) \to \ell A(B_0(B_1, b, B_3(rx_1))) \\
\tilde{\rho}_3\colon &\quad A(x_1) \to \ell\tau(a, b, rx_1) \\
\tilde{\rho}_4\colon &\quad (B_0(x_1, x_2, x_3), B_1, B_3(x_1)) \to (\ell B_0(x_1, x_2, x_3),\ B_1,\ B_3(B_0'(B_1', b, B_3'(rx_1)))) \\
\tilde{\rho}_5\colon &\quad (B_0(x_1, x_2, x_3)), B_1, B_3(x_1)) \to (\ell\tau(x_1, x_2, x_3), a, rx_1) \ .
\end{aligned}$$

To see that $L(G_2') = L(G_1)$ we observe that the tree $K_5 = B_0(B_1, x_1, B_3(x_2))$ generates the same terminal trees as $B(x_1, x_2)$ (as formalized in the Claim in the proof of Theorem 49), and hence $B_0(B_1, b, B_3(x_1))$ generates the same terminal trees as $A(x_1)$. □

Example 51 As another, very simple example we again consider the spCFTG G with the following rules

$$S \to A(e) \qquad A(x_1) \to \sigma(A(x_1)) \qquad \text{and} \qquad A(x_1) \to \tau(a, x_1, b) \ ,$$

which was also discussed before Theorem 49. The only skeleton of A needed by the equivalent footed MCFTG G' is $A_0(A_1, x_1, A_3)$ where $A_0 = \langle A, 3, \varepsilon\rangle$, $A_1 = \langle A, 0, 1\rangle$, and $A_3 = \langle A, 0, 3\rangle$. Its big nonterminals are $S' = \langle S, 0, \varepsilon\rangle$ and (A_0, A_1, A_3), and its rules are

$$\begin{aligned}
S' &\to A_0(A_1, e, A_3) \\
(A_0(x_1, x_2, x_3), A_1, A_3) &\to (\sigma(A_0(x_1, x_2, x_3)), A_1, A_3) \\
(A_0(x_1, x_2, x_3), A_1, A_3) &\to (\tau(x_1, x_2, x_3), a, b) \ .
\end{aligned}$$

Note that G' is not an spCFTG. □

Let us now discuss set-local multi-component tree adjoining grammars (MC-TAGs). In the beginning of this subsection we have defined a *non-strict* MC-TAG (nsMC-TAG) to be a footed MCFTG. To convince the reader familiar with TAGs we add some more terminology, which should make this clear.

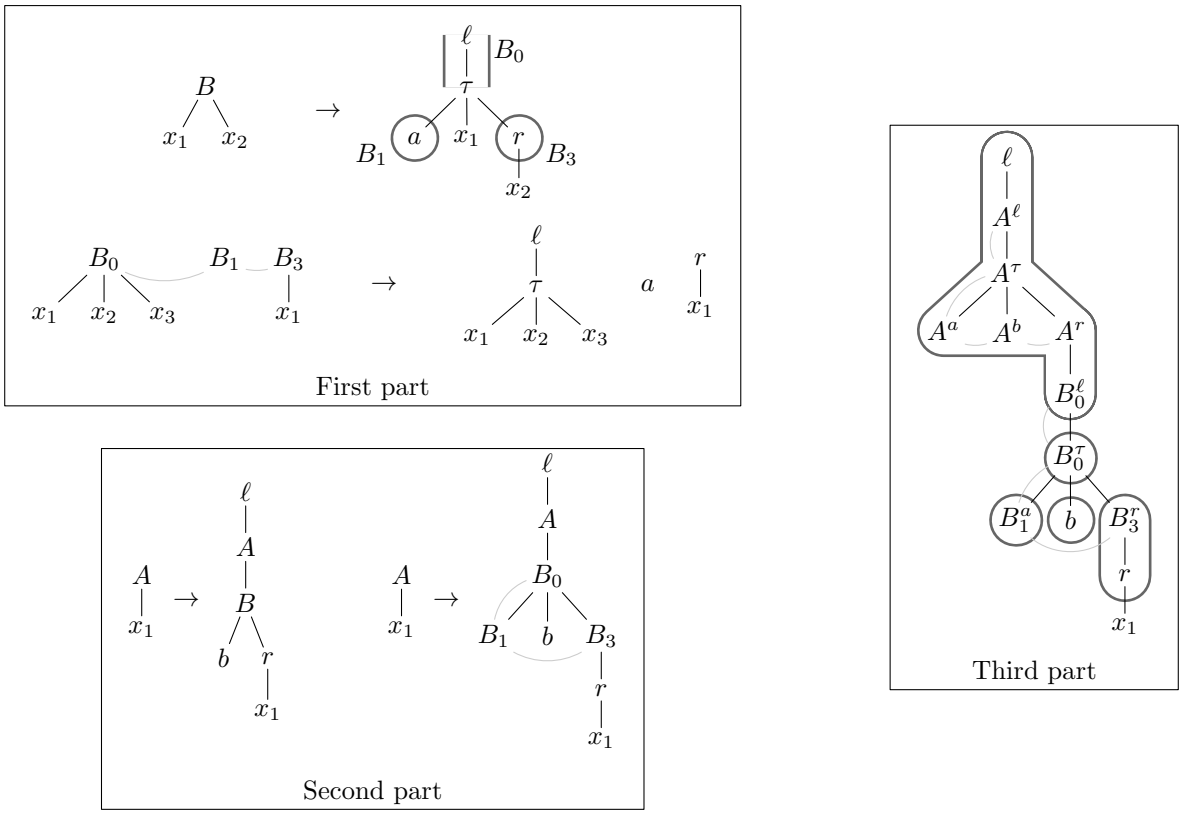


Figure 11: First part: Illustration of the footed decomposition $\langle K_5, \gamma_5 \rangle$ of the right-hand side of rule ρ_5, with the resulting rule $\tilde{\rho}_5$. Second part: Substitution of the skeleton K_5 of B into rule ρ_2. Third part: Adjoining A-decomposition of Example 56.

Let $A \to (u, \mathcal{L})$ be a rule with $A = (A_1, \ldots, A_n)$ and $u = (u_1, \ldots, u_n)$. If the rule is initial (i.e., $A = S$), then the right-hand side u together with the set $\mathcal{L}$ of links is called an *initial tree*, and otherwise it is called an *auxiliary forest*. Application of the rule consists of adjunctions and substitutions. The replacement of the nonterminal A_j by u_j is called an *adjunction* if $\mathrm{rk}(A_j) > 0$ and a *substitution* if $\mathrm{rk}(A_j) = 0$. An occurrence of a nonterminal $C \in N$ in u with $\mathrm{rk}(C) > 0$ has an obligatory adjunction (OA) constraint, whereas an occurrence of a terminal $\sigma \in \Sigma$ in u with $\mathrm{rk}(\sigma) > 0$ has a null adjunction (NA) constraint. In the same manner we handle obligatory and null substitution (OS and NS) constraints. Each big nonterminal $B \in \mathcal{L}$ can be viewed as a selective adjunction/substitution (SA/SS) constraint, which restricts the auxiliary forests that can be adjoined/substituted for B to the right-hand sides of the rules with left-hand side B.

In the literature, MC-TAGs are usually free-choice, which means that the set $\mathcal{L}$ of links can be dropped from the rules (see Section 4.1). By Lemma 20 this is no restriction on footed MCFTGs. An MCFTG is said to be *tree-local* (as opposed to 'set-local') if for every rule as above and every $B \in \mathcal{L}$ there exists $j \in [n]$ such that $\mathrm{occ}(B) \subseteq \mathrm{occ}_N(u_j)$. It can easily be proved that tree-local MCFTGs have the same power as spCFTGs, and similarly that tree-local nsMC-TAGs have the same power as nsTAGs.

The first statement of Theorem 49 shows that nsMC-TAGs have the same tree generating power as MCFTGs. The second statement shows together with Theorem 44 that nsMC-TAGs can be (strongly) lexicalized.

Corollary 52 *For every finitely Δ-ambiguous nsMC-TAG G there is an* $\mathrm{LDT}^{\mathrm{R}}$*-equivalent Δ-lexicalized nsMC-TAG G' such that $\mu(G') \leq (\mu(G) + mrk_\Delta) \cdot mrk_\Sigma \cdot (2 \cdot \theta(G) + 1)$, where Σ is the terminal alphabet of G.*

Remark 53 In Example 50, the finitely Δ-ambiguous spCFTG G_1 is footed and hence an nsTAG. Similarly, the Δ-lexicalized MCFTG G_2' equivalent to G_1 is footed and hence an nsMC-TAG. We now prove that there does not exist a Δ-lexicalized nsTAG equivalent to G_1. In other words, as opposed to nsMC-TAGs, nsTAGs cannot be strongly lexicalized. The proof is a straightforward variant of the one in [65], and we present it here for completeness' sake.

To obtain a contradiction, let $G = (N, \Sigma, S, R)$ be a reduced Δ-lexicalized nsTAG equivalent to G_1. Note that G is a footed spCFTG, and recall from the observations after Definition 48 that every tree in $L(G, A)$ is footed for every nonterminal A. Hence the nonterminals of G have rank 0, 1, or 3. This implies that G is *right-footed*; i.e., for every rule $A(x_1, \ldots, x_k) \to u \in R$ of G with $k \geq 1$, the right-hand side u is of the form $vx_1 \cdots x_k$ with $v \in (N \cup \Sigma)^+$. In fact, if u is not of that form, then it is of the form $v\omega(u_1, u_2, u_3)$ with $v \in (N \cup \Sigma)^*$ and $\omega \in N^{(3)} \cup \{\tau\}$ such that the foot node of u occurs in u_1 or u_2; i.e., either u_1 or u_2 is of the form $v_1\omega'(x_1, \ldots, x_k)v_2$ with $v_1, v_2 \in (N \cup \Sigma)^*$ and $\omega' \in N \cup \Sigma$. But then A generates terminal trees of the form $w\tau(t_1, t_2, t_3)$ with $w \in \Sigma^*$ such that either t_1 or t_2 is of the form $w_1\gamma(x_1, \ldots, x_k)w_2$ with $w_1, w_2 \in \Sigma^*$ and $\gamma \in \{\tau, \ell, r\}$. This contradicts the form of the trees in $L(G_1)$, in which the first and second arguments of τ are always a and b, respectively. Consequently A cannot be reachable, contradicting the fact that G is reduced. Now it is easy to see that every right-footed spCFTG G can be viewed as an ordinary context-free grammar generating $L(G)$ viewed as a string language. We just replace every rule $A(x_1, \ldots, x_k) \to vx_1 \cdots x_k$ by the rule $A \to v$.[32] Thus, it now remains to show that there is no $\{a, b\}$-lexicalized context-free grammar G such that $L(G) = L(G_1)$, where G_1 is the context-free grammar with rules $S \to \ell AAre$, $A \to \ell AAr$, and $A \to \ell\tau abr$. Here '$\{a, b\}$-lexicalized' means that a or b occurs in every right-hand side of a rule of G. For a string $w \in \Sigma^*$, let $c(w) = \#_\ell(w) - \#_r(w)$, where $\#_\ell(w)$ is the number of occurrences of ℓ in w, and similarly for $\#_r(w)$. Since the "parentheses" ℓ and r are balanced in every string in $L(G) = L(G_1)$, it follows from [63, Lemma 4] that for every nonterminal A of G there is a number $c(A) \in \mathbb{N}_0$ such that $c(w) = c(A)$ for every $w \in L(G, A)$. For every $v = v_1 \cdots v_k \in (N \cup \Sigma)^*$ with $v_1, \ldots, v_k \in N \cup \Sigma$, we let $c(v) = \sum_{i=1}^{k} c(v_i)$. Now consider a derivation $S \Rightarrow_G v_1\alpha v_2 \Rightarrow_G^* w_1\alpha w_2$ such that $\alpha \in \{a, b\}$, $v_1, v_2 \in (N \cup \Sigma)^*$, $w_1, w_2 \in \Sigma^*$, and $v_i \Rightarrow_G^* w_i$ for $i \in \{1, 2\}$. Consequently, $w_1\alpha w_2 \in L(G)$. Thus $c(w_1) \in \mathbb{N}_0$, due to the balancing of ℓ and r. By the above, $c(w_1) = c(v_1)$. Since G is $\{a, b\}$-lexicalized and has only finitely many initial rules, this shows that there is a number $\kappa \in \mathbb{N}_0$ with the following property: for every string $w \in L(G)$ there exist $\alpha \in \{a, b\}$ and $w_1, w_2 \in \Sigma^*$ such that $w = w_1\alpha w_2$ and $c(w_1) \leq \kappa$. This is a contradiction because it is easy to see that this does not hold for $w = t_\kappa e \in L(G_1)$, where $t_0 = \ell\tau abr$ and $t_{n+1} = \ell t_n t_n r$ for every $n \in \mathbb{N}_0$.

This shows that nsTAGs cannot be strongly lexicalized. It also shows that context-free grammars cannot be Δ-lexicalized. They can of course be Σ-lexicalized.

The spCFTG G_1 of Example 50 is also monadic; more precisely, it has width $\theta(G_1) = 1$. We finally prove that, as observed in Example 50, there is no Δ-lexicalized monadic spCFTG equivalent to G_1. Let G be such a grammar. By Lemma 40 we may assume that G is nonerasing. It can then be shown as above that G is right-footed. However, in this case we must have $k = 1$ and $\omega = \tau$; moreover, either u_1 or u_2 contains x_1 and hence generates a tree that contains some $\gamma \in \{\tau, \ell, r\}$ because G is nonerasing. The remainder of the proof is the same as above. This shows that to lexicalize an MCFTG G, either the width $\theta(G)$ or the multiplicity $\mu(G)$ must increase. □

We now define strict MC-TAGs as follows. A *(strict set-local) multi-component tree adjoining grammar* (in short, MC-TAG) is a footed MCFTG $G = (N, \mathcal{N}, \Sigma, S, R)$ for which there exists an equivalence relation $\equiv$ on $N \cup \Sigma$ such that

(1) for all $\sigma, \tau \in \Sigma$, if $\sigma \neq \tau$ and $\sigma \equiv \tau$, then $\mathrm{rk}(\sigma) \neq \mathrm{rk}(\tau)$;
(2) for every $C \in N$ there exists $\sigma \in \Sigma$ such that $C \equiv \sigma$; and
(3) for every rule $(A_1, \ldots, A_n) \to ((u_1, \ldots, u_n), \mathcal{L})$ in R and every $j \in [n]$,
   (a) $u_j(\varepsilon) \equiv A_j$ and
   (b) if $\mathrm{rk}(A_j) \geq 1$, then $u_j(p) \equiv A_j$, where p is the foot node of u_j.

The first requirement means that distinct equivalent terminal symbols can be viewed as the same "final" symbol with different ranks. In this way, Σ can be viewed as corresponding to a "final" alphabet, in which each symbol can have a finite number of different ranks, as for example in derivation trees of context-free grammars. The second requirement means that each nonterminal C that is equivalent to terminal σ can be viewed as the same final symbol as σ together with some information that is relevant to SA constraints. The third requirement means that the root and foot node of u_j are equivalent to A_j; i.e., represent the same final symbol as A_j. Thus, intuitively, adjunction always replaces a final symbol by a tree with that same final symbol as root label and foot node label. We define a *tree adjoining grammar* (in short, TAG) to be an MC-TAG of multiplicity 1; i.e., a footed spCFTG that satisfies the requirements above.

[32] This generalizes the fact that every regular tree grammar is a context-free grammar (see Section 2.2).

Example 54 A simple example of a TAG $\overline{G}_1$ is obtained from the spCFTG G_1 in Example 50 by adding a terminal symbol γ of rank 1. The rules of $\overline{G}_1$ are

$$S \to \gamma\ell A(A'(re)) \qquad A(x_1) \to \gamma\ell A(A'(r\gamma x_1)) \qquad \text{and} \qquad A(x_1) \to \gamma\ell\tau(a, b, r\gamma x_1) \ ,$$

where A' is an alias of A. The equivalence relation $\equiv$ is the smallest one such that $S \equiv A \equiv A' \equiv \gamma$. It clearly satisfies the above three requirements. This TAG is closely related to the one in [65]. It can be proved in exactly the same way as in Remark 53 that there is no $\{a, b\}$-lexicalized nsTAG equivalent to $\overline{G}_1$, which slightly generalizes the result of [65].[33] Thus, TAGs cannot be strongly lexicalized by nsTAGs.

The MCFTG G' of Example 51 is an MC-TAG. The equivalence relation $\equiv$ is the smallest one such that $S' \equiv A_0 \equiv \sigma \equiv \tau$, $A_1 \equiv a$, and $A_3 \equiv b$. Note that $\mathrm{rk}(\sigma) \neq \mathrm{rk}(\tau)$. □

Let MC-TAL denote the class of tree languages generated by MC-TAGs. In the next subsection we prove that MCFT and MC-TAL are almost the same class of tree languages.

6.2. MC-TAL almost equals MCFT

By definition, we have MC-TAL $\subseteq$ MCFT. In the other direction, the inclusion MCFT $\subseteq$ MC-TAL does not hold because a tree language from MC-TAL cannot contain two trees of which the roots are labeled with two different symbols of the same rank. In this subsection we show that this is indeed the only necessary restriction. To prove that every language $L \in$ MCFT satisfying this restriction is in MC-TAL, we begin with the case where the root of each tree $t \in L$ is labeled by the same symbol σ_0. In this case we will construct an MC-TAG of a special type, which we define next. We first need some more terminology.

Let $G = (N, \mathcal{N}, \Sigma, S, R)$ be an MCFTG. Recall from Definition 48 that a pattern $t \in P_{N\cup\Sigma}(X_k)$ with $k \in \mathbb{N}_0$ is footed if either $k = 0$, or $k \geq 1$ and there is a position $p \in \mathrm{pos}_{N\cup\Sigma}(t)$, called the foot node of t, with $\mathrm{rk}(t(p)) = k$ and $t(pi) = x_i$ for every $i \in [k]$. Given a footed pattern $t \in P_{N\cup\Sigma}(X_k)$ with $k \geq 1$, we define $\mathrm{rlab}(t) = t(\varepsilon)$ and $\mathrm{flab}(t) = t(p)$ where p is the (unique) foot node of t. Thus, $\mathrm{rlab}(t)$ and $\mathrm{flab}(t)$ are the labels of the root and the foot node of t, respectively. In the case where $k = 0$ we define $\mathrm{rlab}(t) = t(\varepsilon)$ and, for technical convenience, also $\mathrm{flab}(t) = t(\varepsilon)$. Thus, in this case $\mathrm{rlab}(t)$ is also the label of the root of t and $\mathrm{flab}(t) = \mathrm{rlab}(t)$. For $k \geq 1$ we define the *spine* of t to be the set of all ancestors of its foot node (including the foot node itself), whereas for $k = 0$ the spine of t is defined to be the empty set.

An *adjoining* MCFTG is a footed MCFTG G for which there is a mapping $\varphi\colon N \cup \Sigma \to \Sigma$ such that

(1) $\varphi(\sigma) = \sigma$ for every $\sigma \in \Sigma$, and
(2) $\varphi(\mathrm{rlab}(u_j)) = \varphi(\mathrm{flab}(u_j)) = \varphi(A_j)$ for every rule $(A_1, \ldots, A_n) \to ((u_1, \ldots, u_n), \mathcal{L}) \in R$ and every $j \in [n]$.

This implies that φ is rank-preserving for nonterminals of rank at least 1 (assuming that such a nonterminal generates at least one terminal tree). Obviously, every adjoining MCFTG is an MC-TAG with respect to the equivalence relation $\equiv$ that is the kernel of φ; i.e., $\alpha \equiv \beta$ if $\varphi(\alpha) = \varphi(\beta)$. By (1) above, $\equiv$ is the identity on Σ. Vice versa, if G is an MC-TAG with respect to an equivalence relation that is the identity on Σ, then G is an adjoining MCFTG (as can easily be checked).

We now prove that for every footed MCFTG G that generates a tree language in which all trees have the same root label σ_0, there is an equivalent adjoining MCFTG, which is also lexicalized if G is lexicalized. In fact, the next lemma proves a slightly more general fact, which will be needed to prove the theorem following the lemma. The proof of the lemma is very similar to the one of Theorem 49, with a further decomposition of the trees in the right-hand sides of the rules.

Lemma 55 *Let $G = (N, \mathcal{N}, \Sigma, S, R)$ be a footed MCFTG and let $\sigma_0 \in \Sigma$. Then there is an adjoining MCFTG G^{σ_0} such that $L(G^{\sigma_0}) = \{t \in L(G) \mid t(\varepsilon) = \sigma_0\}$ and $\mu(G^{\sigma_0}) = \mu(G) \cdot |\Sigma| \cdot mrk_\Sigma$. Moreover, if G is Δ-lexicalized, then so is G^{σ_0}.*

[33] The language class TAL generated by TAGs is properly included in the language class nsTAL, which is generated by nsTAGs. The tree language $L = \{\ell^n r^n e \mid n \in \mathbb{N}\}$, which is root consistent (cf. Corollary 59 in the next subsection), is a witness for the properness. It is generated by an nsTAG with rules $S \to A(e)$, $A(x_1) \to \ell A(rx_1)$, and $A(x_1) \to \ell r x_1$. For the sake of a contradiction, let $G = (N, \Sigma, S, R)$ be a TAG such that $L(G) = L$. Clearly, $\theta(G) \leq 1$ and G must be right-footed (cf. Remark 53). For any unary nonterminal $A \in N^{(1)}$ we have $L(G, A) \subseteq \{\ell^k x_1 \mid k \in \mathbb{N}\}$ or $L(G, A) \subseteq \{r^k x_1 \mid k \in \mathbb{N}\}$ due to the condition that the root label and foot node label must coincide. However, since G can be viewed as an ordinary context-free grammar generating the string language L, these languages $L(G, A)$ must be finite, due to pumping. Hence we can transform G into an equivalent right-linear context-free grammar, which is a contradiction because L is not regular.

Proof The basic idea of this proof is that, for any alphabet Ω, every string $w \in \Omega^+$ can be decomposed as $w = w_1 \cdots w_n$ such that $1 \leq n \leq |\Omega|$, $w_i \in \Omega^+$, and the first and last symbol of w_i are the same. We quickly prove this by induction on $|\Omega|$. Let a be the first symbol of w, and let w_1 be the longest prefix of w that ends on a. Then $w = w_1 w'$ with $w' \in (\Omega \setminus \{a\})^*$. If $w' = \varepsilon$, then we are ready. Otherwise we apply the induction hypothesis. This decomposition is of course not unique. For example, the proof gives $abab = aba \cdot b$, but another decomposition is $abab = a \cdot bab$.

Let $G = (N, \mathcal{N}, \Sigma, S, R)$ be a footed MCFTG, and let $\sigma_0 \in \Sigma$. We define $G^{\sigma_0} = (N', \mathcal{N}', \Sigma, S^{\sigma_0}, R')$, where N', $\mathcal{N}'$, and R' do not depend on σ_0. The set N' of nonterminals consists of all 4-tuples $\langle C, \sigma, m, p\rangle$ with $C \in N$, $\sigma \in \Sigma$, $m \in \{0, \mathrm{rk}(\sigma)\}$, and $p \in \mathbb{N}^*$ such that $|p| < |\Sigma|$. The rank of $\langle C, \sigma, m, p\rangle$ is m. The initial nonterminal is $S^{\sigma_0} = \langle S, \sigma_0, 0, \varepsilon\rangle$. Let $\varphi\colon N' \cup \Sigma \to \Sigma$ be defined by $\varphi(\langle C, \sigma, m, p\rangle) = \varphi(\sigma) = \sigma$. We will define $\mathcal{N}'$ and R' in such a way that G^{σ_0} is an adjoining MCFTG with respect to φ.

For every nonterminal $C \in N$, a *skeleton* of C is a footed pattern $K \in P_{N'}(X_{\mathrm{rk}(C)})$ such that

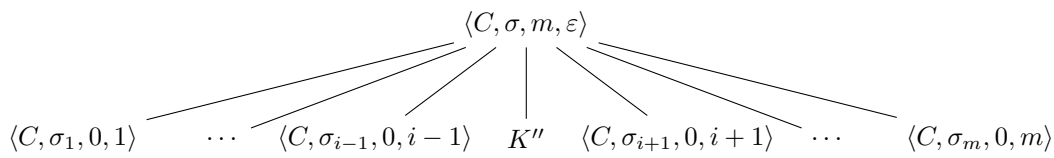
(1) for every $p \in \mathrm{pos}_{N'}(K)$ there exist $\sigma \in \Sigma$ and $m \in \{0, \mathrm{rk}(\sigma)\}$ such that $K(p) = \langle C, \sigma, m, p\rangle$,
(2) every subtree of K in $T_{N'}$ is in $(N')^{(0)}$, and
(3) $\varphi(K(p)) \neq \varphi(K(p'))$ for every two distinct positions $p, p' \in \mathrm{pos}_{N'}(K)$ on the spine of K.

For such a skeleton K, we define $\mathrm{seq}(K) = \mathrm{yd}_{N'}(K)$, which is an element of $(N')^+$. We note that there are only finitely many skeletons of C. In fact, $|\mathrm{pos}_{N'}(K)| \leq |\Sigma| \cdot mrk_\Sigma$ for every skeleton K of C, if $\mathrm{rk}(C) \geq 1$. Additionally, if $\mathrm{rk}(C) = 0$, then every skeleton of C is of the form $\langle C, \sigma, 0, \varepsilon\rangle$ with $\sigma \in \Sigma$. We finally note that K can be reconstructed from $\mathrm{seq}(K)$ because K is footed.

