Preproceedings of the 20th Conference on Formal Grammar

Annie Foret, Glyn Morrill, Reinhard Muskens and Rainer Osswald (eds.)

August 8–9th 2015, Barcelona

Preface

FG provides a forum for the presentation of new and original research on formal grammar, mathematical linguistics and the application of formal and mathematical methods to the study of natural language. Themes of interest include, but are not limited to:

- Formal and computational phonology, morphology, syntax, semantics and pragmatics
- Model-theoretic and proof-theoretic methods in linguistics
- Logical aspects of linguistic structure
- Constraint-based and resource-sensitive approaches to grammar
- Learnability of formal grammar
- Integration of stochastic and symbolic models of grammar
- Foundational, methodological and architectural issues in grammar and linguistics
- Mathematical foundations of statistical approaches to linguistic analysis

Previous Formal Grammar meetings were held in Barcelona (1995), Prague (1996), Aix-en-Provence (1997), Saarbrücken (1998), Utrecht (1999), Helsinki (2001), Trento (2002), Vienna (2003), Nancy (2004), Edinburgh (2005), Malaga (2006), Dublin (2007), Hamburg (2008), Bordeaux (2009), Copenhagen (2010), Ljubljana (2011), Opole (2012), Düsseldorf (2013) and Tübingen (2014).

The present volume collects the papers from the 20th Conference on Formal Grammar celebrated in Barcelona in 2015. This preproceedings comprises two invited contributions, by Robin Cooper and Tim Fernando, and 9 contributed papers selected from 14 submissions.

We thank for support the local organisers of ESSLLI 2015, with which the conference was colocated.

July 2015

Annie Foret
Glyn Morrill
Reinhard Muskens
Rainer Osswald

Program Committee

Alexander Clark	King's College London, UK
Berthold Crysmann	CNRS - LLF, France
Denys Duchier	Université d'Orleans, France
Nissim Francez	Technion, Israel
Philippe de Groote	Inria Nancy, France
Laura Kallmeyer	Heinrich-Heine-Universität Düsseldorf, Germany
Makoto Kanazawa	National Institute of Informatics, Japan
Greg Kobele	University of Chicago, USA
Robert Levine	Ohio State University, USA
Wolfgang Maier	Heinrich-Heine-Universität Düsseldorf, Germany
Stefan Müller	Freie Universität Berlin, Germany
Mark-Jan Nederhof	University of St Andrews, UK
Christian Retoré	LIRMM - Université Montpellier 2, France
Manfred Sailer	Goethe University Frankfurt, Germany
Ed Stabler	UCLA, USA
Jesse Tseng	CNRS - CLLE-ERSS, France
Oriol Valentín	Universitat Politècnica de Catalunya, Spain

Program Chairs and Standing Committee

Annie Foret	IRISA - IFSIC, France
Glyn Morrill	Universitat Politècnica de Catalunya, Spain
Reinhard Muskens	Tilburg University, The Netherlands
Rainer Osswald	Heinrich-Heine-Universität Düsseldorf, Germany

Table of Contents[1]

Invited contributions

Contributed papers

[1] © the individual authors

On the mild context-sensitivity of *k*-Tree Wrapping Grammar

Laura Kallmeyer

Heinrich-Heine-Universität Düsseldorf, Germany
kallmeyer@phil.hhu.de

Abstract. Tree Wrapping Grammar (TWG) has been proposed in the context of formalizing the syntactic inventory of Role and Reference Grammar (RRG). It is close to Tree Adjoining Grammar (TAG) while capturing predicate-argument dependencies in a more appropriate way and being able to deal with discontinuous constituents in a more general way. This paper is concerned with the formal properties of TWG. More particularly, it considers k-TWG, a constrained form of TWG. We show that for every k-TWG, a simple Context-Free Tree Grammar (CFTG) of rank k can be constructed, which is in turn equivalent to a well-nested Linear Context-Free Rewriting System (LCFRS) of fan-out $k+1$. This shows that, when formalizing a grammar theory such as RRG, which is based on thorough and broad empirical research, we obtain a grammar formalism that is mildly context-sensitive.