We will apply the above basic idea to the sequence of φ-images of the labels of the nodes on the spine of a footed pattern u; i.e., to the sequence $(\varphi(u(p_1)), \ldots, \varphi(u(p_n)))$ where $p_1, \ldots, p_n$ are the positions on the spine of u, in the order of increasing length. This leads to a decomposition of u that can be represented by a skeleton K and a substitution function γ such that $u = K[\gamma]$. Formally, let $K \in P_{N'}(X)$ be a skeleton of $C \in N$. A substitution function γ for $\mathrm{occ}_{N'}(K)$ is *adjoining* if, for every $C' \in \mathrm{occ}_{N'}(K)$, the pattern $\gamma(C') \in P_{N' \cup \Sigma}(X)$ is footed and $\varphi(\mathrm{rlab}(\gamma(C'))) = \varphi(\mathrm{flab}(\gamma(C'))) = \varphi(C')$. We say that the pair $\langle K, \gamma\rangle$ is an *adjoining C-decomposition* of the tree $K[\gamma]$.

Basic fact. Every footed pattern u has an adjoining C-decomposition $\mathrm{dec}_C(u)$. More precisely, for every $C \in N$ and every footed pattern $u \in P_{N' \cup \Sigma}(X_{\mathrm{rk}(C)})$ there is a pair $\mathrm{dec}_C(u) = \langle K, \gamma\rangle$ such that K is a skeleton of C, γ is an adjoining substitution function for $\mathrm{occ}_{N'}(K)$, and $K[\gamma] = u$.

Proof of the basic fact. Let $\sigma = \varphi(\mathrm{rlab}(u))$. First suppose that $\mathrm{rk}(u) = 0$. Then $\mathrm{dec}_C(u) = \langle K, \gamma\rangle$ with $K = \langle C, \sigma, 0, \varepsilon\rangle$ and $\gamma(\langle C, \sigma, 0, \varepsilon\rangle) = u$. Now suppose that $\mathrm{rk}(u) \geq 1$. We use induction on the cardinality of the spine of u. Let q be the longest position on the spine of u such that $\varphi(u(q)) = \sigma$, and let $\mathrm{rk}(\sigma) = m$. If q is the foot node of u, then $\mathrm{dec}_C(u) = \langle K, \gamma\rangle$ with $K = \mathrm{in}(\langle C, \sigma, m, \varepsilon\rangle)$ and $\gamma(\langle C, \sigma, m, \varepsilon\rangle) = u$. Otherwise, let $i \in \mathbb{N}$ be the unique integer such that qi is a position on the spine of u. Let $u' = u|_{qi}$, and let $\mathrm{dec}_C(u') = \langle K', \gamma'\rangle$ by the induction hypothesis. Then $\mathrm{dec}_C(u) = \langle K, \gamma\rangle$, where K and γ are defined as follows. Let K'' be obtained from K' by changing every label $\langle C, \sigma', m', p\rangle$ into $\langle C, \sigma', m', ip\rangle$. Then K is the tree



where $\sigma_j = \varphi(u(qj))$ for every $j \in [m] \setminus \{i\}$. Moreover, the substitution function γ is defined by:

- $\gamma(\langle C, \sigma, m, \varepsilon\rangle) = (u|^q)[\square \leftarrow \mathrm{in}(\sigma)]$,
- $\gamma(\langle C, \sigma_j, 0, j\rangle) = u|_{qj}$ for every $j \in [m] \setminus \{i\}$, and
- $\gamma(\langle C, \sigma', m', ip\rangle) = \gamma'(\langle C, \sigma', m', p\rangle)$ for every $\langle C, \sigma', m', ip\rangle \in \mathrm{occ}_{N'}(K'')$.

It is straightforward to verify that K and γ satisfy the requirements, which completes the proof of the basic fact.

The definition of the set $\mathcal{N}'$ of big nonterminals and the set R' of rules is exactly the same as in the proof of Theorem 49.[34] It should be clear that G^{σ_0} is adjoining with respect to φ. The correctness

[34] Except that in the construction of the rule $\langle \rho, \overline{\delta}\rangle$ it should be clear that $u_j[f]$ is a footed pattern in $P_{N' \cup \Sigma}(X_{\mathrm{rk}(C)})$. Moreover, the decomposition $\mathrm{dec}_{A_j}(u_j[f])$ is of course an adjoining A_j-decomposition of $u_j[f]$.

of G^{σ_0} is also proved in the same way as in the proof of Theorem 49. The Claim and its proof are exactly the same. In the case where $A = S$ we obtain in the claim that $\kappa_d(S) = \langle S, \sigma, 0, \varepsilon\rangle$ with $\sigma \in \Sigma$, and hence $\mathrm{val}(q(d)) = \mathrm{val}(d)$. Since G^{σ_0} is adjoining, it is easy to see that $\sigma = \mathrm{val}(q(d))(\varepsilon)$; i.e., the root symbol of $\mathrm{val}(d)$. Hence $\{t \in L(G) \mid t(\varepsilon) = \sigma_0\} \subseteq L(G^{\sigma_0})$. As in the proof of Theorem 49 there is an $\mathrm{LDT}^{\mathrm{R}}$-transducer M that computes $q(d)$ from d, and every derivation tree $d' \in L(G^{\sigma_0}_{\mathrm{der}}, \langle S, \sigma, 0, \varepsilon\rangle)$ can be turned into a derivation tree $d = M'(d') \in L(G_{\mathrm{der}})$ such that $q(d) = d'$ by changing every label $\langle \rho, \overline{\kappa}\rangle$ into ρ. Taking $\sigma = \sigma_0$ this shows that $L(G^{\sigma_0}) \subseteq \{t \in L(G) \mid t(\varepsilon) = \sigma_0\}$, and hence the correctness of G^{σ_0}. ∎

Example 56 Let us consider the MCFTG G'_2 of Example 50. As already observed in Remark 53, G'_2 is footed and hence an nsMC-TAG. Here we illustrate the proof of Lemma 55 by constructing the adjoining MCFTG G^ℓ for $G = G'_2$; note that G^ℓ is equivalent to G'_2 because $t(\varepsilon) = \ell$ for every $t \in L(G'_2)$. We recall that G'_2 has the following rules (where we replace $\tilde{\rho}_i$ by ρ_i, for convenience):

$$\begin{aligned}
\rho_1\colon\quad & S \to \ell A(B_0(B_1, b, B_3(re))) \\
\rho_2\colon\quad & A(x_1) \to \ell A(B_0(B_1, b, B_3(rx_1))) \\
\rho_3\colon\quad & A(x_1) \to \ell\tau(a, b, rx_1) \\
\rho_4\colon\quad & (B_0(x_1, x_2, x_3), B_1, B_3(x_1)) \to (\ell B_0(x_1, x_2, x_3),\ B_1,\ B_3(B'_0(B'_1, b, B'_3(rx_1)))) \\
\rho_5\colon\quad & (B_0(x_1), B_1, B_3(x_1)) \to (\ell\tau(x_1, x_2, x_3), a, rx_1)\ ,
\end{aligned}$$

plus the rules ρ'_4 and ρ'_5 for the alias (B'_0, B'_1, B'_3) of (B_0, B_1, B_3). For rule ρ_5 and $\overline{\kappa} = \varepsilon$, we obtain the skeleton function $\kappa_5 = \kappa_{\rho_5,\varepsilon}$ for (B_0, B_1, B_3) such that

$$\kappa_5(B_0) = B_0^\ell(B_0^\tau(x_1, x_2, x_3)) \qquad \kappa_5(B_1) = B_1^a \qquad \text{and} \qquad \kappa_5(B_3) = B_3^r(x_1)\ ,$$

where $B_0^\ell = \langle B_0, \ell, 1, \varepsilon\rangle$, $B_0^\tau = \langle B_0, \tau, 1, 1\rangle$, $B_1^a = \langle B_1, a, 0, \varepsilon\rangle$, and $B_3^r = \langle B_3, r, 1, \varepsilon\rangle$. The resulting rule $\tilde{\rho}_5 = \langle \rho_5, \varepsilon\rangle$ is

$$\tilde{\rho}_5\colon\quad (B_0^\ell(x_1), B_0^\tau(x_1, x_2, x_3), B_1^a, B_3^r(x_1)) \to (\ell x_1,\ \tau(x_1, x_2, x_3),\ a,\ rx_1)\ .$$

Substituting $\kappa_5(B_i)$ for B_i (and $\kappa'_5(B'_i)$ for B'_i) in the right-hand side u_4 of rule ρ_4, we obtain the forest $u'_4 = (\ell B_0^\ell(B_0^\tau(x_1, x_2, x_3)),\ B_1^a,\ B_3^r(B_0'^\ell(B_0'^\tau(B_1'^a, b, B_3'^r(rx_1)))))$ and from that the following rule $\tilde{\rho}_4 = \langle \rho_4, (\kappa_5, \kappa'_5)\rangle$:

$$\begin{aligned}
\tilde{\rho}_4\colon\quad & (B_0^\ell(x_1), B_0^\tau(x_1, x_2, x_3), B_1^a, B_3^r(x_1)) \\
& \to (\ell B_0^\ell(x_1),\ B_0^\tau(x_1, x_2, x_3),\ B_1^a,\ B_3^r(B_0'^\ell(B_0'^\tau(B_1'^a, b, B_3'^r(rx_1))))))\ ,
\end{aligned}$$

and the skeleton function $\kappa_{\rho_4,(\kappa_5,\kappa'_5)} = \kappa_5$ for (B_0, B_1, B_3). Thus, these are all the new rules obtained from ρ_4 and ρ_5. We now turn to rules ρ_3 and ρ_2. The only skeleton needed for A is the tree

$$K = \kappa_{\rho_3,\varepsilon}(A) = A^\ell(A^\tau(A^a, A^b, A^r(x_1)))\ ,$$

where $A^\ell = \langle A, \ell, 1, \varepsilon\rangle$, $A^\tau = \langle A, \tau, 1, 1\rangle$, $A^a = \langle A, a, 0, 11\rangle$, $A^b = \langle A, b, 0, 12\rangle$, and $A^r = \langle A, r, 1, 13\rangle$. The resulting rule $\tilde{\rho}_3 = \langle \rho_3, \varepsilon\rangle$ is

$$\tilde{\rho}_3\colon\quad (A^\ell(x_1), A^\tau(x_1, x_2, x_3), A^a, A^b, A^r(x_1)) \to (\ell x_1,\ \tau(x_1, x_2, x_3),\ a,\ b,\ rx_1)\ .$$

Substituting K for A and $\kappa_5(B_i)$ for B_i in the right-hand side $u_2 = \ell A(B_0(B_1, b, B_3(rx_1)))$ of ρ_2, we obtain the tree

$$u'_2 = \ell A^\ell(A^\tau(A^a, A^b, A^r(B_0^\ell(B_0^\tau(B_1^a, b, B_3^r(rx_1))))))\ .$$

It has the adjoining A-decomposition $\mathrm{dec}_A(u'_2) = \langle K, \gamma\rangle$ such that $\gamma(A^\ell) = \ell A^\ell(A^\tau(A^a, A^b, A^r(B_0^\ell(x_1))))$, $\gamma(A^\tau) = B_0^\tau(x_1, x_2, x_3)$, $\gamma(A^a) = B_1^a$, $\gamma(A^b) = b$, and $\gamma(A^r) = B_3^r(rx_1)$, which is illustrated in the third part of Figure 11. The resulting rule $\tilde{\rho}_2 = \langle \rho_2, (\kappa_{\rho_3,\varepsilon}, \kappa_5)\rangle$ is

$$\begin{aligned}
\tilde{\rho}_2\colon\quad & (A^\ell(x_1), A^\tau(x_1, x_2, x_3), A^a, A^b, A^r(x_1)) \\
& \to (\ell A^\ell(A^\tau(A^a, A^b, A^r(B_0^\ell(x_1)))),\ B_0^\tau(x_1, x_2, x_3),\ B_1^a,\ b,\ B_3^r(rx_1))\ .
\end{aligned}$$

Finally, we consider rule ρ_1. The only skeleton needed for S is $S^\ell = \langle S, \ell, 0, \varepsilon\rangle$, which is the initial nonterminal of G^ℓ. Substituting K for A and $\kappa_5(B_i)$ for B_i in the right-hand side $\ell A(B_0(B_1, b, B_3(re)))$ of ρ_1, we obtain the tree $u_2'[x_1 \leftarrow e]$ and the new rule

$$\tilde{\rho}_1\colon \quad S^\ell \to \ell A^\ell(A^\tau(A^a, A^b, A^r(B_0^\ell(B_0^\tau(B_1^a, b, B_3^r(re)))))) \ ,$$

where $\tilde{\rho}_1 = \langle \rho_1, (\kappa_{\rho_3,\varepsilon}, \kappa_5)\rangle$. Thus, G^ℓ has the rules $\{\tilde{\rho}_1, \tilde{\rho}_2, \tilde{\rho}_3, \tilde{\rho}_4, \tilde{\rho}_5, \tilde{\rho}_4', \tilde{\rho}_5'\}$. Clearly, the tree K generates the same terminal trees as $A(x_1)$ and the tree $\kappa_5(B_i)$ generates the same terminal trees as $\mathrm{in}(B_i)$ for every $i \in [3]$. It is easy to check that G^ℓ is an $\{a, b\}$-lexicalized MC-TAG with respect to the smallest equivalence $\equiv$ such that $C^x \equiv x$ for every $C \in \{S, A, B_0, B_0', B_1, B_1', B_3, B_3'\}$ and every $x \in \{\ell, \tau, a, b, r\}$.

We finally mention that, in Example 50, the first rule of the grammar G_2 could be changed into the rule $S \to \ell B(b, B'(b, re))$, because $B(b, x_1)$ generates the same terminal trees as $A(x_1)$. This makes the nonterminal A superfluous. We have not done this, for the sake of illustration of our constructions. As a result of this change, the three rules $\tilde{\rho}_1, \tilde{\rho}_2, \tilde{\rho}_3$ of G^ℓ can be changed into the one rule $S^\ell \to \ell B_0^\ell(B_0^\tau(B_1^a, b, B_3^r(B_0'^\ell(B_0'^\tau(B_1'^a, b, B_3'^r(re)))))))$. □

Example 57 As another, similar example, let us consider the $\{a, b\}$-lexicalized MCFTG G obtained from G_2' by changing in its rules every ℓ into $\gamma\ell$ and every r (except the one in ρ_1) into $r\gamma$, where γ has rank 1. Thus, G has the rules

$$\begin{array}{llll}
\rho_1\colon & S \to \gamma\ell A(B_0(B_1, b, B_3(re))) & \rho_3\colon & A(x_1) \to \gamma\ell\tau(a, b, r\gamma x_1) \\
\rho_2\colon & A(x_1) \to \gamma\ell A(B_0(B_1, b, B_3(r\gamma x_1))) & \rho_5\colon & B \to \big(\gamma\ell\tau(x_1, x_2, x_3), a, r\gamma x_1\big) \\
\rho_4\colon & B \to \big(\gamma\ell B_0(x_1, x_2, x_3),\ B_1,\ B_3(B_0'(B_1', b, B_3'(r\gamma x_1)))\big) \ , & &
\end{array}$$

where $B = (B_0(x_1, x_2, x_3), B_1, B_3(x_1))$. Clearly, G is equivalent to the TAG $\overline{G}_1$ of Example 54, for which there is no equivalent $\{a, b\}$-lexicalized nsTAG.

Since ρ_2 and ρ_3 are MC-TAG rules with respect to $A \equiv \gamma$, they do not have to be changed. It is not difficult to see that the only skeleton function needed for (B_0, B_1, B_3) is κ_5 with $\kappa_5(B_0) = B_0^\gamma(B_0^\ell(B_0^\tau(x_1)))$, $\kappa_5(B_1) = B_1^a$, and $\kappa_5(B_3) = B_3^r(B_3^\gamma(x_1))$, where $B_0^\gamma = \langle B_0, \gamma, 1, \varepsilon\rangle$, $B_0^\ell = \langle B_0, \ell, 1, 1\rangle$, $B_0^\tau = \langle B_0, \tau, 1, 11\rangle$, and similarly for B_3, and $B_1^a = \langle B_1, a, 0, \varepsilon\rangle$. Given these skeletons, it is straightforward to construct the following rules for G^γ:

$$\begin{array}{llll}
\tilde{\rho}_1\colon & S^\gamma \to \gamma\ell A(B_0^\gamma(B_0^\ell(B_0^\tau(B_1^a, b, B_3^r(B_3^\gamma(re)))))) & \tilde{\rho}_3\colon & A(x_1) \to \gamma\ell\tau(a, b, r\gamma x_1) \\
\tilde{\rho}_2\colon & A(x_1) \to \gamma\ell A(B_0^\gamma(B_0^\ell(B_0^\tau(B_1^a, b, B_3^r(B_3^\gamma(r\gamma x_1)))))) & \tilde{\rho}_5\colon & \bar{B} \to \big(\gamma x_1, \ell x_1, \tau(x_1, x_2, x_3), a, r x_1, \gamma x_1\big) \\
\tilde{\rho}_4\colon & \bar{B} \to \big(\gamma\ell B_0^\gamma(x_1),\ B_0^\ell(x_1),\ B_0^\tau(x_1, x_2, x_3),\ B_1^a,\ B_3^r(B_3^\gamma(B_0'^\gamma(B_0'^\ell(B_0'^\tau(B_1'^a, b, B_3'^r(B_3'^\gamma(r x_1))))))),\ \gamma x_1\big) & &
\end{array}$$

where $\bar{B} = (B_0^\gamma(x_1), B_0^\ell(x_1), B_0^\tau(x_1, x_2, x_3), B_1^a, B_3^r(x_1), B_3^\gamma(x_1))$. Clearly, G^γ is an $\{a, b\}$-lexicalized MC-TAG equivalent to the TAG $\overline{G}_1$. □

Let us say that a tree language L is *root consistent* if $\mathrm{rk}(t_1(\varepsilon)) \neq \mathrm{rk}(t_2(\varepsilon))$ for all $t_1, t_2 \in L$ such that $t_1(\varepsilon) \neq t_2(\varepsilon)$. It should be clear that every tree language in MC-TAL is root consistent.

Theorem 58 *For every MCFTG G such that $L(G)$ is root consistent, there is an* $\mathrm{LDT}^{\mathrm{R}}$*-equivalent MC-TAG G' such that*

$$\mu(G') \leq \begin{cases} \mu(G) & \text{if } \theta(G) = 0 \\ \mu(G) \cdot |\Sigma| \cdot mrk_\Sigma^2 \cdot (2 \cdot \theta(G) - 1) & \text{if } \theta(G) \geq 1 \ , \end{cases}$$

where Σ is the terminal alphabet of G. Moreover, if G is Δ-lexicalized, then so is G'.

Proof With the help of Theorem 49, we may assume that $G = (N, \mathcal{N}, \Sigma, S, R)$ is a footed MCFTG. The set $\Omega = \{t(\varepsilon) \mid t \in L(G)\}$ can be computed by deciding the emptiness of $L(G^\sigma)$ for every $\sigma \in \Sigma$, where G^σ is the MCFTG of Lemma 55. Now let σ_0 be an arbitrary element of Ω, and construct the adjoining MCFTG $G^{\sigma_0} = (N', \mathcal{N}', \Sigma, S^{\sigma_0}, R')$ as in the proof of Lemma 55. From G^{σ_0} we construct G' by identifying all nonterminals $\langle S, \sigma, 0, \varepsilon\rangle$ such that $\sigma \in \Omega$ and taking the resulting nonterminal S' to be the initial nonterminal of G'. Since G^{σ_0} is adjoining, it is straightforward to check that G' is an MC-TAG with respect to the smallest equivalence $\equiv$ such that $\sigma_1 \equiv \sigma_2 \equiv S'$ for all $\sigma_1, \sigma_2 \in \Omega$ and $\langle C, \sigma, b, p\rangle \equiv \sigma$ for all $\langle C, \sigma, b, p\rangle \in N'$. It is easy to modify the $\mathrm{LDT}^{\mathrm{R}}$-transducers M and M' in the proof of Lemma 55 such that they show the $\mathrm{LDT}^{\mathrm{R}}$-equivalence of G and G'. We finally note that if $\theta(G) = 0$, then $\mu(G^{\sigma_0}) = \mu(G)$ by the proof of Lemma 55. ■

We now can characterize MCFT and MC-TAL in terms of each other in a very simple way.

Corollary 59 *Let # be a new symbol of rank* 1*. Then*

$$\text{MC-TAL} = \{L \in \text{MCFT} \mid L \textit{ is root consistent}\} \qquad \textit{and} \qquad \text{MCFT} = \{L \mid \#(L) \in \text{MC-TAL}\} \ .$$

Proof The first equality is immediate from Theorem 58 and the fact that every tree language in MC-TAL is root consistent. It is easy to see that if $L \in \text{MCFT}$, then $\#(L) \in \text{MCFT}$. This also holds in the other direction because MCFT is closed under tree homomorphisms by Lemma 21. The second equality now follows from Theorem 58 because $\#(L)$ is root consistent. ∎

As observed in the Introduction this corollary settles a problem stated in [93, Section 4.5], which can be reformulated as "it would be interesting to investigate whether MC-TAL is properly included in MCFT". By the first statement of Corollary 59 that is indeed the case; i.e., MCFTGs are slightly more powerful than MC-TAGs. However, by the second statement they have the same power provided that MC-TAGs are allowed to make use of a root-marker. Another obvious way to "force" equality of MCFT and MC-TAL is to allow MCFTGs, and hence MC-TAGs, to use several initial nonterminals instead of just one. It is clear that this does not change the class MCFT. Thus, the proper inclusion of MC-TAL in MCFT is due to minor technicalities. For that reason we feel justified to state that MCFTGs and MC-TAGs have the same tree generating power.

As another corollary we obtain from Theorems 58 and 44 that MC-TAGs can be (strongly) lexicalized. Thus, although TAGs cannot be strongly lexicalized, as proved in [65] (cf. Remark 53), MC-TAGs can. This was illustrated in Example 57.

Theorem 60 *For every finitely Δ-ambiguous MC-TAG G there is an* LDT^{R}*-equivalent Δ-lexicalized MC-TAG G' such that $\mu(G') \leq (\mu(G) + mrk_\Delta) \cdot |\Sigma| \cdot mrk_\Sigma^2 \cdot (2 \cdot \theta(G) + 1)$, where Σ is the terminal alphabet of G.*

6.3. Monadic MCFTGs

We say that an MCFTG G is *monadic* if $\theta(G) \leq 1$. For instance, the grammars of Examples 6, 7, and 50 are monadic. As observed in the beginning of this section, it is shown in [61] that nsTAGs have the same tree generating power as monadic spCFTGs. Similarly, on the basis of Theorem 58, we can now prove that MCFTGs have the same tree generating power as monadic MCFTGs. The construction in the proof is the same as in [40].

Theorem 61 *For every MCFTG G with $\theta(G) \geq 2$ there is an* LDT^{R}*-equivalent monadic MCFTG G' such that $\mu(G') \leq \mu(G) \cdot |\Sigma| \cdot mrk_\Sigma^2 \cdot (2 \cdot \theta(G) - 1)$, where Σ is the terminal alphabet of G. Moreover, if $\Delta \subseteq \Sigma^{(0)}$ and G is Δ-lexicalized, then G' is Δ-lexicalized.*

Proof It should be clear from Lemma 55 and the proof of Corollary 59 that we may assume that $G = (N, \mathcal{N}, \Sigma, S, R)$ is an adjoining MCFTG with respect to a mapping $\varphi\colon N \cup \Sigma \to \Sigma$, as defined in Section 6.2.[35] We define the monadic $G' = (N, \mathcal{N}, \Sigma, S, R')$ such that every nonterminal $C \in N$ with $\text{rk}(C) \geq 2$ in G now has rank $\text{rk}'(C) = 1$ in G', and $\text{rk}'(C) = \text{rk}(C)$ for the nonterminals with $\text{rk}(C) \leq 1$. The idea of the proof is that every occurrence of a nonterminal $C(x_1, \ldots, x_m)$ of rank $m \geq 1$ is replaced by $C(\sigma(x_1, \ldots, x_m))$ where $\sigma = \varphi(C)$, such that in G' the nonterminal C does not generate the foot node of the tree generated by C in G. Thus, for a footed pattern $t \in P_{N \cup \Sigma}(X)$ of rank at least 1, let $\text{cut}(t)$ denote the unique pattern of rank 1 such that $t = \text{cut}(t)[x_1 \leftarrow \text{in}(\text{flab}(t))]$. For instance, $\text{cut}(\sigma(a, \tau(x_1, x_2))) = \sigma(a, x_1)$. Moreover, for simplicity, let $\text{cut}(t) = t$ for every tree $t \in T_{N \cup \Sigma}$. Now let $\rho = A \to ((u_1, \ldots, u_n), \mathcal{L})$ be a rule in R with $A = (A_1, \ldots, A_n)$, and let f be the substitution function for N such that $f(C) = C(\text{in}(\varphi(C)))$ if $\text{rk}(C) \geq 1$ and $f(C) = C$ if $\text{rk}(C) = 0$, for every $C \in N$. Then R' contains the rule $\rho' = A \to ((u'_1, \ldots, u'_n), \mathcal{L})$ where $u'_j = \text{cut}(u_j[f])$ for every $j \in [n]$. It can be shown that $L(G', A) = \{(\text{cut}(t_1), \ldots, \text{cut}(t_n)) \mid (t_1, \ldots, t_n) \in L(G, A)\}$ and so $L(G') = L(G)$. The formal proof, together with the proof of LDT^{R}-equivalence, is left to the reader.

If G' is Δ-lexicalized and $\Delta \subseteq \Sigma^{(0)}$, then G is Δ-lexicalized. In fact, the right-hand sides of ρ and ρ' contain the same elements of Δ because the only symbols that are removed or added have rank at least 2. We also observe that, for unrestricted $\Delta \subseteq \Sigma$, if G is n-Δ-lexicalized for $n > \mu(G)$, as defined before Lemma 47, then G' is $(n - \mu(G))$-Δ-lexicalized. In fact, in the definition of ρ' we have that for every $j \in [n]$, $|\text{pos}_\Delta(u_j[f])| \geq |\text{pos}_\Delta(u_j)|$ and $|\text{pos}_\Delta(u'_j)| \geq |\text{pos}_\Delta(u_j[f])| - 1$. ∎

[35] Otherwise, we replace every initial rule $S \to (u, \mathcal{L})$ by $S \to (\#(u), \mathcal{L})$ and after the construction remove # by Lemma 21.

For unrestricted Δ this theorem also holds except that G' is just equivalent to G, not necessarily $\mathrm{LDT}^{\mathrm{R}}$-equivalent. This follows from Lemma 47 and the last paragraph of the proof of Theorem 61. Thus, for every Δ-lexicalized MCFTG G with $\theta(G) \geq 2$ there is an equivalent Δ-lexicalized monadic MCFTG G' such that $\mu(G') \leq \mu(G) \cdot |\Sigma| \cdot mrk_{\Sigma}^2 \cdot (2 \cdot \theta(G) - 1)$.

Example 62 We consider the MCFTG $G = (N, \mathcal{N}, \Sigma, S, R)$ with $N = \{S, A^{(2)}, B^{(2)}\}$, $\mathcal{N} = \{S, (A, B)\}$, $\Sigma = \{\sigma^{(2)}, \tau^{(2)}, a^{(0)}, b^{(0)}, e^{(0)}\}$, and the rules

$$\begin{aligned} S &\to A(a, B(e, b)) \\ (A(x_1, x_2), B(x_1, x_2)) &\to (\sigma(a, A(x_1, x_2)),\ B(\tau(x_1, x_2), b)) \\ (A(x_1, x_2), B(x_1, x_2)) &\to (\sigma(x_1, x_2),\ \tau(x_1, x_2))\ . \end{aligned}$$

It generates the tree language $L(G) = \{(\sigma a)^n \tau^n e b^n \mid n \geq 1\}$. Note that we here use string notation. Thus, e.g., $(\sigma a)^2 \tau^2 e b^2$ is the tree $\sigma a \sigma a \tau \tau e b b$ which can be written as the term $\sigma(a, \sigma(a, \tau(\tau(e, b), b)))$. Obviously, G is an adjoining MCFTG with $\varphi(S) = \varphi(A) = \sigma$ and $\varphi(B) = \tau$. The equivalent monadic grammar G' as constructed in the proof of Theorem 61 has the rules

$$\begin{aligned} S &\to A(\sigma(a, B(\tau(e, b)))) \\ (A(x_1), B(x_1)) &\to (\sigma(a, A(x_1)),\ B(\tau(x_1, b))) \\ (A(x_1), B(x_1)) &\to (x_1,\ x_1)\ . \end{aligned}$$

Note that G' is not footed. □

As observed in the Introduction, Theorem 61 does not hold for spCFTGs; i.e., spCFTGs do not have the same tree generating power as monadic spCFTGs. In fact, it is shown in [30, Theorem 6.5] (see also [67, Lemma 24]) that spCFTGs (and arbitrary context-free tree grammars) give rise to a strict hierarchy with respect to $\theta(G)$. It is shown in [67, Theorem 10]) that every "straight-line" spCFTG can be transformed into an equivalent monadic one in polynomial time; the construction is similar to the one for Theorem 61 (in particular to the one in the proof of Theorem 49).

We finally observe that some tree languages in MCFT cannot be generated by an MCFTG that is both monadic and footed. An example is the language $L = \{(ca)^n (da)^n e \mid n \in \mathbb{N}_0\}$ that is generated by the spCFTG with rules $S \to A(e)$, $A(x_1) \to c(a, A(d(a, x_1)))$, and $A(x_1) \to x_1$. If G is a monadic footed MCFTG with $L(G) = L$, then G must be an MRTG because there is no terminal symbol of rank 1.[36] It follows from Theorem 76 in Section 8 and [81, p. 277] that all tree languages in MRT have regular "path languages". However, the intersection of the path language of L with $c^* d^* e$ is $\{c^n d^n e \mid n \in \mathbb{N}_0\}$, which is not regular. Thus, L is not in MRT (see also the last paragraph of Section 8).

7. Multiple context-free grammars

In this section we define the multiple context-free (string) grammars (MCFG) of [87, 92]. We first prove that MCFGs can be lexicalized. Then we prove that every tree language in MCFT can be generated by an MCFG, which is possible because we defined T_Σ as a subset of Σ^*. Using this we prove that MCFTGs have the same string generating power as MCFGs, by taking the yields of the generated tree languages. Moreover, we show that MCFTGs can be parsed in polynomial time.

7.1. String generating power of MCFTGs

To avoid the formalities involved in defining MCFGs in the classical way, we define them as a special case of MCFTGs. We introduce a special symbol $\triangleright$ of rank 0 and we identify, as usual, the strings over a finite (unranked) alphabet Σ with the trees over the "monadic" ranked alphabet $\Sigma \cup \{\triangleright\}$, where every symbol in Σ has rank 1. Thus, $w \in \Sigma^*$ is identified with $w \triangleright \in T_{\Sigma \cup \{\triangleright\}}$.

A *multiple context-free grammar* (in short, MCFG) is an MCFTG $G = (N \cup \{S\}, \mathcal{N}, \Sigma \cup \{\triangleright\}, S, R)$ such that $S \notin N$, every nonterminal in N has rank 1, $\triangleright \notin \Sigma$, and every terminal in Σ has rank 1.

[36] We already observed below Definition 48 that every tree of the forest $(t_1, \ldots, t_n) \in L(G, (A_1, \ldots, A_n))$ is footed. Suppose that $\mathrm{rk}(A_j) = 1$ for some $j \in [n]$, then the corresponding tree $t_j \in P_\Sigma(X_1)$ only contains terminal symbols (and the variable x_1). The foot node label of t_j must have rank 1, but the ranked alphabet Σ does not contain a unary symbol. Hence no unary nonterminal can be useful.

We also require (without loss of generality) that G is start-separated; i.e., that S does not occur in the right-hand sides of rules. With the above identification we have $L(G) \subseteq \Sigma^*$, and for every $A \in \mathcal{N} \setminus \{S\}$ we have $L(G, A) \subseteq P_\Sigma(X_1)^+$ and $P_\Sigma(X_1) = \Sigma^* x_1$. Note that every rule of G is either of the form $S \to (u \triangleright, \mathcal{L})$ with $u \in (N \cup \Sigma)^*$ or of the form $(A_1, \ldots, A_n) \to ((u_1 x_1, \ldots, u_n x_1), \mathcal{L})$ where $A_1, \ldots, A_n \in N$ and $u_1, \ldots, u_n \in (N \cup \Sigma)^*$. For a uniquely N-labeled tree $t = vCw \triangleright$ (or $vCwx_1$) with $v, w \in (N \cup \Sigma)^*$ and $C \in N$, the rewriting of C by ux_1 with $u \in (N \cup \Sigma)^*$ results in the tree $t[C \leftarrow ux_1]$, which equals $vuw \triangleright$ (or $vuwx_1$); thus, it is the usual rewriting of a nonterminal in a sentential form of a context-free grammar. It is straightforward to see that this definition of MCFG is equivalent to the classical notion of multiple context-free grammar [87, 92], taking into account the information-lossless condition (f3) of [87, Lemma 2.2]. The class of languages generated by MCFGs will be denoted by MCF.

Through the above identification of strings with monadic trees, MCFTGs can also generate strings directly as opposed to taking yields of the generated trees. In the next lemma we show that every MCFTG that generates strings in this way, has an equivalent MCFG.

Lemma 63 *For every MCFTG G with terminal alphabet $\Sigma \cup \{\triangleright\}$, where every symbol in Σ has rank 1, there is an $\mathrm{LDT}^{\mathrm{R}}$-equivalent MCFG G'. Moreover, $\mu(G') = \mu(G)$.*

Proof Due to the specific form of the terminal alphabet, it should be clear that reachable and useful big nonterminals cannot contain nonterminals of rank strictly larger than 1. Consequently, we may assume that G is monadic without the help of Theorem 61. We transform G into an MCFG G' with the same big nonterminals and the same nonterminals, which all have rank 1 in G' except for the initial nonterminal S of rank 0. Additionally, in the right-hand side of every initial rule we replace every occurrence of a nullary nonterminal C by $C(\triangleright)$, and in the right-hand side of every non-initial rule we replace every occurrence of a nullary nonterminal C by $C(x_1)$ and every occurrence of $\triangleright$ by x_1. ■

Let strMCFT denote the class of all string languages generated by MCFTGs, where strings over Σ are viewed as monadic trees over $\Sigma \cup \{\triangleright\}$ as explained above.

Corollary 64 strMCFT = MCF.

Another consequence of Lemma 63 is that MCFGs can be lexicalized, as stated in [95, Section 4.4] for the case $\Delta = \Sigma$. This should be contrasted to the fact that context-free grammars cannot be Δ-lexicalized for every Δ, as shown in Remark 53.

Corollary 65 *For every finitely Δ-ambiguous MCFG G there is a Δ-lexicalized MCFG G' that is $\mathrm{LDT}^{\mathrm{R}}$-equivalent to G. Moreover, $\mu(G') = \mu(G) + 1$.*

Proof By Theorem 44 there is an $\mathrm{LDT}^{\mathrm{R}}$-equivalent Δ-lexicalized MCFTG G' such that $\theta(G') = 2$ and $\mu(G') = \mu(G) + 1$. Next we apply Lemma 63. ■

Example 66 Consider the context-free grammar G with rules $S \to \ell AAr$, $A \to \ell AAr$, and $A \to \ell\tau abr$ (cf. Example 50 and Remark 53). Obviously, we may view G as an MCFG of multiplicity 1 with an alias A' of A. Its terminal alphabet is $\Sigma \cup \{\triangleright\}$ with $\Sigma = \{\tau^{(1)}, \ell^{(1)}, r^{(1)}, a^{(1)}, b^{(1)}\}$, and its rules for S and A are

$$S \to \ell A(A(r \triangleright)) \qquad A(x_1) \to \ell A(A'(rx_1)) \qquad \text{and} \qquad A(x_1) \to \ell\tau abrx_1 \ .$$

Let $\Delta = \{a, b\}$. Since G is Δ-growing, it has finite Δ-ambiguity. Applying a slightly simplified version of the proof of Corollary 65, we obtain a Δ-lexicalized MCFG G' of multiplicity 2 such that $L(G') = L(G)$. It has the big nonterminals $\{S, A, (B, C), (B', C')\}$, where (B', C') is an alias of (B, C), and the following rules, in which we omit $\triangleright$ and x_1 (and all the parentheses in trees) for readability:

$$S \to \ell ABbCr \qquad A \to \ell ABbCr \qquad A \to \ell\tau abr \qquad (B, C) \to (\ell B,\, CB'bC'r) \qquad (B, C) \to (\ell\tau a,\, r) \ .$$

Clearly, the string BbC generates the same terminal strings as A. □

In the next theorem, we show that every tree language that is generated by an MCFTG can also be generated by an MCFG provided that we change the ranks of the terminal symbols. In this theorem we (temporarily) identify each tree t over the ranked alphabet Σ (which is defined as a string over the unranked alphabet Σ) with the tree $t \triangleright$ over the ranked alphabet $\Sigma \cup \{\triangleright\}$, in which every symbol of Σ has rank 1. As an example, the tree $\sigma(a, b) = \sigma ab$ is identified with the tree $\sigma ab \triangleright = \sigma(a(b(\triangleright)))$. The idea behind the proof is essentially the same as the one of [28, Theorem 15]. In the case of an spCFTG, the resulting MCFG is well-nested (see [44, 55, 56, 72]).

Theorem 67 *For every MCFTG G there is an* $\mathrm{LDT}^{\mathrm{R}}$*-equivalent MCFG G'. If G is Δ-lexicalized, then so is G'. Moreover, $\mu(G') = \mu(G) \cdot (\theta(G) + 1)$ and $\lambda(G') = \lambda(G)$. If G is footed (i.e., is an nsMC-TAG) then $\mu(G') = 2 \cdot \mu(G)$.*

Proof By Lemma 22 we may assume that $G = (N, \mathcal{N}, \Sigma, S, R)$ is permutation-free. We will define the MCFG $G' = (N' \cup \{S'\}, \mathcal{N}' \cup \{S'\}, \Sigma \cup \{\triangleright\}, S', R')$, where S' is a new nonterminal and all the symbols in Σ now have rank 1. First of all, we let $N' = \{\langle C, i\rangle \mid C \in N,\ 0 \leq i \leq \mathrm{rk}(C)\}$. For every $C \in N^{(k)}$ the intuition behind this is that $\langle C, i\rangle(x_1)$ generates the string $w_i x_1$, when $C(x_1, \ldots, x_k)$ generates (as part of a big nonterminal) the terminal tree $w_0 x_1 w_1 \cdots x_k w_k \in PF_\Sigma(X_k)$ with $w_1, \ldots, w_k \in \Sigma^*$. For every $C \in N^{(k)}$, let its expansion be $\exp(C) = (\langle C, 0\rangle, \langle C, 1\rangle, \ldots, \langle C, k\rangle) \in (N')^+$, and for every $A = (A_1, \ldots, A_n) \in \mathcal{N}$, let $\exp(A) = \exp^*(A) = \exp(A_1) \cdots \exp(A_n) \in (N')^+$ be the concatenation of the expansions of its nonterminals. Then we define $\mathcal{N}' = \{\exp(A) \mid A \in \mathcal{N}\}$.

In the remainder of this proof we need the following two bijections π and λ. The right-hand side forest $u = (u_1 x_1, \ldots, u_n x_1) \in P_{N' \cup \Sigma}(X_1)^+$ of a possible non-initial rule of G' is in one-to-one correspondence with the string $\pi(u) = u_1 x_1 \cdots u_n x_1 \in (N' \cup \Sigma \cup X_1)^*$ that ends on x_1 and with the sequence $\lambda(u) = (u_1, \ldots, u_n)$ of strings $u_1, \ldots, u_n \in (N' \cup \Sigma)^*$. For the definition of the rules of G' we need the expansion of the right-hand side forests of the rules of G. For every $t \in T_{N \cup \Sigma}(X)$ we define $\exp(t) = \pi^{-1}(\exp'(t) \cdot x_1) \in P_{N' \cup \Sigma}(X_1)^+$, where π is the bijection defined above and where $\exp'(t) \in (N' \cup \Sigma \cup X_1)^*$ is defined inductively as follows:

$$\exp'(t) = \begin{cases} x_1 & \text{if } t \in X \\ \sigma \cdot \exp'(t_1) \cdots \exp'(t_k) & \text{if } t = \sigma(t_1, \ldots, t_k) \text{ with } \sigma \in \Sigma \\ \langle C, 0\rangle \cdot \exp'(t_1) \cdot \langle C, 1\rangle \cdots \exp'(t_k) \cdot \langle C, k\rangle & \text{if } t = C(t_1, \ldots, t_k) \text{ with } C \in N \end{cases} .$$

We note that $\exp'(t) = t[x \leftarrow x_1 \mid x \in X]$ if $t \in T_\Sigma(X)$. Given $t = (t_1, \ldots, t_n) \in T_{N \cup \Sigma}(X)^+$ we let $\exp(t) = \exp^*(t) = \exp(t_1) \cdots \exp(t_n)$ be the concatenation of the expansions of its elements.

Now, if $\rho = A \to (u, \{B_1, \ldots, B_k\}) \in R$, then R' contains the non-initial rule

$$\rho_{\exp} = \exp(A) \to (\exp(u), \{\exp(B_1), \ldots, \exp(B_k)\}) .$$

Clearly, the rule ρ can be reconstructed from $\rho_{\exp}$. Finally we define the initial rules of G'. If $\rho_{\exp} = \langle S, 0\rangle \to (v x_1, \mathcal{L})$ is a rule in R' as constructed above for $A = S$, then R' contains the additional rule $\rho'_{\exp} = S' \to (v \triangleright, \mathcal{L})$.[37] At this point, we completed the construction of G'. To prove its correctness we need the following claim.

Claim. Given a tree $t \in T_{N \cup \Sigma}(X)$, a repetition-free sequence $(A_1, \ldots, A_n) \in N^n$, and permutation-free patterns $s_1, \ldots, s_n \in PF_\Sigma(X)$ such that $\mathrm{rk}(A_i) = \mathrm{rk}(s_i)$ for every $i \in [n]$, we have

$$\exp(t[(A_1, \ldots, A_n) \leftarrow (s_1, \ldots, s_n)]) = \exp(t)[\exp(A_1) \cdots \exp(A_n) \leftarrow \exp(s_1) \cdots \exp(s_n)] .$$

Proof of claim. It can first be shown that $\exp'(t[(A_1, \ldots, A_n) \leftarrow (s_1, \ldots, s_n)]) = h^*(\exp'(t))$, where h is the string homomorphism over $N' \cup \Sigma \cup X_1$ such that the string $h(\langle A_i, j\rangle)$ is the $(j+1)$-th element of the sequence $\lambda(\exp(s_i))$, with the bijection λ defined above, for every $i \in [n]$ and $\langle A_i, j\rangle \in N'$. Moreover, h is the identity for the remaining elements of $N' \cup \Sigma \cup X_1$. The straightforward proof is left to the reader; it is by induction on the structure of t using the obvious fact that $\exp'(s[x_i \leftarrow t_i \mid 1 \leq i \leq k] = g^*(s)$ for all $s \in T_\Sigma(X_k)$ and $t_1, \ldots, t_k \in T_{N \cup \Sigma}(X)$, where g is the string homomorphism over $\Sigma \cup X_k$ such that $g(x_i) = \exp'(t_i)$ for $i \in [k]$ and $g(\sigma) = \sigma$ for $\sigma \in \Sigma$. Thus, the left-hand side of the equation is $\pi^{-1}(h^*(\exp'(t)) \cdot x_1)$. We now observe that this is equal to $\pi^{-1}(h^*(\pi(\exp(t))))$, which clearly equals the right-hand side of the equation. *This proves the claim.*

For every derivation tree $d \in L(G_{\mathrm{der}}, A)$ there is a derivation tree $d' \in L(G'_{\mathrm{der}}, \exp(A))$ such that $\mathrm{val}(d') = \exp(\mathrm{val}(d))$. In fact, d' is obtained from d by changing every label ρ simply into $\rho_{\exp}$. Let $d = \rho(d_1, \ldots, d_k)$, and let d'_i be such that $\mathrm{val}(d'_i) = \exp(\mathrm{val}(d_i))$. Now consider $d' = \rho_{\exp}(d'_1, \ldots, d'_k)$. Then we have

$$\mathrm{val}(d') = \exp(u)[\exp(B_1) \cdots \exp(B_k) \leftarrow \exp(\mathrm{val}(d_1)) \cdots \exp(\mathrm{val}(d_k))] \text{ and}$$

[37] The rule $\rho_{\exp} = \langle S, 0\rangle \to (v x_1, \mathcal{L})$ is then superfluous, but we keep it to simplify the correctness proof.

$$\mathrm{val}(d) = u[B_1 \cdots B_k \leftarrow \mathrm{val}(d_1) \cdots \mathrm{val}(d_k)] \ .$$

Thus $\mathrm{val}(d') = \exp(\mathrm{val}(d))$ by the above claim. Hence if $d \in L(G_{\mathrm{der}})$, then $d' \in L(G'_{\mathrm{der}}, \langle S, 0\rangle)$ and hence $d'' \in L(G'_{\mathrm{der}})$ where d'' is obtained from d' by priming the label of its root. Moreover, $\mathrm{val}(d') = \exp(\mathrm{val}(d)) = \mathrm{val}(d)x_1$. Hence, by Lemma 2, $\mathrm{val}(d'') = \mathrm{val}(d')[x_1 \leftarrow \triangleright] = \mathrm{val}(d)\triangleright = \mathrm{val}(d)$. This shows that $L(G) \subseteq L(G')$. Clearly, there is a two-state LDT-transducer that transforms d into d''. In fact, it is a finite-state relabeling. Since, obviously, every derivation tree in $L(G'_{\mathrm{der}})$ is of the form d'' with $d \in L(G_{\mathrm{der}})$, it also follows that $L(G') \subseteq L(G)$. Clearly, there is a one-state LDT-transducer that transforms d'' into d by changing every $\rho_{\exp}$ and $\rho'_{\exp}$ into ρ.

If G is footed, then the nonterminals $\langle C, 1\rangle, \ldots, \langle C, k-1\rangle$, where $k = \mathrm{rk}(C)$, are superfluous because they always generate x_1. Thus, in this case it suffices to define $\exp(C) = (\langle C, 0\rangle, \langle C, k\rangle)$ and adapt the construction accordingly. The resulting construction is similar to the one described in [93, Section 4.5.1] where it is shown that the yield of a tree language in MC-TAL is in MCF. ■

Example 68 We consider the permutation-free MCFTG $G = (N, \mathcal{N}, \Sigma, S, R)$ with $N = \{S, A^{(2)}, B^{(0)}\}$, $\mathcal{N} = \{S, (A, B)\}$, $\Sigma = \{\sigma^{(2)}, \alpha^{(0)}, \beta^{(0)}, \gamma^{(0)}\}$, and the following three rules:

$$S \to \sigma(A(\alpha, \beta), B)$$
$$\big(A(x_1, x_2), B\big) \to \big(\sigma(\alpha, A(\sigma(\beta, x_1), \sigma(\gamma, x_2))),\ \sigma(B, \beta)\big) \qquad \big(A(x_1, x_2), B\big) \to \big(\sigma(x_1, x_2),\ \gamma\big) \ .$$

Clearly, $L(G, (A, B))$ consists of all forests $\big((\sigma\alpha)^n\sigma(\sigma\beta)^n x_1(\sigma\gamma)^n x_2,\ \sigma^n\gamma\beta^n\big)$ with $n \in \mathbb{N}_0$. Consequently, $L(G) = \{\sigma(\sigma\alpha)^n\sigma(\sigma\beta)^n\alpha(\sigma\gamma)^n\beta\sigma^n\gamma\beta^n \mid n \in \mathbb{N}_0\}$. The MCFG G' constructed in the proof of Theorem 67 has the following four rules (in which we omit all parentheses in trees):

$$S' \to \sigma\langle A, 0\rangle\alpha\langle A, 1\rangle\beta\langle A, 2\rangle\langle B, 0\rangle\triangleright$$
$$\langle S, 0\rangle x_1 \to \sigma\langle A, 0\rangle\alpha\langle A, 1\rangle\beta\langle A, 2\rangle\langle B, 0\rangle x_1$$
$$(\langle A, 0\rangle x_1, \langle A, 1\rangle x_1, \langle A, 2\rangle x_1, \langle B, 0\rangle x_1) \to (\sigma\alpha\langle A, 0\rangle\sigma\beta x_1,\ \langle A, 1\rangle\sigma\gamma x_1,\ \langle A, 2\rangle x_1,\ \sigma\langle B, 0\rangle\beta x_1)$$
$$(\langle A, 0\rangle x_1, \langle A, 1\rangle x_1, \langle A, 2\rangle x_1, \langle B, 0\rangle x_1) \to (\sigma x_1,\ x_1,\ x_1,\ \gamma x_1) \ .$$

Clearly, $L(G', (\langle A, 0\rangle, \langle A, 1\rangle, \langle A, 2\rangle, \langle B, 0\rangle)) = \big\{\big((\sigma\alpha)^n\sigma(\sigma\beta)^n x_1,\ (\sigma\gamma)^n x_1,\ x_1,\ \sigma^n\gamma\beta^n x_1\big) \mid n \in \mathbb{N}_0\big\}$ and hence $L(G') = L(G)$. The second rule of G' is of course superfluous.

For the next lemma and corollary we note that the MCFG G'' that is obtained from G' by removing σ and thus has the rules

$$S' \to \langle A, 0\rangle\alpha\langle A, 1\rangle\beta\langle A, 2\rangle\langle B, 0\rangle\triangleright$$
$$(\langle A, 0\rangle x_1, \langle A, 1\rangle x_1, \langle A, 2\rangle x_1, \langle B, 0\rangle x_1) \to (\alpha\langle A, 0\rangle\beta x_1,\ \langle A, 1\rangle\gamma x_1,\ \langle A, 2\rangle x_1,\ \langle B, 0\rangle\beta x_1)$$
$$(\langle A, 0\rangle x_1, \langle A, 1\rangle x_1, \langle A, 2\rangle x_1, \langle B, 0\rangle x_1) \to (x_1,\ x_1,\ x_1,\ \gamma x_1) \ ,$$

generates the string language $\mathrm{yd}(L(G)) = \{\alpha^n\beta^n\alpha\gamma^n\beta\gamma\beta^n \mid n \in \mathbb{N}_0\}$. □

Theorem 67 suggests that we do not need MCFTGs at all, because MCFGs can generate the "same" languages. However, the MCFTG is a way of guaranteeing that all intermediate results during the generation process are trees, which supports the structured generation of the trees.