Keywords: tree rewriting grammars, Role and Reference Grammar, simple context-free tree grammar, mild context-sensitivity

1 Introduction

Tree Wrapping Grammar (TWG) [6] has been introduced in the context of formalizing the syntactic inventory of Role and Reference Grammar (RRG) [15, 14]. A TWG consists of elementary trees, very much in the spirit of Tree Adjoining Grammar (TAG) [5], and from these elementary trees, larger trees are obtained by the operations of *(wrapping) substitution* and *sister adjunction.* (Wrapping) substitutions are supposed to add syntactic arguments while sister adjunction is used to add modifiers and functional operators. A discontinuous argument can be split via the wrapping substitution operation: the argument tree has a specific split node v. When adding such an argument to a predicate tree, the lower part (rooted in v) fills an argument slot via substitution while the upper ends up above the root of the target predicate tree.

[6] adopts the rather flat syntactic structure from RRG with categories CORE, CLAUSE and SENTENCE. A sample RRG-inspired TWG derivation is shown in Fig. 1. This example involves substitutions of the arguments *John* and *Mary* into the *hates* tree and of *Bill* into *claims*, sister adjunction of *definitely* into *hates* (sister adjunction simply adds a new daughter to a node where

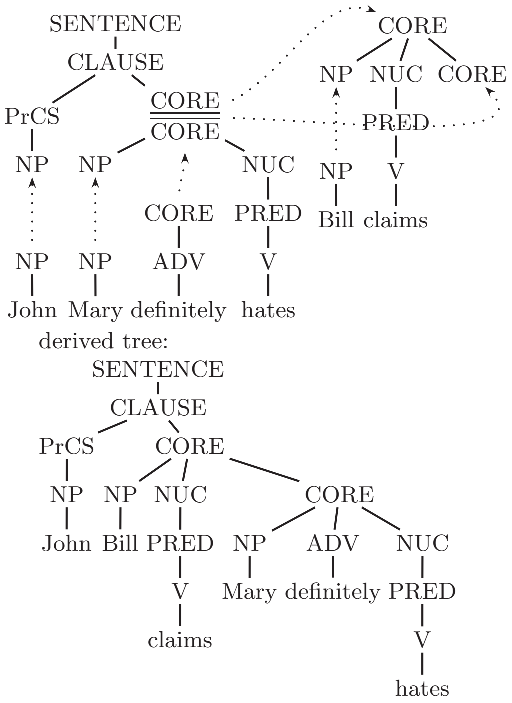


Fig. 1. RRG-style TWG derivation for *John Bill claims Mary definitely hates*

the root of the adjoining tree has to match the target node), and wrapping substitution of the *hates* tree around the *claims* tree.

It was shown in [6] that, in contrast to TAG, the TWG operations enable us to add even sentential arguments with long-distance extractions by a substitution operation. In this, TWG is close to other formalisms in the context of TAG-related grammar research that have been proposed in order to obtain derivation structures that reflect dependencies in a more appropriate way than it is done by TAG [10, 11, 1].

The focus of this paper is on the formal properties of TWG. After an introduction to TWG and in particular to k-TWG, a restricted form of TWG, we show how to construct an equivalent simple context-free tree grammar (CFTG) for a given k-TWG.

2 Tree Wrapping Grammar

The following introduction to TWG is largely taken from [6], except for the declarative definition of the notion of k-TWG, based on properties of the derivation trees decorated with wrapping substitution markings.

Borrowing from the TAG terminology, trees that can be added by substitution are called *initial* trees. In addition, we need *adjunct trees* for modeling modifiers and functional elements in RRG. These trees are added by sister adjunction. We distinguish **l**eft-adjoining, **r**ight-adjoining and unrestricted adjunct trees, resp. called *l-adjunct*, *r-adjunct* and *d-adjunct* trees. (The latter can add a **d**aughter at any position.)