It follows from Theorem 67 that known properties of MCF languages (see, e.g., [54, 87, 92]) also hold for MCFT tree languages. Thus MCFT $\subseteq$ LOG(CFL); i.e., the recognition problem for an MCFT tree language is log-space reducible to that of a context-free string language. Also, every tree language generated by an MCFTG G can be parsed in polynomial time by first parsing the given tree according to the MCFG G' of Theorem 67 in polynomial time and then transforming the resulting derivation tree of G' by the corresponding $\mathrm{LDT}^{\mathrm{R}}$-transducer into one of G in linear time. This will be discussed in more detail in Section 7.2. Additionally, every MCFT tree language is semi-linear.

Next we show that MCFGs generate exactly the yield languages of the tree languages generated by MCFTGs. We recall that the yield of a tree $t \in T_\Sigma$ is defined as $\mathrm{yd}(t) = \mathrm{yd}_{\Sigma^{(0)}\setminus\{e\}}(t)$, where e is a special symbol e of rank 0 that satisfies $\mathrm{yd}(e) = \varepsilon$. For a class $\mathcal{X}$ of tree languages, let $\mathrm{y}\mathcal{X}$ be the class of all languages $\mathrm{yd}(L)$ with $L \in \mathcal{X}$. Thus, we will show that yMCFT = MCF. In fact, this is already a consequence of (the second equation of) Corollary 59 in Section 6.2, which implies that yMCFT = yMC-TAL, and the equation yMC-TAL = MCF which was shown in [93].[38] We additionally

[38] The equality yMCFT = MCF is also stated in [8, Theorem 1].

prove $\mathrm{LDT^R}$-yd-equivalence, for which we refer to Definition 15. In the first half of the next lemma we consider, more generally, a subset $\Delta \subseteq \Sigma$ of lexical symbols and we prove $\mathrm{LDT^R}$-yd_Δ-equivalence (where yd_Δ is defined in the paragraph on homomorphisms in Section 2.1). This general case will be used in the proof of Theorem 72.

Lemma 69 *Let $\Delta \subseteq \Sigma$.*
(1) For every MCFTG G there is an MCFG G'' that is $\mathrm{LDT^R}$-yd_Δ-equivalent to G.
(2) For every MCFG G there is an MRTG G_1 such that G is $\mathrm{LDT^R}$-yd-equivalent to G_1.

Proof It is straightforward to generalize the well-known proofs for RTGs and context-free grammars (see, e.g., [16, Theorem 3.28]). To prove statement (1), let $G = (N, \mathcal{N}, \Sigma, S, R)$ be an MCFTG, and let $G' = (N', \mathcal{N}', \Sigma \cup \{\triangleright\}, S', R')$ be the $\mathrm{LDT^R}$-equivalent MCFG that exists by Theorem 67.[39] Clearly, the mapping yd_Δ is a tree homomorphism over the monadic ranked alphabet $\Sigma \cup \{\triangleright\}$. To be precise, let h be the tree homomorphism from $\Sigma \cup \{\triangleright\}$ to $\Delta \cup \{\triangleright\}$ such that $h(\alpha) = x_1$ if $\alpha \in \Sigma \setminus \Delta$ and $h(\alpha) = \mathrm{in}(\alpha)$ otherwise. Then $\hat{h}(t\triangleright) = \mathrm{yd}_\Delta(t)\triangleright$ for every $t \in T_\Sigma$, and so $\hat{h} = \mathrm{yd}_\Delta$. Now let G'' be the grammar G'_h as defined before Lemma 21. Clearly, G'' is again an MCFG and $\mathrm{LDT^R}$-$\hat{h}$-equivalent to G' by that lemma. Since G' and G are $\mathrm{LDT^R}$-equivalent, it follows that G'' is $\mathrm{LDT^R}$-yd_Δ-equivalent to G.

To prove statement (2), let $G = (N, \mathcal{N}, \Sigma \cup \{\triangleright\}, S, R)$ be an MCFG. We construct the MRTG $G_1 = (N, \mathcal{N}, \Sigma \cup \{e, c\}, S, R_1)$, where c is a new terminal symbol of rank 2, and all symbols of $\Sigma \cup \{e\}$ and N have rank 0. The new set R_1 of rules is obtained by replacing each rule $\rho = A \to ((u_1, \ldots, u_n), \mathcal{L})$ of G by the rule $\rho' = A \to ((u'_1, \ldots, u'_n), \mathcal{L})$ of G_1, where $u'_1, \ldots, u'_n$ are defined as follows. For $u \in (N \cup \Sigma)^*\{\triangleright, x_1\}$, if $u \in \{\triangleright, x_1\}$, then $u' = e$, and if $u = \gamma v$ with $\gamma \in N \cup \Sigma$, then $u' = c(\gamma, v')$. Note that ρ can be reconstructed from ρ'. It should be clear that G is $\mathrm{LDT^R}$-yd-equivalent to G_1 because the derivation trees of G_1 are the primed versions of the derivation trees of G. ∎

Recall that if G' is $\mathrm{LDT^R}$-yd-equivalent to G, then $L(G') = \mathrm{yd}(L(G))$. Thus, we immediately obtain from Lemma 69 (with $\Delta = \Sigma^{(0)} \setminus \{e\}$) that MCFGs generate the yield languages of the tree languages generated by MCFTGs.

Corollary 70 yMCFT = MCF = yMRT.

Thus, strMCFT = yMCFT by Corollary 64. This is quite unusual for a class of tree languages as already observed at the end of [28, Section 4]. For instance, the monadic tree languages generated by RTGs are the regular string languages, whereas the yield languages are the context-free string languages.

The proof of MRT $\subseteq$ MCF, and hence of MCF = yMRT, is also straightforward (cf. Example 6). For an MRTG $G = (N, \mathcal{N}, \Sigma, S, R)$, we construct the MCFG $G' = (N \cup \{S'\}, \mathcal{N} \cup \{S'\}, \Sigma, S', R')$, where the set R' consists of all rules $A \to ((u_1x_1, \ldots, u_nx_1), \mathcal{L})$ such that $A \to ((u_1, \ldots, u_n), \mathcal{L}) \in R$ and all rules $S' \to (v\triangleright, \mathcal{L})$ such that $S \to (v, \mathcal{L}) \in R$. Then $L(G') = L(G)$ because this construction is a special case of the construction in the proof of Theorem 67 if, in that proof, $\langle C, 0\rangle$ is identified with C for every $C \in N$. Note that in the constructions that prove MCF = yMRT the multiplicity of the grammars is preserved.

As observed before Theorem 67, the above proofs show that $\mathrm{CFT_{sp}} \subseteq \mathrm{MCF_{wn}}$ and $\mathrm{yCFT_{sp}} \subseteq \mathrm{MCF_{wn}}$, where $\mathrm{MCF_{wn}}$ denotes the class of languages generated by well-nested MCFGs. It is, in fact, not difficult to prove that $\mathrm{yCFT_{sp}} = \mathrm{MCF_{wn}}$ as stated in [56]. The multiplicity of the well-nested MCFG equals one plus the width of the spCFTG. It is proved in [59] that $\mathrm{MCF_{wn}}$ is properly included in MCF.

7.2. Parsing of MCFTGs

In the remainder of this section we consider the parsing problem for MCFTGs. We start by showing the well-known fact (cf., e.g., [54, 87]) that every MCFG G can be parsed in polynomial time in the sense that given a string w as input, the parsing algorithm outputs an RTG H_w that generates all derivation trees of G with value w. In fact, the usual CYK parsing algorithm for MCFGs constructs the RTG H_w in such a way that all its nonterminals are useful. Clearly, $w \in L(G)$ if and only if $L(H_w) \neq \emptyset$, which can be tested in linear time. Moreover, a derivation tree with value w can be computed from H_w in linear time provided that $L(H_w) \neq \emptyset$. In the next lemma we also state the degree of the polynomial, as taken

[39] For the purpose of this proof, there is no need to reconsider the construction of G' in its proof.

from [87].[40] It involves both the multiplicity $\mu(G)$ and the rule-width $\lambda(G)$ of G.[41] It should be noted that, as shown in [84] (see also [6, 52]), the uniform membership problem for MCFGs is NP-hard, even when $\mu(G)$ or $\lambda(G)$ is fixed (except of course for $\mu(G) = 1$ and for the trivial case $\lambda(G) = 0$).

Lemma 71 *For every MCFG G with terminal alphabet $\Sigma \cup \{\triangleright\}$ there is a polynomial time algorithm that, on input $w \in \Sigma^*$, outputs an RTG H_w such that $L(H_w) = \{d \in L(G_{\mathrm{der}}) \mid \mathrm{val}(d) = w\}$. The degree of the polynomial is $\mu(G) \cdot (\lambda(G) + 1)$.*

Proof Let $G = (N \cup \{S\}, \mathcal{N}, \Sigma \cup \{\triangleright\}, S, R)$ and $w \in \Sigma^*$. Moreover, let $w = \sigma_1 \cdots \sigma_n$ with $n \in \mathbb{N}_0$ and $\sigma_1, \ldots, \sigma_n \in \Sigma$. We define the set of positions of w by $\mathrm{pos}(w) = \{0, 1, \ldots, n\}$. Intuitively, position 0 is just before σ_1 and position i is just after σ_i for every $i \in [n]$. For positions $i, j \in \mathrm{pos}(w)$ with $i \leq j$ we let $w[i, j] = \sigma_{i+1} \cdots \sigma_j$ be the substring of w between positions i and j. Note that $w[i, i] = \varepsilon$ for every $i \in \mathrm{pos}(w)$.

The construction of H_w is similar to the usual "triple construction" for proving that the intersection of a context-free language with a regular language is again context-free (in this case the regular language $\{w\}$). We construct the RTG $H_w = (N_w, R, S_w, R_w)$, in which N_w is the set of all sequences $(\langle \ell_1, A_1, r_1\rangle, \ldots, \langle \ell_m, A_m, r_m\rangle)$ such that $(A_1, \ldots, A_m) \in \mathcal{N}$ and $0 \leq \ell_i \leq r_i \leq n$ for all $j \in [m]$. Moreover, $S_w = \langle 0, S, n\rangle$. The idea of the proof is that $(\langle \ell_1, A_1, r_1\rangle, \ldots, \langle \ell_m, A_m, r_m\rangle)$ generates all derivation trees $d \in L(G_{\mathrm{der}}, (A_1, \ldots, A_m))$ such that $\mathrm{val}(d) = (w[\ell_1, r_1], \ldots, w[\ell_m, r_m])$.

We now define the set R_w of rules of H_w. Let $\rho = A \to (u, \mathcal{L})$ be a rule in R with $A = (A_1, \ldots, A_m)$, $\mathcal{L} = \{B_1, \ldots, B_k\}$, and $u = (u_1x_1, \ldots, u_mx_1)$ if $A \neq S$ and $u = u_1 \triangleright$ otherwise (with $A_j \in N$ and $u_j \in \Sigma^*$ for every $j \in [m]$). Moreover, let $\ell_1, r_1, \ldots, \ell_m, r_m \in \mathrm{pos}(w)$ and let ℓ and r be mappings from $\mathrm{occ}_N(u)$ to $\mathrm{pos}(w)$ such that

(a) $\ell_i \leq r_i$ for every $i \in [m]$ and $\ell(C) \leq r(C)$ for every $C \in \mathrm{occ}_N(u)$,
(b) for every $j \in [m]$, if $u_j = v_0C_1v_1 \cdots C_pv_p$ with $p \in \mathbb{N}_0$, $v_0, v_i \in \Sigma^*$, and $C_i \in N$ for every $i \in [p]$, then
  (1) $v_0 = w[\ell_j, \ell_j + |v_0|]$ and $v_i = w[r(C_i), r(C_i) + |v_i|]$ for every $i \in [p]$,
  (2) $\ell(C_1) = \ell_j + |v_0|$ and $\ell(C_{i+1}) = r(C_i) + |v_i|$ for every $i \in [p-1]$,
  (3) $r_j = l_j + |v_0|$ if $p = 0$ and $r_j = r(C_p) + |v_p|$ otherwise.

Then the set R_w contains the rule $(\langle \ell_1, A_1, r_1\rangle, \ldots, \langle \ell_m, A_m, r_m\rangle) \to \rho(\hat{h}(B_1), \ldots, \hat{h}(B_k))$, where h is the string homomorphism from $\mathrm{occ}_N(u)$ to $\mathrm{pos}(w) \times N \times \mathrm{pos}(w)$ such that $h(C) = \langle \ell(C), C, r(C)\rangle$ for every $C \in \mathrm{occ}_N(u)$. Note that $\mathrm{occ}_N(u) = \bigcup_{i=1}^{k} \mathrm{occ}(B_i)$.

The above proof idea can easily be shown by induction on the structure of d. Thus, S_w generates all derivation trees in $d \in L(G_{\mathrm{der}})$ such that $\mathrm{val}(d) = w[0, n] = w$. Before constructing the rules of R_w, the set $\{i \in \mathrm{pos}(w) \mid v = w[i, i + |v|]\}$ can be computed for every string $v \in \Sigma^*$ that occurs in a rule of R. Since G is fixed, this can be done in linear time and takes care of the conditions in (1) above. When constructing a rule in R_w corresponding to the rule $\rho \in R$ as above, it clearly suffices to choose $\ell_1, \ldots, \ell_m$ and the mapping r, because $r_1, \ldots, r_m$ are determined by (3) above and the mapping ℓ is determined by (2) above. Since each rule of R_w can be constructed in constant time, constructing the rules corresponding to ρ takes time $O(n^q)$ where $q = m + \sum_{i=1}^{k} |B_i|$ is the number of possible choices of $\ell_1, \ldots, \ell_m$ and r. Thus, the algorithm runs in time $O(n^k)$ where $k = \mu(G) + \lambda(G) \cdot \mu(G) = \mu(G) \cdot (\lambda(G) + 1)$.

We note that the set N_w can be constructed in quadratic time.[42] In fact, it should be clear that H_w can be constructed in such a way that only useful nonterminals occur in its rules. Such a construction corresponds directly to a CYK parsing algorithm. ∎

We now generalize this result to MCFTGs. Let G be an MCFTG with terminal alphabet Σ, and let $\Delta \subseteq \Sigma^{(0)} \setminus \{e\}$ be a set of lexical symbols. We can use the MCFTG G to specify the MCF string language $\mathrm{yd}_\Delta(L(G))$ together with a set of "syntactic trees", where every tree t in $L(G)$ is viewed as a syntactic tree for the string $\mathrm{yd}_\Delta(t)$. In such a case, the parsing problem for G amounts to finding the syntactic trees for a given string over Δ.

[40] In [87] a recognition algorithm is presented for MCFGs in a certain normal form. In [54, Section 7] a parsing algorithm is presented for all MCFGs, with the RTG defined as a chart with back-pointers, but the degree of the polynomial is not analyzed.

[41] As defined after Definition 5, the rule-width of G is $\lambda(G) = \max\{|\mathcal{L}(\rho)| \mid \rho \in R\}$ where R is the set of rules of G.

[42] In the trivial case where $\lambda(G) = 0$ (and hence $\mu(G) = 1$) we can take $N_w = \{S_w\}$.

Theorem 72 *For every MCFTG G with terminal alphabet Σ and every $\Delta \subseteq \Sigma$, there is a polynomial time algorithm that, on input $w \in \Delta^*$, outputs an RTG H_w and an MCFTG G_w such that*

$$L(H_w) = \{d \in L(G_{\mathrm{der}}) \mid \mathrm{yd}_\Delta(\mathrm{val}(d)) = w\} \qquad \textit{and} \qquad L(G_w) = \{t \in L(G) \mid \mathrm{yd}_\Delta(t) = w\} \ .$$

The degree of the polynomial is $\mu(G) \cdot (\theta(G)+1) \cdot (\lambda(G)+1)$. If G is footed (i.e., is an nsMC-TAG) then the degree is $2 \cdot \mu(G) \cdot (\lambda(G)+1)$.

Proof Let G' be the $\mathrm{LDT}^{\mathrm{R}}$-$\mathrm{yd}_\Delta$-equivalent MCFG that exists by Lemma 69(1), and let M be the $\mathrm{LDT}^{\mathrm{R}}$-transducer from G to G'. It can easily be verified that $\mu(G') = \mu(G) \cdot (\theta(G)+1)$ and $\lambda(G') = \lambda(G)$, and that M is a (composition of) finite-state relabeling(s). Moreover, let $w \in \Delta^*$. By Lemma 71 we can construct an RTG H'_w such that $L(H'_w) = \{d \in L(G'_{\mathrm{der}}) \mid \mathrm{val}(d) = w\}$, in the required polynomial time. Then, by Proposition 14 and using a product construction with the RTG G_{der}, we construct in linear time an RTG H_w such that

$$L(H_w) = M^{-1}(L(H'_w)) \cap L(G_{\mathrm{der}}) = \{d \in L(G_{\mathrm{der}}) \mid \mathrm{val}(M(d)) = w\} \ ,$$

which satisfies the requirement because $\mathrm{val}(M(d)) = \mathrm{yd}_\Delta(\mathrm{val}(d))$. It remains to construct G_w from G and H_w, which we achieve in linear time by an easy product construction. Let $G = (N, \mathcal{N}, \Sigma, S, R)$ be the MCFTG and $H_w = (N_w, R, S_w, R_w)$ be the constructed RTG. We construct $G_w = (N', \mathcal{N}', \Sigma, S', R')$ such that $N' = N \times N_w$ and $\mathcal{N}'$ consists of all $(\langle A_1, C\rangle, \ldots, \langle A_n, C\rangle)$ with $(A_1, \ldots, A_n) \in \mathcal{N}$ and $C \in N_w$. For $A = (A_1, \ldots, A_n)$, we denote $(\langle A_1, C\rangle, \ldots, \langle A_n, C\rangle)$ by $A \otimes C$. The initial nonterminal of G_w is $S' = S \otimes S_w = \langle S, S_w\rangle$. If $A \to (u, \mathcal{L})$ is a rule in R with $\mathcal{L} = \{B_1, \ldots, B_k\}$ and $C_0 \to \rho(C_1, \ldots, C_k)$ is a rule in R_w, then R' contains the rule $A \otimes C_0 \to (u', \mathcal{L}')$, in which $u' = u[B_i \leftarrow \mathrm{in}(B_i \otimes C_i) \mid 1 \le i \le k]$ and $\mathcal{L}' = \{B_1 \otimes C_1, \ldots, B_k \otimes C_k\}$. It is easy to show that $L(G_w, A \otimes C) = \mathrm{val}(L(G_{\mathrm{der}}, A) \cap L(H_w, C))$ for every big nonterminal $A \otimes C \in \mathcal{N}'$. Hence $L(G_w) = \mathrm{val}(L(H_w))$, which shows that G_w satisfies the requirement. ∎

For $\Delta = \Sigma$ this theorem shows that MCFTGs can be parsed as tree grammars in polynomial time. For every input tree $t \in T_\Sigma$ the parsing algorithm produces as output an RTG H_t such that $L(H_t) = \{d \in L(G_{\mathrm{der}}) \mid \mathrm{val}(d) = t\}$. The algorithm can easily be extended to test in linear time whether or not $t \in L(G)$ by testing whether $L(H_t)$ is nonempty. Additionally, if $L(H_t) \neq \emptyset$, then it can also compute in linear time an element of $L(H_t)$; i.e., a derivation tree $d \in L(G_{\mathrm{der}})$ such that $\mathrm{val}(d) = t$.

For $\Delta \subseteq \Sigma^{(0)} \setminus \{e\}$ we are in the situation described before the theorem. For every input string $w \in \Delta^*$ the parsing algorithm outputs an MCFTG G_w such that $L(G_w)$ is the set of all syntactic trees $t \in L(G)$ with $\mathrm{yd}_\Delta(t) = w$. Using H_w as in the previous case, the algorithm can be extended to test in linear time whether $w \in \mathrm{yd}_\Delta(L(G))$, and if so compute a derivation tree $d \in L(G_{\mathrm{der}})$ such that $\mathrm{yd}_\Delta(\mathrm{val}(d)) = w$. Moreover, it can then compute $t = \mathrm{val}(d)$ in linear time; i.e., a syntactic tree $t \in L(G)$ with $\mathrm{yd}_\Delta(t) = w$.

We note that, by the proof of Theorem 72, these parsing algorithms are directly based on a parsing algorithm for MCFGs; i.e., any algorithm that satisfies Lemma 71. If such a parsing algorithm for the $\mathrm{LDT}^{\mathrm{R}}$-$\mathrm{yd}_\Delta$-equivalent MCFG G' does not output an RTG H'_w for *all* derivation trees d' with value w, but outputs just *one* such derivation tree d', then there is no need to construct H_w and G_w because the above derivation tree $d \in L(G_{\mathrm{der}})$ and syntactic tree $t \in L(G)$ can be obtained in linear time as $d = M'(d')$ and $t = \mathrm{val}(d)$, where M' is the $\mathrm{LDT}^{\mathrm{R}}$-transducer from G' to G.

8. Characterization

In this section we prove that MCFT is equal to the class $\mathrm{DMT}_{\mathrm{fc}}(\mathrm{RT})$ of images of the regular tree languages under (total) deterministic finite-copying macro tree transducers, and hence equal to the class DMSOT(RT) of images of the regular tree languages under (total) deterministic MSO tree transducers.[43] After proving this result we discuss a number of consequences, in particular several alternative characterizations of MCFT. As opposed to the usual notation in the literature [26, 27, 34, 39], we use Y as the set of input variables and X as the set of output variables (or parameters) for macro tree transducers. We only consider *total deterministic* macro tree transducers that are *simple* (i.e., linear and nondeleting) *in the parameters*; this is indicated by 'D' and 'sp', respectively.

[43] Since the domain of a macro tree transduction is a regular tree language [34, Theorem 7.4], the class $\mathrm{DMT}_{\mathrm{fc}}(\mathrm{RT})$ does not depend on the totality of the transducers. The same is true for MSO tree transductions and the class DMSOT(RT).

A *macro tree transducer* (in short, $\mathrm{DMT_{sp}}$-transducer) is a system $M = (Q, \Omega, \Sigma, q_0, R)$, where Q is a finite ranked alphabet of *states*, Ω and Σ are finite ranked alphabets of *input* and *output symbols*, respectively, with $Q \cap \Sigma = \emptyset$, $q_0 \in Q^{(0)}$ is the *initial state*, and R is a finite set of *rules*. For every $q \in Q^{(m)}$ and $\omega \in \Omega^{(k)}$ with $m, k \in \mathbb{N}_0$ there is exactly one rule of the form

$$\langle q, \omega(y_1, \ldots, y_k)\rangle(x_1, \ldots, x_m) \to \zeta$$

in R such that $\zeta \in P_{(Q\times Y_k)\cup\Sigma}(X_m)$, where every element $\langle q', y_i\rangle$ of $Q \times Y_k$ has the same rank as q'. We denote ζ by $\mathrm{rhs}_M(q, \omega)$.

For every input tree $s \in T_\Omega$ and every state $q \in Q$, the *q-translation* of s by M, denoted by $M_q(s)$, is a tree in $P_\Sigma(X_{\mathrm{rk}(q)})$ defined inductively as follows. Let $s = \omega(s_1, \ldots, s_k)$ and consider the above rule. Then $M_q(s) = \zeta[\langle q', y_i\rangle \leftarrow M_{q'}(s_i) \mid q' \in Q, 1 \le i \le k]$. As in the case of $\mathrm{LDT^R}$-transducers, we define $M(s) = M_{q_0}(s)$ and call it the *translation* of s by M. Since q_0 has rank 0, $M(s)$ is a tree in T_Σ. The *tree transduction realized by* M, also denoted by M, is the total function $M = \{(s, M(s)) \mid s \in T_\Omega\}$ from T_Ω to T_Σ. A $\mathrm{DMT_{sp}}$-transducer is a (total deterministic) *top-down tree transducer* (in short, DT-transducer) if all its states have rank 0.

Finite-copying macro tree transducers were introduced in [26]. To define them, we need the well-known notion of "state sequence" (cf. [30, Definition 3.1.8]). Let $(q_1, \ldots, q_n) \in Q^*$ with $n \in \mathbb{N}_0$ and $q_1, \ldots, q_n \in Q$, and let $\omega \in \Omega^{(k)}$ for some $k \in \mathbb{N}_0$. For $i \in [k]$ we define $\mathrm{sts}_{\omega,i}(q_1, \ldots, q_n) \in Q^*$ to be the sequence of states

$$\mathrm{sts}_{\omega,i}(q_1, \ldots, q_n) = \pi_i^*(\mathrm{rhs}_M(q_1, \omega) \cdots \mathrm{rhs}_M(q_n, \omega)) \ ,$$

where π_i is the string homomorphism from $(Q\times Y_k)\cup\Sigma\cup X$ to Q such that $\pi_i(\langle q', y_i\rangle) = q'$ for every $q' \in Q$ and $\pi_i(\alpha) = \varepsilon$ for every $\alpha \in \Sigma \cup X$. For $s \in T_\Omega$ and $p \in \mathrm{pos}(s)$, we define the *state sequence* of M at p, denoted by $\mathrm{sts}(s, p)$, inductively as follows: (i) $\mathrm{sts}(s, \varepsilon) = q_0$ and (ii) if $\mathrm{sts}(s, p) = (q_1, \ldots, q_n)$ and $s(p) = \omega \in \Omega^{(k)}$, then $\mathrm{sts}(s, pi) = \mathrm{sts}_{\omega,i}(q_1, \ldots, q_n)$ for every $i \in [k]$. The *set of state sequences* of M, denoted by $\mathrm{sts}(M)$, is defined by $\mathrm{sts}(M) = \{\mathrm{sts}(s, p) \mid s \in T_\Omega, p \in \mathrm{pos}(s)\}$. Note that it is the smallest subset S of Q^* such that (i) $q_0 \in S$ and (ii) if $\overline{q} \in S$, then $\mathrm{sts}_{\omega,i}(\overline{q}) \in S$ for all $k \in \mathbb{N}_0$, $\omega \in \Omega^{(k)}$, and $i \in [k]$. We say that the $\mathrm{DMT_{sp}}$-transducer M is *finite-copying* (in short, $\mathrm{DMT_{fc}}$-transducer) if $\mathrm{sts}(M)$ is finite; it is *m-copying* for $m \in \mathbb{N}$, if the state sequences in $\mathrm{sts}(M)$ have length at most m. A $\mathrm{DT_{fc}}$-transducer is a finite-copying DT-transducer.

For a notion of $\mathcal{X}$-transducer, we denote by $\mathcal{X}$ the class of transductions realized by $\mathcal{X}$-transducers. For a class $\mathcal{X}$ of transductions, we denote by $\mathcal{X}(\mathrm{RT})$ the class of all tree languages $M(L)$, where $M \in \mathcal{X}$ and $L \in \mathrm{RT}$ is a regular tree language.

The finite-copying macro tree transducers of [26] are not necessarily simple; i.e., linear and nondeleting in the parameters. However, it follows from the results of [26, Section 6] that adding the feature of regular look-ahead, which we do not need here, to the above finite-copying macro tree transducers yields the same expressive power as in [26]. In particular, our notion of state sequence corresponds to the one in Definition 6.8 and Lemma 6.9 of [26]. Since regular look-ahead can be simulated by a relabeling of the input tree (see [17]), the class $\mathrm{DMT_{fc}}(\mathrm{RT})$, which we are interested in here, coincides with the one in [26] (denoted $\mathrm{MTT_{fc}}(\mathrm{REGT})$ there). Let us finally note that it is decidable whether or not a macro tree transducer is finite-copying [29, Lemma 4.10], and if so, its set of state sequences can be computed by iteration.

The inclusion $\mathrm{MCFT} \subseteq \mathrm{DMT_{fc}}(\mathrm{RT})$ is a direct consequence of the next lemma and Theorem 9. The lemma shows that 'val' can be realized by a $\mathrm{DMT_{fc}}$-transducer. In its proof we use the following additional terminology. For $\bar{q} = (q_1, \ldots, q_n) \in Q^+$ with $n \in \mathbb{N}$ and $q_1, \ldots, q_n \in Q$, we define the $\bar{q}$-translation of $s \in T_\Omega$ by $M_{\bar{q}}(s) = (M_{q_1}(s), \ldots, M_{q_n}(s))$.

Lemma 73 *For every MCFTG G there is a DMT_{fc}-transducer M such that $M(d) = \mathrm{val}(d)$ for every $d \in L(G_{\mathrm{der}})$. If G is an MRTG, then M is a DT_{fc}-transducer.*

Proof Let $G = (N, \mathcal{N}, \Sigma, S, R)$ be an MCFTG. Since the result is obvious if $L(G) = \emptyset$, we may assume that $\Sigma^{(0)} \neq \emptyset$. We construct the macro tree transducer $M = (N, R, \Sigma, S, R_M)$. Thus, M uses the nonterminals of G with the same rank as states, of which S is the initial state. Moreover, the input alphabet is R and the output alphabet is Σ. If $\rho = (A_1, \ldots, A_n) \to ((u_1, \ldots, u_n), \mathcal{L})$ is a rule in R such that $\mathcal{L} = \{B_1, \ldots, B_k\}$ with $B_1, \ldots, B_k \in \mathcal{N}$, then R_M contains the following rule for every $j \in [n]$:

$$\langle A_j, \rho(y_1, \ldots, y_k)\rangle(x_1, \ldots, x_{\mathrm{rk}(A_j)}) \to u_j[C \leftarrow \mathrm{in}(\langle C, y_i\rangle) \mid C \in \mathrm{occ}(B_i),\ 1 \le i \le k] \ .$$

Moreover, it has the (dummy) rule $\langle C, \rho(y_1, \ldots, y_k)\rangle(x_1, \ldots, x_m) \to t_m$ for every $C \in N \setminus \{A_1, \ldots, A_n\}$ of rank m, where t_m is an arbitrary element of $P_\Sigma(X_m)$.[44]

Clearly, M is simple in the parameters because $u_j \in P_{N\cup\Sigma}(X_{\mathrm{rk}(A_j)})$. Let $d \in L(G_{\mathrm{der}})$. We claim that $\mathrm{sts}(d, p)$, the state sequence of M at a node p of d, is a permutation of the left-hand side of the rule $d(p)$ of G. This is obvious for the root of d with state sequence S, and if it holds for p, then it holds for pi for every $i \in [k]$ by the definition of the above rules of M. Hence M is finite-copying on $L(G_{\mathrm{der}})$. It is, in fact, finite-copying everywhere because the state sequence becomes empty due to the dummy rules as soon as there is a type error in the input tree (which means that the input tree is not a derivation tree of G).

We now claim that $M_A(d) = \mathrm{val}(d)$ for every $A \in \mathcal{N}$ and every derivation tree $d \in L(G_{\mathrm{der}}, A)$, where $M_A(d)$ is defined just before this lemma. The proof is by induction on the structure of d. Let $d = \rho(d_1, \ldots, d_k)$. For the above rule ρ of G, let $A = (A_1, \ldots, A_n)$ and $u = (u_1, \ldots, u_n)$. Then $\mathrm{val}(d) = u[B_i \leftarrow \mathrm{val}(d_i) \mid 1 \le i \le k]$. From the definition of the rules of M we obtain that

$$M_A(d) = u[C \leftarrow M_C(d_i) \mid C \in \mathrm{occ}(B_i),\, 1 \le i \le k] = u[B_i \leftarrow M_{B_i}(d_i) \mid 1 \le i \le k] \ .$$

By the induction hypotheses, $M_{B_i}(d_i) = \mathrm{val}(d_i)$ for every $i \in [k]$. Consequently, $M_A(d) = \mathrm{val}(d)$. In particular, if $d \in L(G_{\mathrm{der}})$, then $M(d) = M_S(d) = \mathrm{val}(d)$. ∎

For the converse inclusion we need a normal form for $\mathrm{DMT}_{\mathrm{fc}}$-transducers from [26], which is based on the same result for $\mathrm{DT}_{\mathrm{fc}}$-transducers in [91]. The $\mathrm{DMT}_{\mathrm{fc}}$-transducer M is *repetition-free* if all its state sequences in $\mathrm{sts}(M)$ are repetition-free.

Proposition 74 *For every DMT_{fc}-transducer M there is a repetition-free DMT_{fc}-transducer M' that realizes the same tree transduction as M. Moreover, if M is a DT_{fc}-transducer, then so is M'.*

Proof It is proved in [26, Lemma 6.10] that there is a single-use restricted $\mathrm{DMT}_{\mathrm{sp}}$-transducer M' that realizes the same tree transduction as M. It is in fact proved for macro tree transducers with regular look-ahead, but the construction preserves the absence of look-ahead. Moreover, in the proof of [26, Theorem 6.12] it is shown that single-use restricted $\mathrm{DMT}_{\mathrm{sp}}$-transducers are finite-copying and repetition-free. The construction in [26, Lemma 6.10] preserves $\mathrm{DT}_{\mathrm{fc}}$-transducers, but for them the result was already proved in [91, Lemma 5.3]. ∎

Lemma 75 $\mathrm{DMT}_{\mathrm{fc}}(\mathrm{RT}) \subseteq \mathrm{MCFT}$ *and* $\mathrm{DT}_{\mathrm{fc}}(\mathrm{RT}) \subseteq \mathrm{MRT}$.

Proof Let $M = (Q, \Omega, \Sigma, q_0, R_M)$ be a $\mathrm{DMT}_{\mathrm{fc}}$-transducer, of which we assume, by Proposition 74, that it is repetition-free. Moreover, let $G = (N, \Omega, S, R)$ be an RTG. We can assume that in each of its rules $C \to \omega(C_1, \ldots, C_k)$, with $C, C_1, \ldots, C_k \in N$ and $\omega \in \Omega^{(k)}$, the sequence $(C_1, \ldots, C_k)$ is repetition-free (cf. Section 2.1). We will construct an MCFTG $G' = (N', \mathcal{N}', \Sigma, S', R')$ such that $L(G') = M(L(G))$. The MCFTG G' will simulate both M and G. Thus, we define $N' = Q \times N$, where every $\langle q, C\rangle \in N'$ has the same rank as q, and $S' = \langle q_0, S\rangle$. For every nonempty state sequence $\bar{q} = (q_1, \ldots, q_n) \in Q^+$ and nonterminal $C \in N$, we abbreviate the sequence $(\langle q_1, C\rangle, \ldots, \langle q_n, C\rangle) \in (N')^+$ by $\bar{q} \otimes C$. Then we define $\mathcal{N}' = \{\bar{q} \otimes C \mid \bar{q} \in \mathrm{sts}(M) \setminus \{\varepsilon\},\, C \in N\}$, so in other words, the big nonterminals of G' are of the form $(\langle q_1, C\rangle, \ldots, \langle q_n, C\rangle)$, where $(q_1, \ldots, q_n)$ is a nonempty state sequence of M, and C is a nonterminal of G. It remains to define the rules of G'. Let $\rho = C \to \omega(C_1, \ldots, C_k)$ be a rule of G, and let $\bar{q} = (q_1, \ldots, q_n)$ be a nonempty state sequence of M. Then R' contains the rule

$$\rho_{\bar{q}} = (\langle q_1, C\rangle, \ldots, \langle q_n, C\rangle) \to ((u_1, \ldots, u_n), \mathcal{L})$$

with left-hand side $\bar{q} \otimes C$, where $u_j = \mathrm{rhs}_M(q_j, \omega)[\langle q, y_i\rangle \leftarrow \mathrm{in}(\langle q, C_i\rangle) \mid q \in Q,\, 1 \le i \le k]$ for every $j \in [n]$ and $\mathcal{L} = \{\mathrm{sts}_{\omega,i}(\bar{q}) \otimes C_i \mid i \in [k]\} \cap \mathcal{N}'$. Note that $(u_1, \ldots, u_n)$ is uniquely N'-labeled because $(C_1, \ldots, C_k)$ is repetition-free and every state sequence $\mathrm{sts}_{\omega,i}(\bar{q})$ is repetition-free. The correctness of G' is a direct consequence of the following claim.

Claim. For every nonempty state sequence $\bar{q} \in \mathrm{sts}(M) \setminus \{\varepsilon\}$, nonterminal $C \in N$, and forest $t \in P_\Sigma(X)^+$ we have $t \in L(G', \bar{q} \otimes C)$ if and only if there exists $s \in L(G, C)$ such that $M_{\bar{q}}(s) = t$.[45]

[44] If $\Sigma^{(k)} \neq \emptyset$ for some $k \ge 2$, then $P_\Sigma(X_m) \neq \emptyset$ for all m (recall that $\Sigma^{(0)} \neq \emptyset$). If $\Sigma = \Sigma^{(0)} \cup \Sigma^{(1)}$, then $N = N^{(0)} \cup N^{(1)}$ (because G is reduced) and we only need $t_0 \in \Sigma^{(0)}$ and $t_1 = x_1$.

[45] In other words, $L(G', \bar{q} \otimes C\rangle) = M_{\bar{q}}(L(G, C))$. Recall the definition of $M_{\bar{q}}(s)$ just before Lemma 73.

Proof of sufficiency. We have to show that $M_{\bar{q}}(s) \in L(G', \bar{q} \otimes C)$ for every $s \in L(G, C)$. The proof is by induction on the structure of s. Let $s = \omega(s_1, \ldots, s_k)$. Then there is a rule $\rho = C \to \omega(C_1, \ldots, C_k)$ of G such that $s_i \in L(G, C_i)$ for every $i \in [k]$. Let $\bar{q}_i = \mathrm{sts}_{\omega,i}(\bar{q})$ for every $i \in [k]$. By the induction hypotheses, $M_{\bar{q}_i}(s_i) \in L(G', \bar{q}_i \otimes C_i)$ provided that $\bar{q}_i \neq \varepsilon$. Let $\rho_{\bar{q}}$ be the rule in R' as defined above. Then the least fixed point semantics of G' implies that $L(G', \bar{q} \otimes C)$ contains the forest

$$(u_1, \ldots, u_n)[\bar{q}_i \otimes C_i \leftarrow M_{\bar{q}_i}(s_i) \mid i \in [k],\, \bar{q}_i \neq \varepsilon] \ ,$$

which equals $M_{\bar{q}}(s)$.

Proof of necessity. The proof is similar and proceeds by induction on the structure of a derivation tree $d \in L(G'_{\mathrm{der}}, \bar{q} \otimes C)$ with $\mathrm{val}(d) = t$. Let $d = \rho_{\bar{q}}(d_1, \ldots, d_k)$. Then

$$t = (u_1, \ldots, u_n)[\bar{q}_i \otimes C_i \leftarrow \mathrm{val}(d_i) \mid i \in [k],\, \bar{q}_i \neq \varepsilon] \ .$$

By the induction hypotheses, there exist trees $s_i \in L(G, C_i)$ such that $M_{\bar{q}_i}(s_i) = \mathrm{val}(d_i)$ for every $i \in [k]$ with $\bar{q}_i \neq \varepsilon$. Since we assume that G is reduced, there also exist trees $s_i \in L(G, C_i)$ for every $i \in [k]$ with $\bar{q}_i = \varepsilon$. Consequently, $s \in L(G, C)$ and $M_{\bar{q}}(s) = t$ for $s = \omega(s_1, \ldots, s_k)$. ∎

From Lemmas 73 and 75 we obtain our characterization result, of which the second part was proved in [79, Proposition 4.8].

Theorem 76 $\mathrm{MCFT} = \mathrm{DMT}_{\mathrm{fc}}(\mathrm{RT})$ *and* $\mathrm{MRT} = \mathrm{DT}_{\mathrm{fc}}(\mathrm{RT})$.

We observe that the multiplicity of the MCFTG corresponds to the "copying number" of the corresponding $\mathrm{DMT}_{\mathrm{fc}}$-transducer. For every $m \in \mathbb{N}$, let m-MCFT be the class of tree languages generated by MCFTGs G with $\mu(G) \leq m$, and let $\mathrm{DMT}_{\mathrm{fc}(m)}$ be the class of transductions realized by m-copying $\mathrm{DMT}_{\mathrm{fc}}$-transducers, and similarly for subclasses of these grammars and transducers. Then, checking the proofs above, we obtain that $m\text{-MCFT} = \mathrm{DMT}_{\mathrm{fc}(m)}(\mathrm{RT})$ and $m\text{-MRT} = \mathrm{DT}_{\mathrm{fc}(m)}(\mathrm{RT})$ for every $m \in \mathbb{N}$. For the preservation of the m-copying property in Proposition 74 we additionally need to inspect the proof of [26, Lemma 6.10]). For $m = 1$ we obtain that $\mathrm{CFT}_{\mathrm{sp}} = \mathrm{DMT}_{\mathrm{fc}(1)}(\mathrm{RT})$. A $\mathrm{DMT}_{\mathrm{sp}}$-transducer is *simple* (in short, $\mathrm{DMT}_{\mathrm{si,sp}}$-transducer) if it is also simple (i.e., linear and nondeleting) in the input variables. Clearly, $\mathrm{DMT}_{\mathrm{si,sp}}$-transducers are 1-copying. Checking again the proofs above, it is easy to see that $\mathrm{CFT}_{\mathrm{sp}} = \mathrm{DMT}_{\mathrm{si,sp}}(\mathrm{RT})$.[46]

In the remainder of this section we discuss the consequences of the characterization result in Theorem 76. One immediate consequence is that MCFT is closed under intersection with regular tree languages: If M is a $\mathrm{DMT}_{\mathrm{fc}}$-transducer and R_1 and R_2 are in RT, then $M(R_1) \cap R_2 = M(R_1 \cap M^{-1}(R_2))$. Moreover, $M^{-1}(R_2)$ is in RT by [34, Theorem 7.4] and so $R_1 \cap M^{-1}(R_2)$ is in RT.

From Theorem 76 and Corollary 70 we obtain two known results. First, $\mathrm{MCF} = \mathrm{yDT}_{\mathrm{fc}}(\mathrm{RT})$. Since it is easy to check from the proof of Corollary 70 that $m\text{-MCF} = \mathrm{y}(m\text{-MRT})$, we even obtain that $m\text{-MCF} = \mathrm{yDT}_{\mathrm{fc}(m)}(\mathrm{RT})$ for every $m \in \mathbb{N}$. It was, in fact, proved in [94] that m-MCF equals the class of output languages of deterministic tree-walking transducers with "crossing number" m, which equals $\mathrm{yDT}_{\mathrm{fc}(m)}(\mathrm{RT})$ by [30, Corollary 4.11]. Second, $\mathrm{yDMT}_{\mathrm{fc}}(\mathrm{RT}) = \mathrm{yDT}_{\mathrm{fc}}(\mathrm{RT})$, which was proved in [26, Corollary 7.10]. Vice versa, this equality and Theorem 76 imply that $\mathrm{yMCFT} = \mathrm{yMRT}$ (Corollary 70). We also observe that this equality is a restricted version of $\mathrm{yDMT}_{\mathrm{sp}}(\mathrm{RT}) = \mathrm{yDT}(\mathrm{RT})$, which was proved in [28, Theorem 15] (cf. the last sentence before Theorem 67) and will follow from the results in Section 10.

More interestingly, Theorem 76 implies three other characterizations of MCFT and MCF (of which those of MCF are already known). First, they can be characterized in monadic second-order logic (MSO). Let DMSOT be the class of deterministic (or parameterless) MSO-definable tree transductions (see, e.g., [12, Chapter 8]), and let DMSOTS be the analogous class of tree-to-string transductions. Since regular look-ahead can be simulated by a relabeling of the input tree, it follows from [26, Theorem 7.1] that $\mathrm{DMSOT}(\mathrm{RT}) = \mathrm{DMT}_{\mathrm{fc}}(\mathrm{RT})$ and from [26, Theorem 7.7] that $\mathrm{DMSOTS}(\mathrm{RT}) = \mathrm{yDT}_{\mathrm{fc}}(\mathrm{RT})$.

Corollary 77 $\mathrm{MCFT} = \mathrm{DMSOT}(\mathrm{RT})$ *and* $\mathrm{MCF} = \mathrm{DMSOTS}(\mathrm{RT})$.

[46] The only small technical problem is the deletion of all input variables in the dummy rules in the proof of Lemma 73. This can be easily remedied by introducing an additional state q of rank 1, changing the dummy rules into $\langle C, \rho(y_1, \ldots, y_k)\rangle(x_1, \ldots, x_m) \to \langle q, y_1\rangle(\cdots(\langle q, y_k\rangle(t_m))\cdots)$ and adding additionally all dummy rules of the form $\langle q, \rho(y_1, \ldots, y_k)\rangle(x_1) \to \langle q, y_1\rangle(\cdots(\langle q, y_k\rangle(x_1))\cdots)$.

Since MSO-definable transductions are closed under composition [12, Theorem 7.14], this implies that MCFT is closed under DMSOT-transductions, and hence under $\mathrm{DMT}_{\mathrm{fc}}$-transductions even when they are equipped with regular look-ahead by [26, Theorem 7.1]. Similarly, MCF is closed under deterministic MSO-definable string transductions, which are the transductions realized by two-way deterministic finite-state transducers [23]. In particular, it follows from Lemma 73 that MCFT is closed under control, in the following sense. Let G be an MCFTG and let C be a ("control") tree language in MCFT. Then $\mathrm{val}(L(G_{\mathrm{der}}) \cap C)$ is in MCFT. Intuitively, the derivation trees of the grammar G are restricted to be an element of C; in that way C "controls" the derivation trees (and hence the derivations) of G.

Second, MCFT and MCF can be characterized in terms of context-free graph grammars. It is known that DMSOT(RT) equals the class of tree languages that can be generated by (either hyperedge-replacement or vertex-replacement) context-free graph grammars (see, e.g., [19, Section 6] or the introduction of [12, Section 8.9]). Similarly, DMSOTS(RT) is the class of string languages generated by such grammars. These facts were also used to obtain [26, Corollaries 7.3 and 7.8].

Corollary 78 MCFT *(resp.* MCF*) is the class of tree languages (resp. string languages) generated by context-free graph grammars.*

Remark 79 For completeness' sake we show here how easy it is to simulate an MCFTG by a context-free graph grammar, in particular a hyperedge-replacement grammar (HRG). We assume the reader to be familiar with HRGs (see, e.g., [5, 14, 19]). Let us first recall how trees and forests can be represented as hypergraphs. Let Ω be a ranked alphabet. A forest $t = (t_1, \ldots, t_n) \in P_\Omega(X)^+$ is represented by the hypergraph $\mathrm{gr}(t)$ that has the set of nodes $\mathrm{pos}(t)$ and the set of hyperedges $\{e_p \mid p \in \mathrm{pos}_\Omega(t)\}$ such that e_p has label $t(p)$ and sequence of incident nodes $\mathrm{inc}_t(p) = (p1, \ldots, pk, p)$ where $k = \mathrm{rk}(t(p))$. Moreover, $\mathrm{gr}(t)$ has the sequence of external nodes $\mathrm{ext}(t) = \mathrm{ext}(t_1) \cdots \mathrm{ext}(t_n)$ such that $\mathrm{ext}(t_j) = (p_{j,1}, \ldots, p_{j,k_j}, \#^{j-1})$ where $t_j(p_{j,\ell}) = x_\ell \in X$ for every $j \in [n]$ and $\ell \in [k_j]$ with $k_j = \mathrm{rk}(t_j)$. We say that an HRG is *tree generating* (or, generates a tree language) if its terminal alphabet is a ranked alphabet Σ and the generated hypergraph language is a subset of $\{\mathrm{gr}(t) \mid t \in T_\Sigma\}$.

Now let $G = (N, \mathcal{N}, \Sigma, S, R)$ be an MCFTG. We construct an HRG G' that has the set of nonterminals $\mathcal{N}$, with initial nonterminal S, and the set of terminals Σ. Let $A \to (u, \mathcal{L})$ be a rule in R. Then G' has the rule $A \to \mathrm{gr}(u, \mathcal{L})$, where $\mathrm{gr}(u, \mathcal{L})$ is the hypergraph obtained from $\mathrm{gr}(u)$ as follows. For every $B = (u(p_1), \ldots, u(p_m)) \in \mathcal{L}$ with $p_1, \ldots, p_m \in \mathrm{pos}_N(u)$, remove the hyperedges $e_{p_1}, \ldots, e_{p_m}$ and replace them by one new hyperedge e_B that has label B and sequence of incident nodes $\mathrm{inc}_u(p_1) \cdots \mathrm{inc}_u(p_m)$.[47] Intuitively, the hyperedge e_B explicitly links the occurrences in u of the nonterminals $u(p_1), \ldots, u(p_m)$ of the link B. Now let $t_B \in P_\Sigma(X)^+$ be a forest with $\mathrm{rk}(t_B) = \mathrm{rk}(B)$, for every $B \in \mathcal{L}$. Then it is straightforward to check that $\mathrm{gr}(u[B \leftarrow t_B \mid B \in \mathcal{L}])$ is equal to the result of simultaneously substituting $\mathrm{gr}(t_B)$ for the hyperedge e_B in $\mathrm{gr}(u, \mathcal{L})$ for every $B \in \mathcal{L}$. Thus, using the least fixed point semantics of the HRG G' (see [14, Theorem 2.4.2]), we obtain that $L(G') = \{\mathrm{gr}(t) \mid t \in L(G)\}$. It can also easily be checked that the derivations of G, as defined in Section 3.3, can be simulated by the derivations of G': for every $t \in T_{(N \times \mathbb{N}^*) \cup \Sigma}$ and $n \in \mathbb{N}_0$, if $S \otimes \varepsilon \Rightarrow_G^n t$ then $\mathrm{gr}(S) \Rightarrow_{G'}^n \mathrm{gr}(t, \mathcal{L})$, where $\mathrm{gr}(t, \mathcal{L})$ is defined similarly to $\mathrm{gr}(u, \mathcal{L})$ above using the set $\mathcal{L} \subseteq \mathcal{N} \otimes \mathbb{N}^*$ mentioned at the end of Section 3.3. Moreover, these are all possible derivations in G'. Intuitively, the role of the link identifiers in the derivation $S \otimes \varepsilon \Rightarrow_G^n t$ is taken over by explicit hyperedges.

We say that an HRG is in *tree generating normal form* if it can be obtained from an MCFTG in the way described above, eventually followed by a renaming of its nonterminals and an identification of nonterminals that are aliases.[48] Then the above, together with Lemma 40 and Corollary 78, proves that every tree generating HRG has an equivalent HRG in tree generating normal form (see [27, Theorem 7]). We finally note that there is a similar easy construction showing that every string language in MCF can be generated by an HRG (see [19, Theorem 6.4]).

As an example of the above construction we consider the MCFTG G of Example 7. The rules of the HRG G' are shown in Figure 12 (without the rules for the alias B' of B) and the derivation of G' corresponding to the one of G in Figure 5 is shown in Figure 13. By definition, G' is in tree generating normal form. Note that the sequence of external nodes of the right-hand side of rule ρ_4 (and of rule

[47] Every terminal or nonterminal symbol α of an HRG should have a "rank". For every hyperedge e with label α the "rank" of α should be equal to the number of nodes that are incident with e. Moreover, for every rule $A \to g$ the "rank" of A should be equal to the number of external nodes of the hypergraph g. In the grammar G', every terminal $\sigma \in \Sigma$ has "rank" $\mathrm{rk}(\sigma) + 1$ and every nonterminal $A = (A_1, \ldots, A_n)$ has "rank" $\sum_{i=1}^n (\mathrm{rk}(A_i) + 1)$.