Definition 1 (Tree Wrapping Grammar). *A* Tree Wrapping Grammar *(TWG) is a tuple* $G = \langle N, T, I, A_D, A_L, A_R, C\rangle$ *where*

a) N, T *are disjoint alphabets of non-terminal and terminal symbols.*

b) I, A_D, A_L *and* A_R *are disjoint finite sets of ordered labeled trees such that*

- *each non-leaf in a tree is labeled by some element from* $N \cup N^2$*,*
- *there is at most one node with a label from* N^2*,*
- *leaves have labels from* $N \cup T$*, and*
- *the root of each tree in* $A_D \cup A_L \cup A_R$ *has exactly one daughter.*

c) $C \subseteq N$*.*

A non-terminal leaf is called a substitution node*, and the labels from* N^2 *are called* split categories*.*

Every tree in I *is called an* initial *tree, every tree in* $A_D \cup A_L \cup A_R$ *an* adjunct *tree and every tree in* $I \cup A_D \cup A_L \cup A_R$ *an* elementary tree*.*

As we will see later, C is the set of non-terminals that can occur on a *wrapping spine* (i.e., between root and substitution site of the target tree of a wrapping substitution).

There are two TWG composition operations (see Fig. 2):

1. *Standard/Wrapping substitution*: a substitution node v in a tree γ gets replaced with a subtree α' of an initial tree α. If $\alpha' \neq \alpha$, then the root node v' of α' must be labeled with a split category $\langle X, Y\rangle$ such that the root of γ is labeled X and v is labeled Y. α is then split at v' and wraps around γ, i.e., the upper part of α ends up above the root of γ while α' fills the substitution slot. In this case, we call the operation a *wrapping substitution*. Otherwise ($\alpha = \alpha'$), we have a *standard substitution* and the root of α (i.e., v') must have the same label as v.
2. *Sister adjunction*: an adjunct tree β with root category X is added to a node v of γ with label X. The root r_β of β is identified with v and the (unique) daughter of r_β is added as a new daughter to v. Furthermore, if $\beta \in A_L$ (resp. $\beta \in A_R$), then the new daughter must be a leftmost (resp. rightmost) daughter.

A slightly different form of tree wrapping is proposed in [9] for RRG, leading to a flatter structure. One can consider a split node as a very special dominance edge (with specific constraints on how to fill it). In our definition, we would then have for a split node with categories X, Y a dominance edge between a node labeled X and a node labeled Y such that the X-node does not have any other daughters. In the flatter version of wrapping ([9]), the X-node can have other daughters that end up being sisters of the target tree of the wrapping that fills this dominance edge (see Fig. 3).

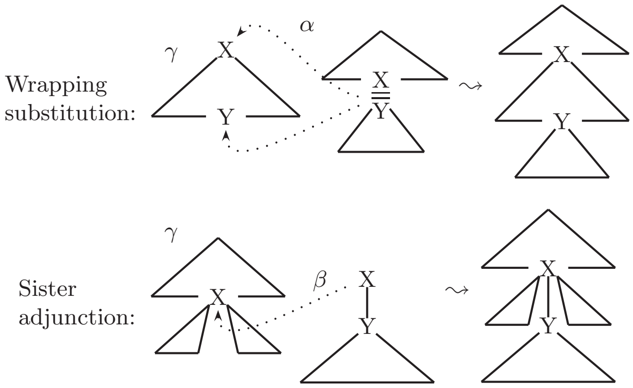


Fig. 2. Operations in TWG

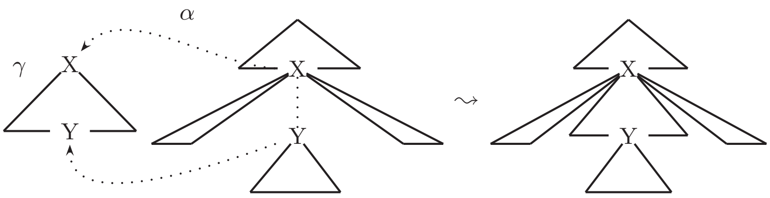


Fig. 3. Wrapping from [9]

It is easy to see that this form of wrapping can be transformed into the one used in this paper, simply by replacing the dominance edge with an immediate dominance edge and splitting the lower node with top and bottom categories X and Y respecitvely. As a result, we obtain trees that are slightly less flat than the ones from [9] and that, if we keep track of which edges have been added in this transformation, can be easily transformed back to the original flatter form. Therefore, without losing anything concerning the desired linguistic structures, for formal properties and parsing considerations we can work with the tree wrapping definition presented here.

Every elementary tree in a TWG G is a derived tree wrt. G, and every tree we can obtain with our composition operations from derived trees in G is again a derived tree. Wrapping substitutions require that, in the target tree, all categories on the path from the root to the substitution node (the *wrapping spine*) are in C. A further constraint is that wrapping substitution can target only initial trees, i.e., we cannot wrap a tree around an adjunct tree. Note that, in contrast to [6], we do not impose a limit on the number of wrapping substitutions stretching across a node in our definition of a TWG derived trees. This constraint comes later with the definition of k-TWG.

So far, wrapping can occur several times at the same elementary tree. Or, to put it differently, a node can be on several wrapping spines that are not nested.

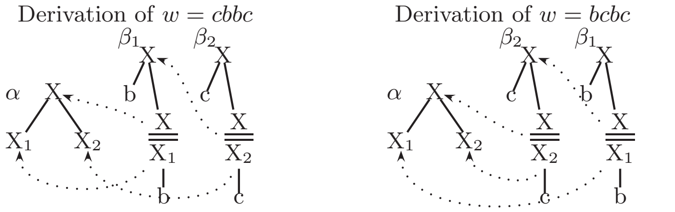


Fig. 4. Sample derivations: wrapping substitutions at sister nodes

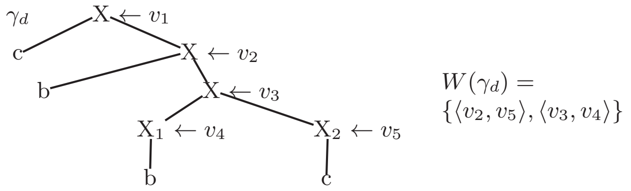


Fig. 5. Decorated derived tree arising from the first derivation in Fig. 4

See Fig. 4 for an example. In the two derivations, the root node of α is part of the two wrapping spines. In other words, both wrappings stretch across the root node of α. This has implications for the generative capacity and also for the parsing complexity.

We now want to restrict the wrapping substitutions in a derivation concerning the number of times a node can be part of a wrapping spine. To this end, we first introduce some decoration for the derived trees in the TWG given a specific derivation: Let γ_d be a tree derived in a TWG with a fixed derivation (there can be several derivations per derived tree and, consequently, several decorations). We define the wrapping decoration of γ_d as the following set of node pairs $W(\gamma_d)$: In every wrapping substitution step of the derivation in question with r and v being the root node r and the substitution node v of the target of the wrapping substitution, $\langle r, v\rangle \in W(\gamma_d)$. Nothing else is in $W(\gamma_d)$. We call every $v_\bot$ such that there exists a $v_\top$ with $\langle v_\top, v_\bot\rangle \in W(\gamma_d)$ a $\bot$ *node* in γ_d. We call a derived tree with such a decoration a *decorated* derived tree. An example is given in Fig. 5.

Once we have that, we can identify for a node v in such a decorated derived tree γ all wrapping substitution sites ($\bot$ nodes) where the wrapping substitution stretches across v.

Definition 2 (Gap set, wrapping degree). *Let $\gamma = \langle V, E, \prec, r, l\rangle$ be a decorated TWG derived tree with decoration $W(\gamma)$, and let $v \in V$.*

1. A set $V_\bot \subset V$ of $\bot$ nodes is a gap set *with respect to v if*

a) for every pair $\langle v_\top, v_\bot\rangle \in W(\gamma)$ with $v_\bot \in V_\bot$, it holds that $v_\top$ dominates v and v strictly dominates $v_\bot$, and

b) for every pair $\langle v_\top, v_\bot\rangle \in W(\gamma)$ with $v_\bot \in V_\bot$, there is no pair $\langle v'_\top, v'_\bot\rangle \in W(\gamma)$, $\langle v'_\top, v'_\bot\rangle \neq \langle v_\top, v_\bot\rangle$ with $v_\top$ dominating $v'_\top$, $v'_\top$ dominating v, v strictly dominating $v'_\bot$, and $v'_\bot$ dominating $v_\bot$.