[48] It can be checked that this is equivalent to [27, Definition 6], provided that the MCFTG is assumed to be nonerasing.

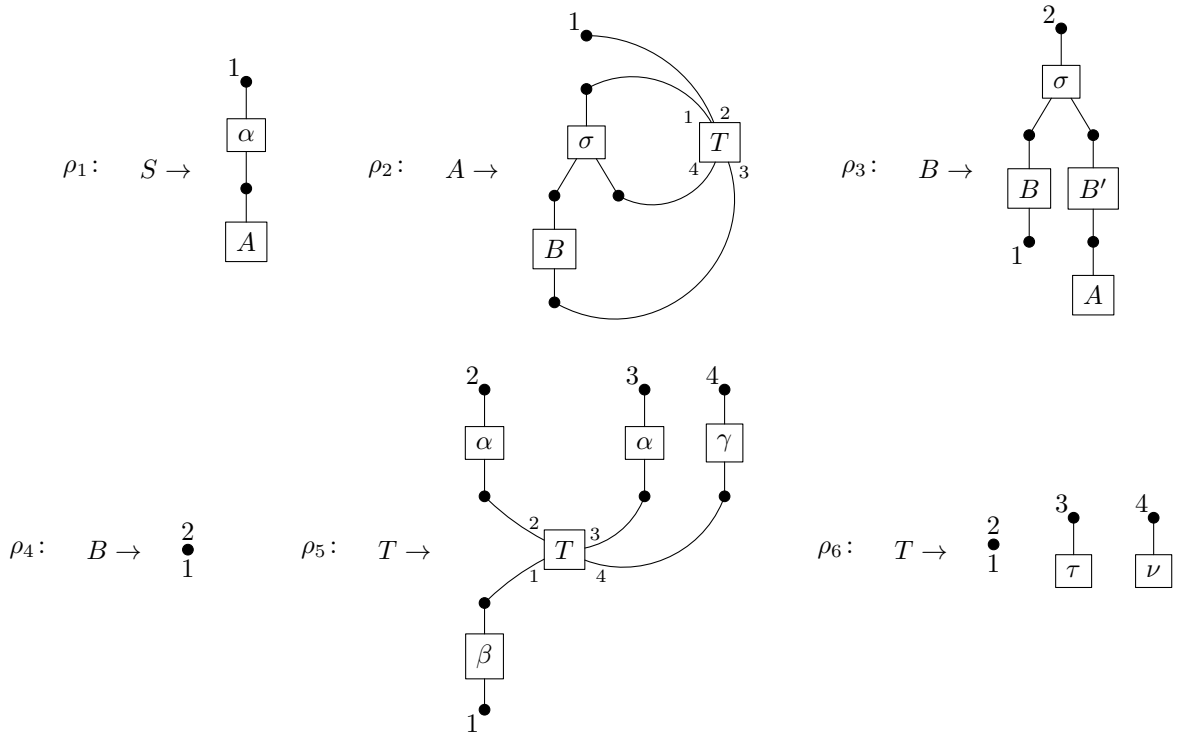


Figure 12: Rules of the HRG G' corresponding to the MCFTG G of Example 7 (without the rules for B'). Hypergraphs are drawn as in [14, 19]. A hyperedge e is drawn as a box containing the label of e. A line with label i connects e with its i-th incident node. If the label of e is in $N \cup \Sigma$ with rank k, then the labels of the incidence lines are dropped; by convention, the first k incident nodes of e are below the box, from left to right, and the last incident node is above the box. The j-th external node of the hypergraph is labeled j.

ρ_6) of G' is not repetition-free, which allows G' to erase hyperedges (or "parts" of hyperedges). For a nonerasing MCFTG G the above construction results in an HRG G' for which all sequences of external nodes (and all sequences of incident nodes) are repetition-free. Thus by Lemma 40, this requirement can be added to the tree generating normal form. □

Third, MCFT and MCF can be characterized in terms of second-order abstract categorial grammars. It is shown in [57] that such grammars have the same tree and string generating power as hyperedge-replacement context-free graph grammars, which was already known for strings from earlier results as discussed in [57].

Corollary 80 MCFT *(resp.* MCF*) is the class of tree languages (resp. string languages) generated by second-order abstract categorial grammars.*

Proof Let TR(2AC) denote the class of tree languages generated by second-order abstract categorial grammars (in short, 2ACGs). It is shown in [57] that TR(2AC) is included in the class of tree languages generated by hyperedge-replacement context-free graph grammars (HRG), and hence TR(2AC) $\subseteq$ MCFT by Corollary 78. In the other direction, it is shown in [57] by a simple construction that every tree language generated by an HRG in tree generating normal form (as in [27, Definition 6] or equivalently in Remark 79) is in TR(2AC). Note that together with the construction in Remark 79 this also shows that there is a simple construction to transform every MCFTG into an equivalent 2ACG. ■

We finally observe (cf. the paragraph after [26, Corollary 7.10]) that MRT is properly included in MCFT. The tree language $\{a^n b^n \triangleright \mid n \in \mathbb{N}_0\}$ over $\Sigma = \{a^{(1)}, b^{(1)}, \triangleright^{(0)}\}$ is in MCFT and even in $\mathrm{CFT}_{\mathrm{sp}}$, but not in $\mathrm{DT}_{\mathrm{fc}}(\mathrm{RT})$ because all tree languages over Σ in this class are regular [81, Theorem 4]. Also $\mathrm{CFT}_{\mathrm{sp}}$ is properly included in MCFT since it is shown in [20, Section 5] that the tree language $L(G)$, where G is the MRTG of Example 6, is not in $\mathrm{CFT}_{\mathrm{sp}}$. Thus, MRT and $\mathrm{CFT}_{\mathrm{sp}}$ are incomparable subclasses of MCFT.

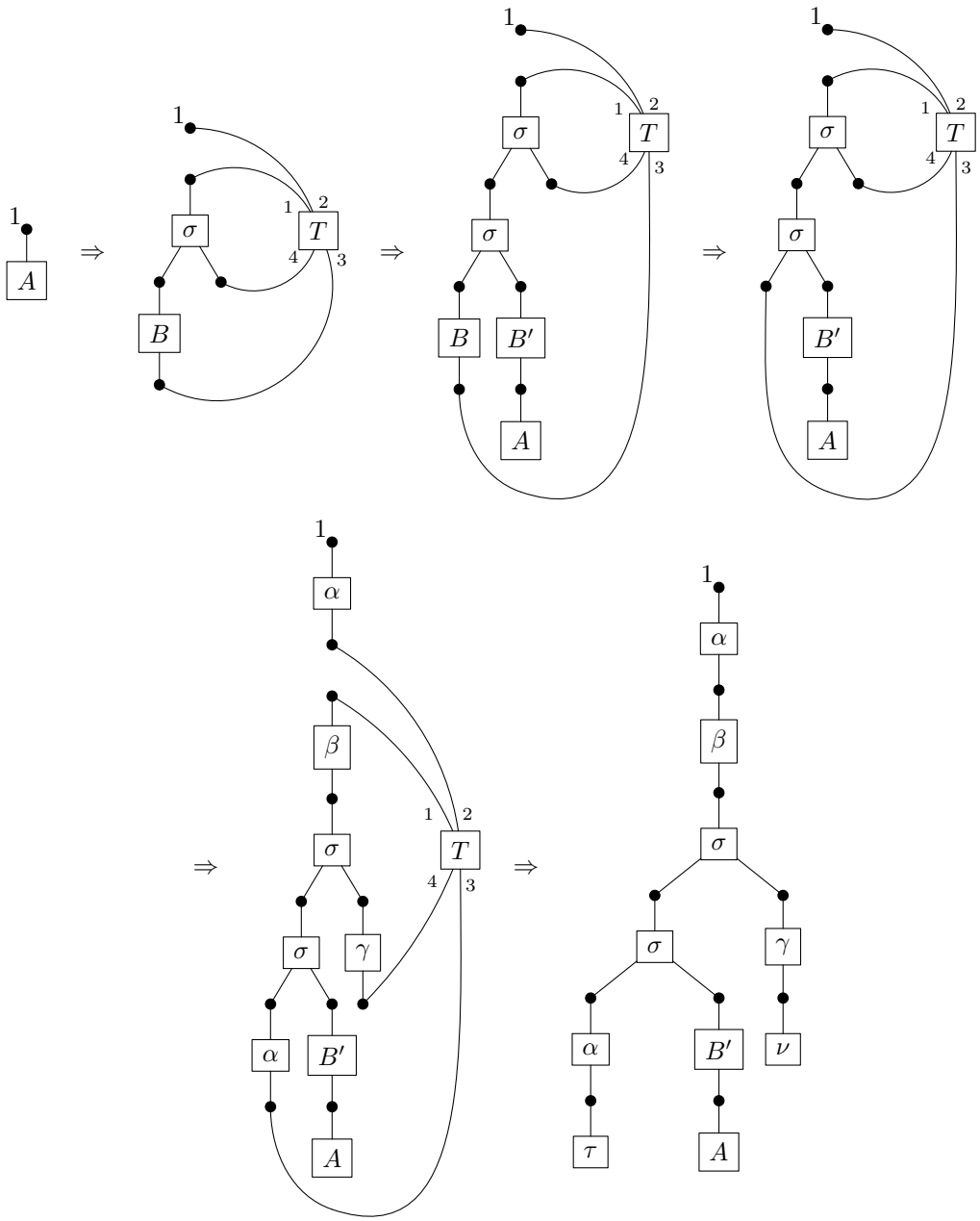


Figure 13: Derivation of the HRG of Figure 12 corresponding to the MCFTG derivation of Figure 5.

9. Translation

As observed in [79] for MRTGs, MCFTGs are not only a natural generation device but also a natural translation device. In general, we can also use an MCFTG G to define a forest language (i.e., an n-ary relation on T_Σ) by considering $L(G, A)$ for a big nonterminal $A = (A_1, \ldots, A_n)$ with $\mathrm{rk}(A_i) = 0$ for every $i \in [n]$. In particular, for the case $n = 2$, the MCFTG can be used as a synchronous translation device, which we will call an MCFT-transducer. After defining MCFT-transducers we present two results analogous to those in [73] (see also [69]). Namely, we prove a characterization of the corresponding MCFT-transductions by macro tree transducers, similar to the one for MCFT tree languages in Theorem 76 (in the previous section), and we present a solution to the parsing and translation problem for MCFT-transducers, similar to the one for MCFTGs in Theorem 72 (in Section 7).

A *multiple context-free tree transducer* (in short, MCFT-transducer) is a system $G = (N, \mathcal{N}, \Sigma, S, R)$, where N, $\mathcal{N}$, Σ, and R are as in Definition 5 and $S = (S_1, S_2) \in \mathcal{N}$ is the *initial big nonterminal* with $S_1, S_2 \in N^{(0)}$. We require (without loss of generality) that G is start-separated; i.e., that S_1 and S_2 do not occur in the right-hand sides of rules. Moreover, we require that N is partitioned into two subsets N_1 and N_2 of input nonterminals and output nonterminals, respectively, such that

(1) $S_1 \in N_1$ and $S_2 \in N_2$, and
(2) for every rule $(A_1, \ldots, A_n) \to ((u_1, \ldots, u_n), \mathcal{L})$ in R, every $j \in [n]$, and every $i \in [2]$ we have $\mathrm{occ}_N(u_j) \subseteq N_i$ if $A_j \in N_i$.

Intuitively this requirement means that the nonterminals in N_1 generate the input tree, and those in N_2 generate the output tree. For every $A \in \mathcal{N}$, the forest language $L(G, A)$ is defined as for MCFTGs, and the *tree transduction realized by* G is the binary relation $\tau(G) = L(G) = L(G, S) \subseteq T_\Sigma \times T_\Sigma$. We also define G_{der} as for MCFTGs. Thus, the initial nonterminal of G_{der} is $S = (S_1, S_2)$. Consequently, $\tau(G) = \mathrm{val}(L(G_{\mathrm{der}}))$ by Theorem 9. Note that the input and output alphabet of G are the same ranked alphabet Σ. This is a slight restriction that could be solved by allowing symbols in a ranked alphabet to have more than one rank. The latter feature is easy to implement, but technically rather tiresome. We will say that G is an MCFT-transducer *over* Σ and that $\tau(G)$ is an MCFT-transduction over Σ. The synchronous context-free tree grammar of [73] is the special case of the MCFT-transducer in which $\mathcal{N} \subseteq N_1 \times N_2$.

Our characterization of MCFT-transductions by macro tree transducers uses a generalization of the notion of bimorphism. Bimorphisms are a classical symmetrical way to characterize classes of string and tree transductions (see, e.g., [2, 68, 75]). Let $\mathcal{X}$ be a class of tree transductions. For a finite ranked alphabet Σ, we define an $\mathcal{X}$-*bimorphism* over Σ to be a transduction $\tau \subseteq T_\Sigma \times T_\Sigma$ such that $\tau = \{(M_1(s), M_2(s)) \mid s \in L\}$, where L is a regular tree language over a finite ranked alphabet Ω and M_1 and M_2 are $\mathcal{X}$-transductions with input alphabet Ω and output alphabet Σ. In the classical case $\mathcal{X}$ is a class of tree homomorphisms (or string homomorphisms in the similar case of strings); cf. the proof of Proposition 14. In the present case we take $\mathcal{X} = \mathrm{DMT}_{\mathrm{fc}}$ and we show that MCFT-transductions are as expressive as $\mathrm{DMT}_{\mathrm{fc}}$-bimorphisms. Clearly, if M_1 and M_2 are $\mathrm{DMT}_{\mathrm{fc}}$-transductions, then the domain $L_1 = \{M_1(s) \mid s \in L\}$ and the range $L_2 = \{M_2(s) \mid s \in L\}$ of the $\mathrm{DMT}_{\mathrm{fc}}$-bimorphism τ are tree languages in MCFT by Theorem 76 and τ can be viewed as translating L_1 into L_2. The inverse of τ is the $\mathrm{DMT}_{\mathrm{fc}}$-bimorphism $\tau^{-1} = \{(M_2(s), M_1(s)) \mid s \in L\}$ which translates L_2 into L_1. Thus, $\mathrm{DMT}_{\mathrm{fc}}$-bimorphisms are a natural symmetrical model for the translation of MCFT languages. To prove the characterization we need a few more definitions.

We first modify the notion of $\mathrm{DMT}_{\mathrm{fc}}$-transducer in such a way that it translates trees into forests of length 2. We define a $\mathrm{DMT}_{\mathrm{sp},2}$-transducer to be a system $M = (Q, \Omega, \Sigma, q_0, R)$, where the only difference to a $\mathrm{DMT}_{\mathrm{sp}}$-transducer is that $q_0 = q_1 q_2$ is the *initial state sequence* with $q_1, q_2 \in Q^{(0)}$. For $s \in T_\Omega$ and $q \in Q$, the tree $M_q(s)$ is defined as for $\mathrm{DMT}_{\mathrm{sp}}$-transducers, and $M(s) = M_{q_0}(s)$ which equals $(M_{q_1}(s), M_{q_2}(s))$ by the definition before Lemma 73. The tree transduction realized by M is defined as for $\mathrm{DMT}_{\mathrm{sp}}$-transducers; i.e., it is the total function $M = \{(s, M(s)) \mid s \in T_\Omega\}$ from T_Ω to $T_\Sigma \times T_\Sigma$. The state sequences of a $\mathrm{DMT}_{\mathrm{sp},2}$-transducer are defined in the same way as for $\mathrm{DMT}_{\mathrm{sp}}$-transducers with $\mathrm{sts}(s, \varepsilon) = q_0$, and finite-copying $\mathrm{DMT}_{\mathrm{sp},2}$-transducers are called $\mathrm{DMT}_{\mathrm{fc},2}$-transducers.

We now define the product of two $\mathrm{DMT}_{\mathrm{sp}}$-transducers M_1 and M_2 with the same input and output alphabets to be the $\mathrm{DMT}_{\mathrm{sp},2}$-transducer $M_1 \otimes M_2$ given as follows. Let $M_i = (Q_i, \Omega, \Sigma, q_i, R_i)$ with $i \in [2]$, where we assume that Q_1 and Q_2 are disjoint. Then $M_1 \otimes M_2 = (Q_1 \cup Q_2, \Omega, \Sigma, q_1 q_2, R_1 \cup R_2)$. It should be clear that for every $s \in T_\Omega$ we have $(M_1 \otimes M_2)(s) = (M_1(s), M_2(s))$. It should also be clear that for every $p \in \mathrm{pos}(s)$ the state sequence of $M_1 \otimes M_2$ at p is the concatenation of the state sequences of M_1 and M_2 at p. This implies that $M_1 \otimes M_2$ is finite-copying (repetition-free) if and only

if M_1 and M_2 are both finite-copying (repetition-free). Vice versa, for every $\mathrm{DMT}_{\mathrm{sp},2}$-transducer M there are $\mathrm{DMT}_{\mathrm{sp}}$-transducers M_1 and M_2 such that M and $M_1 \otimes M_2$ realize the same tree transduction. Clearly, if $M = (Q, \Omega, \Sigma, q_1 q_2, R)$, then we can take $M_1 = (Q, \Omega, \Sigma, q_1, R)$ and $M_2 = (Q', \Omega, \Sigma, q_2', R')$, where the primes indicate a consistent renaming of the states of M such that $Q \cap Q' = \emptyset$. The transducer $M_1 \otimes M_2$ is obviously equivalent to M and it is finite-copying if M is. Thus we have shown that $\mathrm{DMT}_{\mathrm{fc},2} = \{M_1 \otimes M_2 \mid M_1, M_2 \in \mathrm{DMT}_{\mathrm{fc}}\}$. Note that it follows from Proposition 74 that this proposition also holds for $\mathrm{DMT}_{\mathrm{fc},2}$-transducers.

With these preparations, we can now prove our characterization of MCFT-transductions as bimorphisms of macro tree transductions.

Theorem 81 *Let Σ be a finite ranked alphabet. A transduction $\tau \subseteq T_\Sigma \times T_\Sigma$ is an* MCFT*-transduction over Σ if and only if it is a* $\mathrm{DMT}_{\mathrm{fc}}$*-bimorphism over Σ.*

Proof Exactly the same proofs as those of Lemmas 73 and 75 show that the class of MCFT-transductions equals the class $\mathrm{DMT}_{\mathrm{fc},2}(\mathrm{RT})$. The latter class coincides with the class of $\mathrm{DMT}_{\mathrm{fc}}$-bimorphisms because if $M = M_1 \otimes M_2$, where M is a $\mathrm{DMT}_{\mathrm{fc},2}$-transducer and M_1 and M_2 are $\mathrm{DMT}_{\mathrm{fc}}$-transducers, then $M(L) = \{(M_1(s), M_2(s)) \mid s \in L\}$ for every regular tree language $L \in \mathrm{RT}$. Note that if M_1 and M_2 have the disjoint sets of states Q_1 and Q_2, then the set N' of nonterminals of the MCFT-transducer G' constructed in the proof of Lemma 75 is partitioned into the set $Q_1 \times N$ of input nonterminals and the set $Q_2 \times N$ of output nonterminals, where N is the set of nonterminals of the given RTG. ∎

We note that we can define MRT-transducers and $\mathrm{DT}_{\mathrm{fc}}$-bimorphisms in the obvious way, and prove as a special case of Theorem 81 that the MRT-transductions (which are the binary rational tree translation of [79]) coincide with the $\mathrm{DT}_{\mathrm{fc}}$-bimorphisms. In [69] the MRT-transducers are called synchronous forest substitution grammars, and it is shown in [69, Theorem 3] that the MRT-transductions are the ld-MBOT-bimorphisms, where ld-MBOT is the class of transductions realized by linear deterministic multi bottom-up tree transducers [24].[49] By [24, Theorem 18] and [26, Theorems 5.10 and 7.4], this is essentially the same result. We also note that we can define $\mathrm{DMT}^{\mathrm{R}}_{\mathrm{fc}}$-bimorphisms in the obvious way, where $\mathrm{DMT}^{\mathrm{R}}_{\mathrm{fc}}$-transducers are defined just as $\mathrm{DMT}_{\mathrm{fc}}$-transducers, but with regular look-ahead as in the definition of $\mathrm{LDT}^{\mathrm{R}}$-transducer. Since regular look-ahead can be simulated by a relabeling of the input tree, the $\mathrm{DMT}^{\mathrm{R}}_{\mathrm{fc}}$-bimorphisms are the same as the $\mathrm{DMT}_{\mathrm{fc}}$-bimorphisms. In other words, the addition of regular look-ahead does not increase the power of these bimorphisms. Moreover, the class of $\mathrm{DMT}^{\mathrm{R}}_{\mathrm{fc}}$-transductions coincides with the class DMSOT of deterministic MSO-definable tree transductions (cf. Corollary 77 and the preceding paragraph). Thus, the MCFT-transductions are the DMSOT-bimorphisms. The notion of DMSOT-bimorphism is quite natural as it is a transduction of the form $\{(M_1(s), M_2(s)) \mid s \in L\}$, where L is an MSO-definable tree language and M_1 and M_2 are deterministic MSO-definable tree transductions. Even if we assume that DMSOT transductions need not be total (cf. footnote 43), it follows that the class of MCFT-transductions properly includes the class DMSOT. To see this note that, in particular, every DMSOT transduction and its inverse are DMSOT-bimorphisms. Thus, since DMSOT is not closed under inverse (see [12, Remark 7.23]), DMSOT is properly included in the class of DMSOT-bimorphisms.

We now turn to the parsing and translation problem for MCFT-transducers, generalizing the parsing algorithm for MCFTGs in Theorem 72. Let G be an MCFT-transducer over Σ, and let $\Delta \subseteq \Sigma^{(0)} \setminus \{e\}$ be a set of lexical symbols. We can view G as translating input strings into output strings, thereby realizing the string transduction $\{(\mathrm{yd}_\Delta(t_1), \mathrm{yd}_\Delta(t_2)) \mid (t_1, t_2) \in \tau(G)\}$. In such a case the parsing and translation problem for G amounts to finding the syntactic trees for a given string over Δ and finding its possible translations together with their syntactic trees. In the next result we show that this can be done in polynomial time. For its proof we need some more terminology. It is straightforward to prove the analogue of Lemma 21 for MCFT-transducers, which shows that MCFT-transductions are closed under tree homomorphisms. For a given MCFT-transducer G and tree homomorphism h, the MCFT-transducer G_h has the same initial big nonterminal as G. Moreover, the lemma implies that $\tau(G_h) = \{(\hat{h}(t_1), \hat{h}(t_2)) \mid (t_1, t_2) \in \tau(G)\}$. As before, (G, h) is said to be a cover of G_h if h is a projection. An MCFT-transducer G over Σ is *i/o-disjoint* if Σ is partitioned into subsets Σ_1 and Σ_2 of input and output terminal symbols, and

(2′) for every rule $(A_1, \ldots, A_n) \to ((u_1, \ldots, u_n), \mathcal{L})$ in R, every $j \in [n]$, and every $i \in [2]$ we have $\mathrm{occ}_{N \cup \Sigma}(u_j) \subseteq N_i \cup \Sigma_i$ if $A_j \in N_i$.

[49] The restriction to *linear* d-MBOT is implicit in [69].

This guarantees that $\tau(G) \subseteq T_{\Sigma_1} \times T_{\Sigma_2}$. It should be clear that every MCFT-transducer G over Σ has a cover (G^{u}, h) such that G^{u} is i/o-disjoint, the terminal alphabet $\Sigma \cup \Sigma'$ of G^{u} is partitioned into $\Sigma_1 = \Sigma$ and $\Sigma_2 = \Sigma'$, and the restriction of h to Σ is 'in'. To construct G^{u} from G, change every u_j with $A_j \in N_2$ in the above rule into u'_j, where u'_j is obtained from u_j by changing every label σ into its primed version σ', and define $h(\sigma') = h(\sigma) = \mathrm{in}(\sigma)$ for every $\sigma \in \Sigma$.

Theorem 82 *For every MCFT-transducer G over Σ and every $\Delta \subseteq \Sigma$, there is a polynomial time algorithm that, on input $w \in \Delta^*$, outputs an RTG H_w and an MCFT-transducer G_w such that*

$$L(H_w) = \{d \in L(G_{\mathrm{der}}) \mid \mathrm{val}(d) \in \tau(G_w)\} \qquad \textit{and} \qquad \tau(G_w) = \{(t_1, t_2) \in \tau(G) \mid \mathrm{yd}_\Delta(t_1) = w\} \ .$$

The degree of the polynomial is $\mu(G) \cdot (\theta(G) + 1) \cdot (\lambda(G) + 1)$.

Proof We first show how to construct the RTG H_w. Note that $L(H_w)$ should consist of all derivation trees $d \in L(G_{\mathrm{der}})$ such that $\mathrm{yd}_\Delta(\mathrm{val}(d)_1) = w$, where $\mathrm{val}(d)_1$ is the first tree of the forest $\mathrm{val}(d)$. To show this, we may assume that G is i/o-disjoint with Σ partitioned into Σ_1 and Σ_2 and with $\Delta \subseteq \Sigma_1$. In fact, let (G^{u}, h) be an i/o-disjoint cover of G with the properties described before this theorem. Now let H^{u}_w be an RTG such that $L(H^{\mathrm{u}}_w) = \{d \in L(G^{\mathrm{u}}_{\mathrm{der}}) \mid \mathrm{yd}_\Delta(\mathrm{val}(d)_1) = w\}$. By the proof of Lemma 21 there is a projection π such that $\hat{\pi}(L(G^{\mathrm{u}}_{\mathrm{der}})) = L(G_{\mathrm{der}})$ and $\mathrm{val}(\hat{\pi}(d)) = \hat{h}(\mathrm{val}(d))$ for every $d \in L(G^{\mathrm{u}}_{\mathrm{der}})$. Applying $\hat{\pi}$ to the rules of H^{u}_w, we obtain an RTG H_w such that $L(H_w) = \hat{\pi}(L(H^{\mathrm{u}}_w))$. Clearly, H_w satisfies the above requirement.