2. *We then define the* wrapping degree *of v as the cardinality of its gap set.*
3. *The* wrapping degree *of a decorated derived tree is the maximal wrapping degree of its nodes, and the* wrapping degree *of a derived tree γ_d in the TWG G is the minimal wrapping degree of any decorated derived tree with respect to G that yields γ_d.*

In the example in Fig. 5, we have for instance a gap set $\{v_4, v_5\}$ for the node v_3. The wrapping degree of this derived tree is 2.

Now we can define the tree language for a TWG in general and in the case where wrapping degrees are limited by a $k \geq 0$:

Definition 3 (Language of a TWG). *Let G be a TWG.*

- *A* saturated *derived tree is a derived tree without substitution nodes and without split categories.*
- *The* tree language *of G is $L_T(G) = \{\gamma \,|\, \gamma$ is a saturated derived initial tree in $G\}$.*
- *The* string language *of G is the set of yields of trees in $L_T(G)$.*
- *The* k-tree language *of G is $L_T^k(G) = \{\gamma \,|\, \gamma$ is a saturated derived initial tree in G with a wrapping degree $\leq k\}$.*
- *The* k-string language *of G is the set of yields of trees in $L_T^k(G)$.*

Some TWGs are such that the maximal wrapping degree is limited, given the form of the elementary trees. But this is not always the case. In the following, we call a TWG a k-TWG if we impose k as a limit for the wrapping degree of derived trees, i.e., for a k-TWG, we consider the k-tree language of the grammar as its tree language.

As an example, Fig. 6 gives a TWG for the copy language. Here, all substitution nodes must be filled by wrapping substitutions since there are no trees with root label A, and the nodes with the split categories are always the middle nodes. The grammar is such that only derived trees with a wrapping degree 1 are possible.

In order to facilitate the construction of an equivalent simple CFTG for a given k-TWG, we show the following normal form lemma:

Lemma 1. *For every k-TWG $G = \langle N, T, I, A_D, A_L, A_R, C\rangle$, there is exists a weakly equivalent k-TWG $G' = \langle N', T, I', \emptyset, \emptyset, \emptyset, C\rangle$, i.e., a k-TWG without adjunct trees.*

The construction idea is again rather simple. For every daughter position, we precompile the possibility to add something by sister adjunction in the following way: We start with $I_{temp} = I$ and $I' = \emptyset$ and we set $A_l = A_L \cup A_D$ and $A_r = A_R \cup A_D$.

1. For every adjunct tree β, we add a subscript l (r or d respectively) to the root label of β if β is in A_l (resp. in A_r or in A_D). The resulting tree is added to I_{temp}.

$G = \langle\{\text{S, A}\}, \{a, b\}, I, \emptyset, \emptyset, \{\text{S, A}\}\rangle$ with $I =$

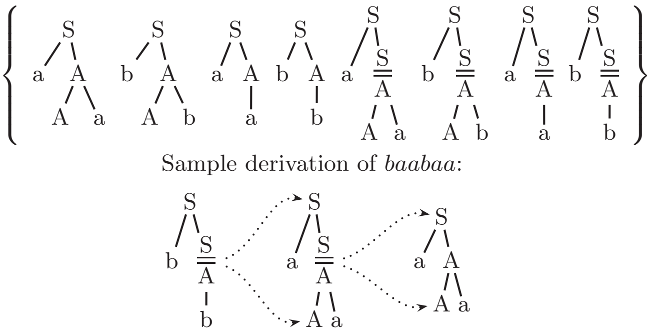


Fig. 6. TWG for the copy language $\{ww \mid w \in \{a, b\}^+\}$

2. For every $\gamma \in I_{temp}$: For every node v in γ that is not the root of a former adjunct tree: If v has i daughters, then we pick one combination $i_1, \ldots, i_k$ ($k \geq 0$) with $0 \leq i_1 < \ldots < i_k \leq i$ of positions between daughters. We then add new daughters to v at all these positions, labeled with the non-terminal of v and a subscript l for position 0, r for position i and d otherwise. The result is added to I'. This is repeated until I' does not change any more, i.e., all possible combinations of daughter positions for the nodes in γ have been taken into account.
3. For every $\gamma \in I'$ that is a former adjunct tree: Add
   - a tree γ_l to I' that consists of γ with an additional leftmost daughter of the root having the same label as the root and a subscript l in case the subscript of the root is l, d otherwise.
   - a tree γ_r to I' that consists of γ with an additional rightmost daughter of the root having the same label as the root and a subscript r in case the subscript of the root is r, d otherwise.
   - a tree γ_{lr} to I' that consists of γ with two additional daughters of the root, one leftmost and one rightmost daughter such that these daughters have the same label as the root and the following subscripts: the leftmost has a subscript l in case the subscript of the root is l, d otherwise. The rightmost has a subscript r in case the subscript of the root is r, d otherwise.