Assuming that $G = (N, \mathcal{N}, \Sigma, (S_1, S_2), R)$ is i/o-disjoint with Σ partitioned into Σ_1 and Σ_2 and with $\Delta \subseteq \Sigma_1$, we construct the MCFTG $G^{\#} = (N \cup \{S'\}, \mathcal{N} \cup \{S'\}, \Sigma \cup \{\#^{(2)}\}, S', R^{\#})$, where S' is a new nonterminal, $\#$ is a new terminal, and $R^{\#}$ contains all rules of R and the rule $\rho_{\#} = S' \to (\#(S_1, S_2), \mathcal{L})$ with $\mathcal{L} = \{(S_1, S_2)\}$. Note that

$$L(G^{\#}) = \{\#(t_1, t_2) \mid (t_1, t_2) \in \tau(G)\} \qquad \text{and} \qquad L(G^{\#}_{\mathrm{der}}) = \{\rho_{\#}(d) \mid d \in L(G_{\mathrm{der}})\} \ .$$

By Theorem 72 there is a polynomial time algorithm that, on input $w \in \Delta^*$, outputs an RTG $H^{\#}_w$ such that $L(H^{\#}_w) = \{d \in L(G^{\#}_{\mathrm{der}}) \mid \mathrm{yd}_\Delta(\mathrm{val}(d)) = w\}$. We construct the RTG H_w from $H^{\#}_w$ by removing $\rho_{\#}$; i.e., changing every initial rule $S \to \rho_{\#}(C)$ of $H^{\#}_w$ into all rules $S \to \rho(C_1, \ldots, C_k)$ such that $C \to \rho(C_1, \ldots, C_k)$ is a rule of $H^{\#}_w$. Then $L(H_w) = \{d \in L(G_{\mathrm{der}}) \mid \mathrm{yd}_\Delta(\mathrm{val}(d)) = w\}$ because $\# \notin \Delta$. Clearly, since Σ_1 and Σ_2 are disjoint and $\Delta \subseteq \Sigma_1$, we have $\mathrm{yd}_\Delta(\mathrm{val}(d)) = w$ if and only if $\mathrm{val}(d) = (t_1, t_2)$ with $\mathrm{yd}_\Delta(t_1) = w$. Thus, H_w satisfies the requirement.

Finally, we construct G_w from G and H_w as in the proof of Theorem 72 with initial big nonterminal $(S_1, S_2) \otimes S_w = (\langle S_1, S_w\rangle, \langle S_2, S_w\rangle)$. Then $\tau(G_w) = \mathrm{val}(L(H_w))$, and hence G_w satisfies the requirement. ∎

Remarks similar to those following Theorem 72 are also valid here. For $\Delta = \Sigma$, Theorem 82 solves the parsing and translation problem for MCFTG-transducers as tree transducers in polynomial time. For every input tree $t \in T_\Sigma$ the algorithm produces as output an RTG H_t such that $L(H_t) = \{d \in L(G_{\mathrm{der}}) \mid \exists t' \in T_\Sigma\colon \mathrm{val}(d) = (t, t')\}$. The algorithm can be extended to test in linear time whether or not t is in the domain of $\tau(G)$, by testing whether $L(H_t)$ is nonempty. Additionally, if $L(H_t) \neq \emptyset$, then it can also compute in linear time a derivation tree $d \in L(H_t)$ and a tree $t' \in T_\Sigma$ such that $\mathrm{val}(d) = (t, t')$. Thus, t' is a possible translation of t.

For $\Delta \subseteq \Sigma^{(0)} \setminus \{e\}$, we are in the situation described before Theorem 82. For every input string $w \in \Delta^*$ the algorithm outputs an MCFT-transducer G_w such that $\tau(G_w)$ is the set of all pairs of syntactic trees $(t_1, t_2) \in \tau(G)$ such that t_1 is a syntactic tree for w; i.e., $\mathrm{yd}_\Delta(t_1) = w$. Using H_w as before, the algorithm can be extended to test in linear time whether w is in the domain of the string transduction $\{(\mathrm{yd}_\Delta(t_1), \mathrm{yd}_\Delta(t_2)) \mid (t_1, t_2) \in \tau(G)\}$ realized by G, and if so compute a derivation tree $d \in L(H_w)$, its value (t_1, t_2) such that $\mathrm{yd}_\Delta(t_1) = w$, and the string $w' = \mathrm{yd}_\Delta(t_2)$. Thus, t_1 is a syntactic tree of w and t_2 is a syntactic tree of a possible translation w' of w. Note that, since the proof of Theorem 82 is based on Theorem 72, these parsing and translation algorithms for MCFT-transducers are, again, based on a parsing algorithm for MCFGs.

Let us finally consider the class of string transductions realized by MCFT-transducers as discussed above. We first restrict attention to the case $\Delta = \Sigma^{(0)} \setminus \{e\}$, which means that each MCFT-transducer G realizes the string transduction $\{(\mathrm{yd}(t_1), \mathrm{yd}(t_2)) \mid (t_1, t_2) \in \tau(G)\}$. Let us call this a yMCFT-transduction. We can define MCF-transducers in the obvious way, with S_1 and S_2 being the only nonterminals of rank 0.

It should now be clear that we can generalize Corollary 70 as follows: The yMCFT-transductions coincide with the MCF-transductions (and with the yMRT-transductions). These MCF-transductions can also be characterized as the $\mathrm{yDT}_{\mathrm{fc}}$-bimorphisms, or equivalently, as the bimorphisms determined by deterministic tree-walking transducers (cf. the third paragraph after Theorem 76). Since there is an analogue of Lemma 21 for MCFT-transducers (as discussed before Theorem 82), the MCF-transductions are closed under string homomorphisms. This implies that, for every MCFT-transducer G and every set $\Delta \subseteq \Sigma^{(0)} \setminus \{e\}$ of lexical symbols, the string transduction $\{(\mathrm{yd}_\Delta(t_1), \mathrm{yd}_\Delta(t_2)) \mid (t_1, t_2) \in \tau(G)\}$ is also a yMCFT-transduction.

10. Parallel and general MCFTG

In this last section we consider two natural extensions of the MCFTG that allow the grammar to make an unbounded number of copies of subtrees. The definitions of the syntax and semantics of these extensions are easy variants of those for the MCFTG. The first extension is the *parallel* MCFTG (or PMCFTG), which is the obvious generalization of the well-known parallel MCFG of [87]. In a parallel MCFTG (or parallel MCFG), two or more occurrences of the same nonterminal may appear in the right-hand side of a rule. In the least fixed point semantics the terminal tree generated by that nonterminal is therefore copied. In the derivation semantics, after application of the rule, the occurrences must be rewritten in exactly the same way in the remainder of the derivation. The second generalization, which we only briefly consider, is the *general* (P)MCFTG, for which we drop the restriction that the rules must be linear. Thus, two or more occurrences of the same variable may appear in the same tree of the right-hand side of a rule and, when the rule is applied in a derivation step, the tree that is the current value of the variable is copied. The classical (nondeleting) IO context-free tree grammar is the general MCFTG of multiplicity 1.

A *parallel multiple context-free tree grammar* (in short, PMCFTG) is a system $G = (N, \mathcal{N}, \Sigma, S, R)$ as in Definition 5 except that the right-hand side u of a rule $A \to (u, \mathcal{L}) \in R$ is not required to be uniquely N-labeled. The least fixed point semantics of G is defined just as for an MCFTG. As an example, the PMCFTG G with $N = \mathcal{N} = \{S\}$ and $\Sigma = \{\sigma^{(2)}, a^{(0)}, b^{(0)}\}$ using the rules

$$S \to (\sigma(S, S), \{S\}) \qquad S \to (a, \emptyset) \qquad \text{and} \qquad S \to (b, \emptyset)$$

generates the tree language $L(G)$ consisting of all full binary trees over Σ of which all leaves have the same label. Thus, $\mathrm{yd}(L(G)) = \{a^{2^n} \mid n \in \mathbb{N}_0\} \cup \{b^{2^n} \mid n \in \mathbb{N}_0\}$. In fact, from the least fixed point semantics we first obtain that a and b are in $L(G)$. Next, we obtain that the trees $\sigma(S, S)[S \leftarrow a] = \sigma(a, a)$ and $\sigma(S, S)[S \leftarrow b] = \sigma(b, b)$ are in $L(G)$, and then we confirm that $\sigma(S, S)[S \leftarrow \sigma(a, a)] = \sigma(\sigma(a, a), \sigma(a, a))$ is in $L(G)$, etc. Here we use the trivial fact that a tree homomorphism (and hence a second-order substitution) replaces different occurrences of the same nonterminal by the same tree. Since $\mathrm{yd}(L(G))$ is not semi-linear, PMCFTGs are more powerful than MCFTGs, even when they are used to define string languages via the yields of the generated tree languages.

Intuitively, for a rule $A \to (u, \mathcal{L})$ of G, it is still the case that every big nonterminal $B \in \mathcal{L}$ occurs "spread-out" exactly once in u, but now each nonterminal of B may occur more than once in u. More precisely, for each big nonterminal $B = (C_1, \ldots, C_m) \in \mathcal{L}$ with $C_1, \ldots, C_m \in N$, there is a unique set $P_B \subseteq \mathrm{pos}_N(u)$ of positions such that $\{u(p) \mid p \in P_B\} = \{C_1, \ldots, C_m\}$, and we have that $P_B \cap P_{B'} = \emptyset$ for every other $B' \in \mathcal{L}$ and $\mathrm{pos}_N(u) = \bigcup_{B \in \mathcal{L}} P_B$. After the application of the rule, all occurrences of each nonterminal C_i must be rewritten in the same way. This idea was first introduced for context-free grammars in [80] with a least fixed point semantics; for a rewriting semantics similar to the one in Section 3.3 we refer to [88].

Derivation trees can be defined for G as in Section 3.2 with the same results, which are proved in the same way, with one notable exception. Statements (1) and (2) of Lemma 10 do not hold and must be reformulated. For our purposes here it suffices to replace them by the following weaker statements:

(1) $\mathrm{occ}_\Delta(\mathrm{val}(d)) = \bigcup_{\rho \in \mathrm{occ}_R(d)} \mathrm{occ}_\Delta(\mathrm{rhs}(\rho))$ for every $\Delta \subseteq \Sigma$, and

(2) $\mathrm{occ}_N(\mathrm{val}(d)) = \bigcup_{B \in \mathrm{occ}_\mathcal{N}(d)} \mathrm{occ}(B)$,

which can easily be proved by induction on the structure of d. The rewriting semantics in Section 3.3 also applies to PMCFTGs without change. For instance, the tree $\sigma(\sigma(a, a), \sigma(a, a))$ is derived by the above grammar in three derivation steps:

$$S^\varepsilon \Rightarrow_G^{\rho_1, \varepsilon} \sigma(S^1, S^1) \Rightarrow_G^{\rho_1, 1} \sigma(\sigma(S^{11}, S^{11}), \sigma(S^{11}, S^{11})) \Rightarrow_G^{\rho_2, 11} \sigma(\sigma(a, a), \sigma(a, a)) \ ,$$

where ρ_1 is the first rule of G and ρ_2 is the second.

The results and proofs of Section 4.1 on basic normal forms are also valid for PMCFTGs. The same is true for Lemmas 30 and 40. However, we did not further study the lexicalization of PMCFTGs. Thus, we leave it as an open problem whether finitely ambiguous PMCFTGs can be lexicalized, which we conjecture to be true. The results and proofs of Section 6 are also valid for PMCFTGs (without the statements on lexicalization). Thus, for every PMCFTG there are an equivalent monadic PMCFTG, an equivalent footed PMCFTG, and an equivalent "parallel" MC-TAG (provided that the generated tree language is root consistent).

Parallel MCFGs (in short, PMCFGs) can be defined as in Section 7, and all the results and proofs in that section are also valid for the parallel case, except Corollary 65 on lexicalization. Thus, we have that yPMCFT = PMCF = yPMRT. Moreover, PMCFGs and PMCFTGs can be parsed in polynomial time; i.e., Lemma 71 and Theorem 72 also hold in the parallel case (cf. [66, 87]). However, as observed in [87], the degree of the polynomial is one more than in those results because in the proof of Lemma 71, in the construction of the rules of H_w, it must be checked additionally in linear time that $w[\ell(C_{i_1}), r(C_{i_1})] = w[\ell(C_{i_2}), r(C_{i_2})]$ whenever $C_{i_1} = C_{i_2}$ (where C_{i_1} may occur in a different u_j than C_{i_2}). It should also be noted that, for a given derivation tree d, the syntactic tree $t = \mathrm{val}(d)$ can no longer be computed in linear time. Instead, it should be clear that in linear time a directed acyclic graph g can be computed that represents the tree t with shared nodes. In the case where $\Sigma^{(0)} \subseteq \Delta$, this graph g can be unfolded into t in time linear in the size of g plus the size of $w = \mathrm{yd}_\Delta(t)$, and thus t is obtained in the required polynomial time from the string w by the parsing algorithm.

The results of Section 8 (except Corollaries 77, 78 and 80) as well as those of Section 9 are also valid for the parallel case provided that we change $\mathrm{DMT}_{\mathrm{fc}}$ into $\mathrm{DMT}_{\mathrm{sp}}$, and $\mathrm{DT}_{\mathrm{fc}}$ into DT. The proofs are also the same, except that in the proof of Lemma 73 we do not have to consider the state sequences of M, and for the proof of Lemma 75 we do not need Proposition 74 and we have to redefine state sequences, as follows. Roughly speaking, the new state sequences are the old ones from which repetitions have been removed; thus, they can be viewed as 'state sets' (cf. [30, Definition 3.1.8]). Formally, let $M = (Q, \Omega, \Sigma, q_0, R)$ be a $\mathrm{DMT}_{\mathrm{sp}}$-transducer, and consider a fixed order $p_1 \sqsubset \cdots \sqsubset p_r$ on the set $Q = \{p_1, \ldots, p_r\}$ of states of M. For a subset $Q' = \{p_{i_1}, \ldots, p_{i_m}\}$ of Q with $i_1 < \cdots < i_m$, we define the state sequence $\mathrm{seq}(Q') = p_{i_1} \cdots p_{i_m}$. Now let $q_1, \ldots, q_n \in Q$ and $n \in \mathbb{N}_0$, and let $\omega \in \Omega^{(k)}$ with $k \in \mathbb{N}_0$. For $i \in [k]$ we (re-)define $\mathrm{sts}_{\omega,i}(q_1, \ldots, q_n) \in Q^*$ to be the sequence of states

$$\mathrm{sts}_{\omega,i}(q_1, \ldots, q_n) = \mathrm{seq}(\{q' \in Q \mid \exists j \in [n] \colon \langle q', y_i \rangle \in \mathrm{occ}_{Q \times Y}(\mathrm{rhs}_M(q_j, \omega))\}) \ .$$

Then $\mathrm{sts}(s, p)$ and $\mathrm{sts}(M)$ can be defined as in Section 8, and with these definitions the proof of Lemma 75 is valid. Note that $\mathrm{sts}(M)$ is now finite for every $\mathrm{DMT}_{\mathrm{sp}}$-transducer. Consequently, we have that PMCFT = $\mathrm{DMT}_{\mathrm{sp}}$(RT) and PMRT = DT(RT). As further consequences we obtain the known result $\mathrm{yDMT}_{\mathrm{sp}}$(RT) = yDT(RT), which was proved in [28, Theorem 15], and the known result PMCF = yDT(RT), which was proved in [91, Theorem 3.1] by taking into account the well-known fact that string-valued attribute grammars without inherited attributes generate yDT(RT). As in Section 8, the multiplicity of the grammars corresponds to the copying power of the transducers. Thus, m-PMCFT = $\mathrm{DMT}_{\mathrm{sp},(m)}$(RT) and m-PMRT = $\mathrm{DT}_{(m)}$(RT) and m-PMCF = $\mathrm{yDT}_{(m)}$(RT), where the prefix 'm-' means that the grammars have multiplicity at most m and the subscript '(m)' means that the transducers are m-copying (with the new definition of state sequence). As shown in [30, Theorem 3.2.5] by a pumping lemma for $\mathrm{yDT}_{(m)}$(RT), the language $L_m = \{a_1^n a_2^n \cdots a_{2m+2}^n \mid n \in \mathbb{N}_0\}$ is in $(m+1)$-MCF but not in m-PMCF. As results analogous to those in Section 9 we obtain that the PMCFT-transductions are the same as the $\mathrm{DMT}_{\mathrm{sp}}$-bimorphisms, and the PMRT-transductions are the same as the DT-bimorphisms, and hence by [38] they coincide with the d-MBOT-bimorphisms, where the d-MBOTs are not necessarily linear. Moreover, PMCFT-transductions can be parsed and translated in polynomial time (with the degree of the polynomial one more than in Theorem 82).

Finally we consider a further extension of PMCFTGs. Until now we have restricted our grammars to be simple (i.e., linear and nondeleting), which means that for every rule $(A_1, \ldots, A_n) \to ((u_1, \ldots, u_n), \mathcal{L})$ and every $j \in [n]$, the tree u_j contains every variable in $X_{\mathrm{rk}(A_j)}$ exactly once. We now drop the linearity condition and just require every such variable to occur at least once. Technically it is convenient to achieve this by *redefining the notion of pattern* (see the first paragraph of Section 2.3). Thus, we redefine the set $P_\Sigma(X_k)$ of patterns of rank k to consist of all trees $t \in T_\Sigma(X_k)$ such that $\mathrm{occ}_X(t) = X_k$; i.e., each $x \in X_k$ occurs at least once in t. It should be noted that this also changes our definition of tree homomorphism, which is now only required to be nondeleting, and hence that of second-order

substitution. Clearly, Lemma 1 is not true anymore. For our purposes here it can be replaced by the following weaker statements:

(1) $\mathrm{occ}_X(\hat{h}(t)) = \mathrm{occ}_X(t)$, and

(2) $\mathrm{occ}_\Sigma(\hat{h}(t)) = \bigcup_{\tau \in \mathrm{occ}_\Sigma(t)} \mathrm{occ}_\Sigma(h(\tau))$.

The remaining definitions and results of Section 2.3 can be taken over without change.

The definition of a *general parallel multiple context-free tree grammar* (in short, gPMCFTG) is identical to the one of a PMCFTG with the new meaning of $P_{N\cup\Sigma}(X)$ as above. The semantics of a gPMCFTG G is defined just as for an MCFTG. The class of tree languages generated by gPMCFTGs is denoted by $\mathrm{PMCFT_g}$. Derivation trees are defined for G just as for an MCFTG, and Section 3.2 is valid for gPMCFTGs with the same change of Lemma 10 as stated above for PMCFTGs. The rewriting semantics in Section 3.3 is also valid for gPMCFTGs. The semantics of a PMCFTG is essentially an "inside-out" semantics in the sense of [31]. In fact, consider a classical IO context-free tree grammar G such that (i) G is nondeleting (i.e., every variable in the left-hand side of a rule also occurs in the right-hand side) and (ii) the right-hand side of each rule is uniquely N-labeled (i.e., every nonterminal occurs at most once in the right-hand side of each rule). Viewing G as a gPMCFTG in the obvious way, it is easy to see that the least fixed point semantics of G as a gPMCFTG coincides with the least fixed point semantics of G as an IO context-free tree grammar as stated in [31, Theorem 3.4]. Since requirements (i) and (ii) are a normal form for IO context-free tree grammars (cf. [35, Theorem 3.1.10]), this shows that all IO context-free tree languages can be generated by gPMCFTGs. More precisely, they are the tree languages generated by the (nonparallel) gMCFTGs of multiplicity 1.

As an example, the gPMCFTG G with $N = \mathcal{N} = \{S^{(0)}, A^{(1)}, B^{(1)}\}$ and $\Sigma = \{\sigma^{(2)}, a^{(0)}, b^{(0)}\}$ using the rules

$$S \to A(b) \qquad A(x_1) \to B(A(\sigma(a, x_1))) \qquad A(x_1) \to x_1 \qquad \text{and} \qquad B(x_1) \to \sigma(x_1, x_1) \ ,$$

generates the tree language $L(G)$ consisting of all trees $t_1[x_1 \leftarrow t_2]$, where t_1 is a full binary tree over $\{\sigma, x_1\}$ of height n and t_2 equals $(\sigma a)^n b$. Thus, $\mathrm{yd}(L(G)) = L_{\mathrm{ec}} = \{(a^n b)^{2^n} \mid n \in \mathbb{N}\}$. For $n = 2$, the tree $t = \sigma(\sigma(\sigma a\sigma ab, \sigma a\sigma ab), \sigma(\sigma a\sigma ab, \sigma a\sigma ab))$ is obtained by the derivation

$$\begin{aligned} S^\varepsilon &\Rightarrow_G^{\rho_1,\varepsilon} A^1(b) \Rightarrow_G^{\rho_2,1} B^{11}(A^{12}(\sigma ab)) \Rightarrow_G^{\rho_2,12} B^{11}(B^{121}(A^{122}(\sigma a\sigma ab))) \\ &\Rightarrow_G^{\rho_3,122} B^{11}(B^{121}(\sigma a\sigma ab)) \Rightarrow_G^{\rho_4,121} B^{11}(\sigma(\sigma a\sigma ab, \sigma a\sigma ab)) \Rightarrow_G^{\rho_4,11} t \ , \end{aligned}$$

which corresponds to the "inside-out" derivation of the IO context-free tree grammar G, but is, for instance, also obtained by the "outside-in" derivation

$$\begin{aligned} S^\varepsilon &\Rightarrow_G^{\rho_1,\varepsilon} A^1(b) \Rightarrow_G^{\rho_2,1} B^{11}(A^{12}(\sigma ab)) \Rightarrow_G^{\rho_4,11} \sigma(A^{12}(\sigma ab), A^{12}(\sigma ab)) \\ &\Rightarrow_G^{\rho_2,12} \sigma(B^{121}(A^{122}(\sigma a\sigma ab)), B^{121}(A^{122}(\sigma a\sigma ab))) \\ &\Rightarrow_G^{\rho_4,121} \sigma(\sigma(A^{122}(\sigma a\sigma ab), A^{122}(\sigma a\sigma ab)), \sigma(A^{122}(\sigma a\sigma ab), A^{122}(\sigma a\sigma ab))) \Rightarrow_G^{\rho_3,122} t \ . \end{aligned}$$

The language L_{ec} is the well-known example of an IO context-free tree language that is not an OI context-free tree language (see [35, Section 4.3]). It is shown in [18, Theorem 3.16], using again the pumping lemma for yDT(RT), that L_{ec} is not in yDT(RT), and hence not in PMCF. Thus, gPMCFTGs are more powerful than PMCFTGs, even when they are used to define string languages via the yields of the generated tree languages. Note that the above grammar is even a gMCFTG because the right-hand sides of its rules are uniquely N-labeled.[50] The multiple context-free tree grammars in [8] are the gMCFTGs, whereas our MCFTGs are there called *linear* multiple context-free tree grammars. It is shown in [8] that the closure of MCF under IO-substitution is included in $\mathrm{yMCFT_g}$ and that the string languages in this closure satisfy the constant-growth property and can be recognized in polynomial time.

The only result we have for gPMCFTGs is their characterization in terms of macro tree transducers. Let $\mathrm{DMT_{np}}$ denote the class of tree transductions realized by macro tree transducers with the new definition of pattern (where 'np' stands for 'nondeleting in the parameters'). The semantics of such transducers is as in Section 8. Using the redefined notion of state sequence as for PMCFTGs, the proofs of Lemmas 73 and 75 are still valid. Thus, we obtain that $\mathrm{PMCFT_g} = \mathrm{DMT_{np}}(\mathrm{RT})$. Now let

[50] We do not know whether there is a tree language in PMCFT that is not in $\mathrm{MCFT_g}$; i.e., we do not know whether PMCFT and $\mathrm{MCFT_g}$ are incomparable subclasses of $\mathrm{PMCFT_g}$.

DMT denote the class of tree transductions realized by all (total deterministic) macro tree transducers as known from the literature, which means that also deletion of parameters is allowed; i.e., for a rule $\langle q, \omega(y_1, \ldots, y_k)\rangle(x_1, \ldots, x_m) \to \zeta$, it is just required that $\zeta \in T_{(Q\times Y_k)\cup\Sigma}(X_m)$. Their semantics is still the same as in Section 8. It is proved in [26, Lemma 6.6] that for every DMT-transducer with regular look-ahead there is an equivalent one that is nondeleting in the parameters. Since regular look-ahead can be simulated by relabeling the input tree, this implies that $\mathrm{DMT(RT)} = \mathrm{DMT_{np}(RT)}$. Thus we obtain the characterization $\mathrm{PMCFT_g} = \mathrm{DMT(RT)}$. We observe that the two types (P and g) of copying subtrees that can be realized by gPMCFTGs, correspond for macro tree transducers to the copying of input variables (from Y) and the copying of output variables (or parameters, from X), respectively.

At the end of this section we discuss the class S-CF of synchronized-context-free tree languages introduced in [9] and applied, e.g., in [7]. The logic programs generating these tree languages are essentially tree-valued attribute grammars, which means that S-CF = AT(RT), where AT denotes the class of attributed tree transductions (see, e.g., [26, 39]). It was shown in [22] that AT(RT) is the class of tree languages obtained by unfolding the term graphs generated by a context-free graph grammar, where a term graph is a directed acyclic graph representing a tree with shared subtrees (cf. Corollary 78). It is well known that $\mathrm{DT} \subsetneq \mathrm{AT} \subsetneq \mathrm{DMT}$ (see, e.g., [39]). Thus, the class AT(RT) is included in $\mathrm{PMCFT_g}$. It seems to be unknown whether the inclusion is proper. It follows from [26, Theorem 7.1] that $\mathrm{DMT_{fc}(RT)} \subseteq \mathrm{AT(RT)}$. Thus, MCFT is included in AT(RT), but the relationship of AT(RT) to PMCFT is not clear. However, $\mathrm{PMCF} = \mathrm{yDT(RT)} \subsetneq \mathrm{yAT(RT)}$, because $L_{\mathrm{ec}} \in \mathrm{yAT(RT)}$. Hence we have $\mathrm{MCFT} \subsetneq \mathrm{AT(RT)} \subseteq \mathrm{PMCFT_g}$ and $\mathrm{MCF} \subsetneq \mathrm{PMCF} \subsetneq \mathrm{yAT(RT)} \subseteq \mathrm{yPMCFT_g}$. We finally note that the class $\mathrm{CFT_{sp}}$ is characterized in terms of a special type of attributed tree transducers in [72].

11. Conclusion

We have proved in Theorem 44 that every finitely ambiguous MCFTG can be lexicalized, for an arbitrary set Δ of lexical symbols. A remaining question is whether the given bounds on the multiplicity and width of the resulting MCFTG are optimal. In the particular case where all lexical symbols in Δ have rank 0, the multiplicity stays the same, but the width increases by 1. By Theorems 44 and 61 together, there is also an equivalent lexicalized grammar of width at most 1 but with increased multiplicity. A similar question is relevant for the transformation of an MCFTG into an equivalent MC-TAG (Theorem 58), and for the lexicalization of MC-TAGs (Theorem 60). As shown in [25], the factor mrk_Σ^2 in Theorems 58, 60, and 61 can be reduced to mrk_Σ by combining the two constructions in the proofs of Theorem 49 and Lemma 55 into one.

All our grammar transformations produce an MCFTG that is grammatically close (i.e., $\mathrm{LDT^R}$-equivalent) to the given MCFTG, except for the transformation of an MCFTG into a monadic MCFTG (Lemma 47 and Theorem 61), for which we could only prove $\mathrm{LDT^R}$-equivalence in the special case in which all lexical symbols in Δ have rank 0. As already observed in footnotes 20 and 24, this problem can be "solved" by considering the weaker notion of $\mathrm{DT^R_{fc}}$-equivalence instead of $\mathrm{LDT^R}$-equivalence, where $\mathrm{DT^R_{fc}}$ is the class of transductions realized by finite-copying top-down tree transducers with regular look-ahead. The definition of $\mathrm{DT^R_{fc}}$-equivalence is the same as that of $\mathrm{LDT^R}$-equivalence in Definition 15. Since $\mathrm{DT^R_{fc}}$ is closed under composition (see, e.g., [30, Theorem 5.4]), this is indeed an equivalence relation. Actually, we feel that $\mathrm{DT^R_{fc}}$-equivalence is a better formalization of the notion of grammatical closeness than $\mathrm{LDT^R}$-equivalence because it can also handle the combination of rules as needed, e.g., in the proof of Lemma 47. Such a combination of rules is also needed for the binarization of grammars (which we did not study for MCFTGs), to transform the derivation trees of the binarized grammar into those of the original one. An MCFTG G is *binary* if its rule-width $\lambda(G)$ is at most 2. In view of Lemma 71 and Theorem 72, binarization is important for parsing (see, e.g., [45, 78]). We note that most of our constructions preserve $\lambda(G)$. The two exceptions are Lemmas 28 and 47 which decrease and increase $\lambda(G)$, respectively.

In Theorem 76 we have proved a characterization of MCFTGs in terms of finite-copying macro tree transducers, and from that we have deduced characterizations in terms of monadic second-order logic (Corollary 77), context-free graph grammars (Corollary 78), and abstract categorial grammars (Corollary 80). It would be worthwhile to investigate whether there are more results from the literature on macro tree transducers that can be applied to MCFTGs.

In Section 9 we have introduced the MCFT-transducer and we have shown that they realize the $\mathrm{DMT_{fc}}$-bimorphisms and hence the DMSOT-bimorphisms. This class of MCFT-transductions deserves further study. Only subclasses have been investigated in the literature. As stated in [77, Example 5], the

MRT-transductions are not closed under composition. We do not know whether the MCFT-transductions are closed under composition or whether composition gives rise to a proper hierarchy. Another question is whether or not every functional MCFT-transduction is a composition of deterministic macro tree transductions.

Our remaining problems concern the extensions of MCFTGs discussed in Section 10: the PMCFTGs and the g(P)MCFTGs. As observed in that section it is open whether PMCFTGs can be lexicalized, and the same is true for g(P)MCFTGs. Although Theorem 76 can be generalized to PMCFTGs and gPMCFTGs, it is not clear whether there are natural generalizations of the three corollaries mentioned above. Also, a characterization of $\mathrm{MCFT_g}$ is missing. Finally, it would be interesting to determine the correctness (or incorrectness) of the obvious Hasse diagram of the six classes MRT, MCFT, PMRT, PMCFT, $\mathrm{MCFT_g}$, $\mathrm{PMCFT_g}$. The tree language $\{a^n b^n \triangleright \mid n \in \mathbb{N}_0\}$, which we considered at the end of Section 8, is in MCFT (even in $\mathrm{CFT_{sp}}$) but not in PMRT because all monadic tree languages in the class DT(RT) are regular [81, Theorem 4]. The IO context-free tree language L_{ec} that we considered in Section 10 is in $\mathrm{MCFT_g}$ but not in PMCFT. The PMRTG (of multiplicity 1) that we considered in the second paragraph of Section 10, generates a tree language that is not in MCFT. However, we do not know whether there exists a tree language in PMCFT (or even in PMRT) that is not in $\mathrm{MCFT_g}$. If we also add the six classes (as above) with multiplicity 1, then the situation is less clear. In view of [33, Corollary 3.5] we guess that 1-PMRT = HOM(RT) where HOM is the class of all (not necessarily simple) tree homomorphisms. Thus, apart from the trivial inclusions, we obtain the additional inclusion 1-PMRT $\subseteq$ 1-$\mathrm{MCFT_g}$ because the class of IO context-free tree languages is closed under arbitrary tree homomorphisms [32, Corollary 6.4]. The tree language of Example 6, which we also considered at the end of Section 8, is in MRT but not in 1-$\mathrm{MCFT_g}$ because it cannot be generated by an IO context-free tree grammar as shown in [20, Section 5]. However, we do not know whether there exists a tree language in MRT that is not in 1-$\mathrm{PMCFT_g}$; i.e., that cannot be generated by a parallel IO context-free tree grammar.

References

References

[1] Rajeev Alur and Loris D'Antoni. Streaming tree transducers. *CoRR*, abs/1104.2599, 2011.

[2] André Arnold and Max Dauchet. Bi-transductions de forêts. In S. Michaelson and Robin Milner, editors, *ICALP*, pages 74–86. Edinburgh University Press, 1976.

[3] Jean-Michel Autebert, Jean Berstel, and Luc Boasson. Context-free languages and pushdown automata. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 1, chapter 3, pages 111–174. Springer, 1997.

[4] Brenda S. Baker. Composition of top-down and bottom-up tree transductions. *Inform. and Control*, 41(2):186–213, 1979.

[5] Michel Bauderon and Bruno Courcelle. Graph expressions and graph rewritings. *Math. Systems Theory*, 20(2-3):83–127, 1987.

[6] Henrik Björklund, Martin Berglund, and Petter Ericson. Uniform *vs.* nonuniform membership for mildly context-sensitive languages: A brief survey. *Algorithms*, 9(2):32, 2016.

[7] Yohan Boichut, Jacques Chabin, and Pierre Réty. Towards more precise rewriting approximations. In Adrian Horia Dediu, Enrico Formenti, Carlos Martín-Vide, and Bianca Truthe, editors, *Proc. 9th Int. Conf. Language and Automata Theory and Applications*, volume 8977 of LNCS, pages 652–663. Springer, 2015.

[8] Pierre Bourreau, Laura Kallmeyer, and Sylvain Salvati. On IO-copying and mildly-context sensitive formalisms. In Glyn Morrill and Mark-Jan Nederhof, editors, *Proc. 17th and 18th Int. Conf. Formal Grammar*, volume 8036 of LNCS, pages 1–16. Springer, 2013.

[9] Jacques Chabin, Jing Chen, and Pierre Réty. Synchronized-contextfree tree-tuple languages. Technical Report RR-2006-13, INRIA, France, 2006. Available at `https://hal.inria.fr/inria-00464114`.

[10] John Chen. *Towards Efficient Statistical Parsing using Lexicalized Grammatical Information.* PhD thesis, University of Delaware, Newark, USA, 2001.

[11] Bruno Courcelle. An axiomatic definition of context-free rewriting and its application to NLC graph grammars. *Theoret. Comput. Sci.*, 55(2–3):141–181, 1987.

[12] Bruno Courcelle and Joost Engelfriet. *Graph Structure and Monadic Second-Order Logic — A Language-Theoretic Approach*, volume 138 of *Encyclopedia of mathematics and its applications.* Cambridge University Press, 2012.

[13] Bruno Courcelle and Paul Franchi-Zannettacci. Attribute grammars and recursive program schemes. *Theoret. Comput. Sci.*, 17:163–191, 235–257, 1982.

[14] Frank Drewes, Hans-Jörg Kreowski, and Annegret Habel. Hyperedge replacement graph grammars. In Grzegorz Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformations, Volume 1: Foundations*, pages 95–162. World Scientific, 1997.

[15] Joost Engelfriet. Bottom-up and top-down tree transformations—a comparison. *Math. Systems Theory*, 9(3):198–231, 1975.

[16] Joost Engelfriet. Tree automata and tree grammars. Technical Report DAIMI FN-10, Aarhus University, 1975. A slightly revised version is available at `http://arxiv.org/abs/1510.02036`.

[17] Joost Engelfriet. Top-down tree transducers with regular look-ahead. *Math. Systems Theory*, 10:289–303, 1977.
[18] Joost Engelfriet. Three hierarchies of transducers. *Math. Systems Theory*, 15(2):95–125, 1982.
[19] Joost Engelfriet. Context-free graph grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 3, pages 125–213. Springer, 1997.
[20] Joost Engelfriet and Gilberto Filé. The formal power of one-visit attribute grammars. *Acta Inform.*, 16:275–302, 1981.
[21] Joost Engelfriet and Linda Heyker. The string generating power of context-free hypergraph grammars. *J. Comput. System Sci.*, 43(2):328–360, 1991.
[22] Joost Engelfriet and Linda Heyker. Context-free hypergraph grammars have the same term-generating power as attribute grammars. *Acta Inform.*, 29(2):161–210, 1992.
[23] Joost Engelfriet and Hendrik Jan Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Trans. Comput. Log.*, 2(2):216–254, 2001.
[24] Joost Engelfriet, Eric Lilin, and Andreas Maletti. Extended multi bottom-up tree transducers. *Acta Inform.*, 46(8):561–590, 2009.
[25] Joost Engelfriet and Andreas Maletti. Multiple context-free tree grammars and multi-component tree adjoining grammars. In Ralf Klasing and Marc Zeitoun, editors, *Proc. 21st Int. Symp. Fundamentals of Computation Theory*, LNCS. Springer, 2017. to appear.
[26] Joost Engelfriet and Sebastian Maneth. Macro tree transducers, attribute grammars, and MSO definable tree translations. *Inform. and Comput.*, 154(1):34–91, 1999.
[27] Joost Engelfriet and Sebastian Maneth. Tree languages generated by context-free graph grammars. In Hartmut Ehrig, Gregor Engels, and Hans-Jörg Kreowski, editors, *Proc. 8th Int. Workshop Theory and Application of Graph Transformation*, volume 1764 of LNCS, pages 15–29, 2000.
[28] Joost Engelfriet and Sebastian Maneth. Output string languages of compositions of deterministic macro tree transducers. *J. Comput. System Sci.*, 64:350–395, 2002.
[29] Joost Engelfriet and Sebastian Maneth. Macro tree translations of linear size increase are MSO definable. *SIAM J. Comput.*, 32(4):950–1006, 2003.
[30] Joost Engelfriet, Grzegorz Rozenberg, and Giora Slutzki. Tree transducers, L systems, and two-way machines. *J. Comput. System Sci.*, 20(2):150–202, 1980.
[31] Joost Engelfriet and Erik M. Schmidt. IO and OI I. *J. Comput. System Sci.*, 15(3):328–353, 1977.
[32] Joost Engelfriet and Erik M. Schmidt. IO and OI II. *J. Comput. System Sci.*, 16(1):67–99, 1978.
[33] Joost Engelfriet and Sven Skyum. The copying power of one-state tree transducers. *J. Comput. System Sci.*, 25(3):418–435, 1982.
[34] Joost Engelfriet and Heiko Vogler. Macro tree transducers. *J. Comput. System Sci.*, 31(1):71–146, 1985.
[35] Michael J. Fischer. *Grammars with Macro-Like Productions*. PhD thesis, Harvard University, 1968.
[36] Akio Fujiyoshi. Epsilon-free grammars and lexicalized grammars that generate the class of the mildly context-sensitive languages. In *Proc. 7th Int. Workshop Tree Adjoining Grammar and Related Formalisms*, pages 16–23, 2005.
[37] Akio Fujiyoshi and Takumi Kasai. Spinal-formed context-free tree grammars. *Theory Comput. Syst.*, 33(1):59–83, 2000.
[38] Zoltán Fülöp, Armin Kühnemann, and Heiko Vogler. A bottom-up characterization of deterministic top-down tree transducers with regular look-ahead. *Inform. Process. Lett.*, 91(2):57–67, 2004.
[39] Zoltán Fülöp and Heiko Vogler. *Syntax-Directed Semantics—Formal Models Based on Tree Transducers*. EATCS Monographs on Theoretical Computer Science. Springer, 1998.
[40] Kilian Gebhardt and Johannes Osterholzer. A direct link between tree-adjoining and context-free tree grammars. In Thomas Hanneforth and Christian Wurm, editors, *Proceedings of the 12th International Conference on Finite-State Methods and Natural Language Processing, FSMNLP 2015*. The Association for Computer Linguistics, 2015.
[41] Ferenc Gécseg and Magnus Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984. A re-edition is available at `http://arxiv.org/abs/1509.06233`.
[42] Ferenc Gécseg and Magnus Steinby. Tree languages. In Grzegorz Rozenberg and Arto Salomaa, editors, *Handbook of Formal Languages*, volume 3, chapter 1, pages 1–68. Springer, 1997.
[43] Jonathan Goldstine, Hing Leung, and Detlef Wotschke. On the relation between ambiguity and nondeterminism in finite automata. *Inform. and Comput.*, 100(2):261–270, 1992.
[44] Carlos Gómez-Rodríguez, Marco Kuhlmann, and Giorgio Satta. Efficient parsing of well-nested linear context-free rewriting systems. In *Proc. 2010 Int. Conf. HLT-NAACL*, pages 276–284. Association for Computational Linguistics, 2010.
[45] Carlos Gómez-Rodríguez and Giorgio Satta. An optimal-time binarization algorithm for linear context-free rewriting systems with fan-out two. In Keh-Yih Su, Jian Su, and Janyce Wiebe, editors, *ACL 2009, Proc. of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 985–993. The Association for Computer Linguistics, 2009. Available at `http://www.aclweb.org/anthology/P09-1111`.
[46] James N. Gray and Michael A. Harrison. On the covering and reduction problems for context-free grammars. *J. ACM*, 19(4):675–698, 1972.
[47] Hendrik Jan Hoogeboom and Paulien ten Pas. Monadic second-order definable text languages. *Theory Comput. Syst.*, 30(4):335–354, 1997.
[48] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to automata theory, languages, and computation*. Addison-Wesley series in computer science. Addison Wesley, second edition, 2001.
[49] Aravind K. Joshi, Leon S. Levy, and Masako Takahashi. Tree adjunct grammars. *J. Comput. System Sci.*, 10(1):136–163, 1975.
[50] Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars and lexicalized grammars. In Maurice Nivat and Andreas Podelski, editors, *Tree Automata and Languages*. North-Holland, 1992.
[51] Aravind K. Joshi and Yves Schabes. Tree-adjoining grammars. In Grzegorz Rozenberg and Arto Salomaa, editors, *Beyond Words*, volume 3 of *Handbook of Formal Languages*, pages 69–123. Springer, 1997.
[52] Yuichi Kaji, Ryuchi Nakanishi, Hiroyuki Seki, and Tadao Kasami. The computational complexity of the universal

recognition problem for parallel multiple context-free grammars. *Computational Intelligence*, 10:440–452, 1994.
[53] Laura Kallmeyer. A declarative characterization of different types of multicomponent tree adjoining grammars. *Res. Lang. Comput.*, 7(1):55–99, 2009.
[54] Laura Kallmeyer. *Parsing Beyond Context-Free Grammars*. Cognitive Technologies. Springer, 2010.
[55] Makoto Kanazawa. The convergence of well-nested mildly context-sensitive grammar formalisms. Invited talk at the 14th Int. Conf. Formal Grammar, 2009. Slides available at `research.nii.ac.jp/~kanazawa`.
[56] Makoto Kanazawa. The pumping lemma for well-nested multiple context-free languages. In Volker Diekert and Dirk Nowotka, editors, *Proc. 13th Int. Conf. Developments in Language Theory*, volume 5583 of LNCS, pages 312–325. Springer, 2009.
[57] Makoto Kanazawa. Second-order abstract categorial grammars as hyperedge replacement grammars. *J. Log. Lang. Inf.*, 19(2):137–161, 2010.
[58] Makoto Kanazawa. Multidimensional trees and a Chomsky-Schützenberger-Weir representation theorem for simple context-free tree grammars. *J. Log. Comput.*, 26(5):1469–1516, 2016.
[59] Makoto Kanazawa and Sylvain Salvati. The copying power of well-nested multiple context-free grammars. In Adrian Horia Dediu, Henning Fernau, and Carlos Martín-Vide, editors, *Proc. 4th Int. Conf. Language and Automata Theory and Applications*, volume 6031 of LNCS, pages 344–355. Springer, 2010.
[60] Makoto Kanazawa and Ryo Yoshinaka. Lexicalization of second-order ACGs. Technical Report NII-2005-012E, National Institute of Informatics, Tokyo, Japan, 2005.
[61] Stephan Kepser and James Rogers. The equivalence of tree adjoining grammars and monadic linear context-free tree grammars. *J. Log. Lang. Inf.*, 20(3):361–384, 2011.
[62] Ines Klimann, Sylvain Lombardy, Jean Mairesse, and Christophe Prieur. Deciding unambiguity and sequentiality from a finitely ambiguous max-plus automaton. *Theoret. Comput. Sci.*, 327(3):349–373, 2004.
[63] Donald E. Knuth. A characterization of parenthesis languages. *Inform. and Control*, 11(3):269–289, 1967.
[64] Marco Kuhlmann. *Dependency Structures and Lexicalized Grammars: An Algebraic Approach*, volume 6270 of LNAI. Springer, 2010.
[65] Marco Kuhlmann and Giorgio Satta. Tree-adjoining grammars are not closed under strong lexicalization. *Comput. Linguist.*, 38(3):617–629, 2012.
[66] Peter Ljunglöf. Practical parsing of parallel multiple context-free grammars. In *Proc. 11th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+11)*, pages 144–152, Paris, France, September 2012. Available at `http://www.aclweb.org/anthology/W12-4617`.
[67] Markus Lohrey, Sebastian Maneth, and Manfred Schmidt-Schauß. Parameter reduction and automata evaluation for grammar-compressed trees. *J. Comput. System Sci.*, 78(5):1651–1669, 2012.
[68] Andreas Maletti. Compositions of extended top-down tree transducers. *Inform. and Comput.*, 206(9–10):1187–1196, 2008.
[69] Andreas Maletti. Synchronous forest substitution grammars. In Traian Muntean, Dimitrios Poulakis, and Robert Rolland, editors, *Proc. 5th Int. Conf. Algebraic Informatics*, volume 8080 of LNCS, pages 235–246. Springer, 2013.
[70] Andreas Maletti and Joost Engelfriet. Strong lexicalization of tree adjoining grammars. In *Proc. 50th Ann. Meeting Association for Computational Linguistics*, pages 506–515, 2012.
[71] Uwe Mönnich. Adjunction as substitution: An algebraic formulation of regular, context-free and tree adjoining languages. In *Proc. 3rd Int. Conf. Formal Grammar*, pages 169–178. Université de Provence, France, 1997. Available at `arxiv.org/abs/cmp-lg/9707012v1`.
[72] Uwe Mönnich. Well-nested tree languages and attributed tree transducers. In *Proc. 10th Int. Conf. Tree Adjoining Grammars and Related Formalisms*. Yale University, 2010. Available at `www2.research.att.com/~srini/TAG+10/papers/uwe.pdf`.
[73] Mark-Jan Nederhof and Heiko Vogler. Synchronous context-free tree grammars. In *Proc. 11th Int. Workshop Tree Adjoining Grammars and Related Formalisms*, pages 55–63. Association for Computational Linguistics, 2012.
[74] Anton Nijholt. *Context-Free Grammars: Covers, Normal Forms, and Parsing*, volume 93 of LNCS. Springer, 1980.
[75] Maurice Nivat. Transductions des langages de Chomsky. *Annales de l'institut Fourier*, 18(1):339–455, 1968. Available at `https://eudml.org/doc/73950`.
[76] Andreas Potthoff and Wolfgang Thomas. Regular tree languages without unary symbols are star-free. In *Proc. 9th Int. Symp. Fundamentals of Computation Theory*, volume 710 of LNCS, pages 396–405. Springer, 1993.
[77] Frank G. Radmacher. An automata theoretic approach to rational tree relations. In Viliam Geffert, Juhani Karhumäki, Alberto Bertoni, Bart Preneel, Pavol Návrat, and Mária Bieliková, editors, *Proc. SOFSEM 2008, 34th Conference on Current Trends in Theory and Practice of Computer Science,*, volume 4910 of LNCS, pages 424–435. Springer, 2008.
[78] Owen Rambow and Giorgio Satta. Independent parallelism in finite copying parallel rewriting systems. *Theoret. Comput. Sci.*, 223(1–2):87–120, 1999.
[79] Jean-Claude Raoult. Rational tree relations. *Bull. Belg. Math. Soc.*, 4:149–176, 1997.
[80] Gene F. Rose. An extension of ALGOL-like languages. *Commun. ACM*, 7(2):52–61, 1964.
[81] William C. Rounds. Mappings and grammars on trees. *Math. Systems Theory*, 4(3):257–287, 1970.
[82] Sylvain Salvati. Encoding second order string ACG with deterministic tree walking transducers. In Shuly Wintner, editor, *Proc. 11th Int. Conf. Formal Grammars*, FG Online Proceedings, pages 143–156. CSLI Publications, 2007.
[83] Aniello De Santo, Alëna Aksënova, and Thomas Graf. An alternate view on strong lexicalization in TAG. In David Chiang and Alexander Koller, editors, *Proceedings of the 12th International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+12)*, pages 93–102. The Association for Computer Linguistics, 2016.
[84] Giorgio Satta. Recognition of linear context-free rewriting systems. In Henry S. Thompson, editor, *Proc. 30th Annual Meeting of the Association for Computational Linguistics*, pages 89–95. Association for Computational Linguistics, 1992.
[85] Yves Schabes. *Mathematical and Computational Aspects of Lexicalized Grammars*. PhD thesis, University of Pennsylvania, Philadelphia, USA, 1990.
[86] Yves Schabes, Anne Abeillé, and Aravind K. Joshi. Parsing strategies with 'lexicalized' grammars: Application to tree adjoining grammars. In *Proc. 12th Int. Conf. Computational Linguistics*, pages 578–583. John von Neumann Society

for Computing Sciences, Budapest, 1988.
[87] Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. On multiple context-free grammars. *Theoret. Comput. Sci.*, 88(2):191–229, 1991.
[88] Sven Skyum. On extensions of ALGOL-like languages. *Inform. and Control*, 26(1):82–97, 1974.
[89] Heiko Stamer. *Restarting Tree Automata: Formal Properties and Possible Variations*. PhD thesis, University of Kassel, Germany, 2009.
[90] Heiko Stamer and Friedrich Otto. Restarting tree automata and linear context-free tree languages. In *Proc. 2nd Int. Conf. Algebraic Informatics*, volume 4728 of LNCS, pages 275–289. Springer, 2007.
[91] Nikè van Vugt. Generalized context-free grammars. Technical Report 96-12, Department of Computer Science, Leiden University, 1996. Master's Thesis, available at: `http://liacs.leidenuniv.nl/assets/PDF/vvugt.96.pdf`.
[92] K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. Characterizing structural descriptions produced by various grammatical formalisms. In *Proc. 25th Ann. Meeting Association for Computational Linguistics*, pages 104–111. Association for Computational Linguistics, 1987.
[93] David J. Weir. *Characterizing Mildly Context-Sensitive Grammar Formalisms*. PhD thesis, University of Pennsylvania, 1988.
[94] David J. Weir. Linear context-free rewriting systems and deterministic tree-walking transducers. In Henry S. Thompson, editor, *Proc. 30th Ann. Meeting Association for Computational Linguistics*, pages 136–143. Association for Computational Linguistics, 1992.
[95] Ryo Yoshinaka. *Extensions and Restrictions of Abstract Categorial Grammars*. PhD thesis, University of Tokyo, 2006.