An example of this construction can be found in Fig. 7.

The equivalence of the original grammar and the constructed one is obvious. The latter requires the same number of wrapping substitutions as the original one and has the same string language for the same k. But it generates different derived trees since in cases of multiple adjunctions between two nodes we obtain a binary structure.

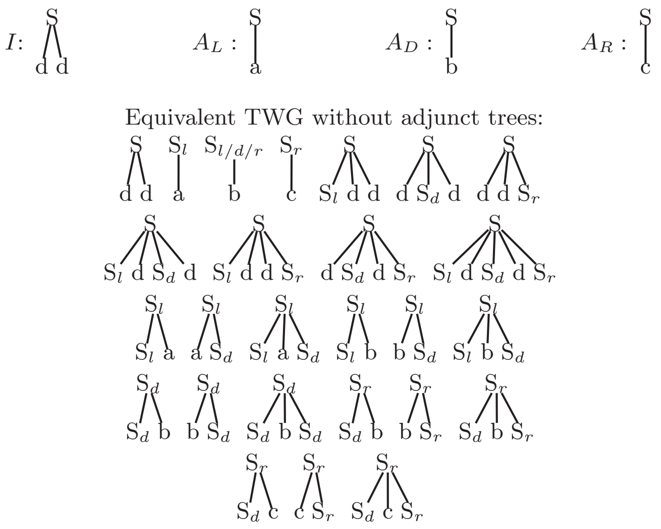


Fig. 7. Sample elimination of adjunct trees

3 Relation to context-free tree gramamrs

We now show that for every k-TWG one can construct an equivalent simple context-free tree grammar of rank k. This, in turn, is weakly equivalent to well-nested $(k+1)$-LCFRS (see [16, 13] for the definition of LCFRS and [7, 3] for well-nested LCFRS).

Without loss of generality, we assume the k-TWG to be without adjunct trees.

3.1 Context-free tree grammars

The following introduction to context-free tree grammars is taken from [8].

A ranked alphabet is a union $\Delta = \bigcup_{r \in \mathbb{N}} \Delta^{(r)}$ of disjoint sets of symbols. If $f \in \Delta^{(r)}$, r is the *rank* of f.

A tree over a ranked alphabet Δ is a labeled ordered tree where each node with n daughters is labeled by some $f \in \Delta^{(n)}$. We use the term representation of trees. The set $\mathcal{T}_\Delta$ of trees over Δ is defined as follows: 1. If $f \in \Delta^{(0)}$, then $f \in \mathcal{T}_\Delta$. 2. If $f \in \Delta^{(n)}$ and $t_1, \ldots, t_n \in \mathcal{T}_\Delta (n \geq 1)$, then $(f t_1 \ldots t_n) \in \mathcal{T}_\Delta$.

If Σ is an (unranked) alphabet and Δ a ranked alphabet ($\Sigma \cap \Delta = \emptyset$), let $\mathcal{T}_{\Sigma,\Delta}$ be the set of trees such that whenever a node is labeled by some $f \in \Delta$, then the number of its children is equal to the rank of f.

For a set $X = \{x_1, \ldots, x_n\}$ of variables, $\mathcal{T}_\Delta(X)$ denotes the set of trees over $\Delta \cup X$ where members of X all have rank 0. Such a tree t containing the variables X is often written $t[x_1, \ldots, x_n]$. If $t[x_1, \ldots, x_n] \in \mathcal{T}_\Delta(X)$ and $t_1, \ldots, t_n \in \mathcal{T}_\Delta$, then $t[t_1, \ldots, t_n]$ denotes the result of substituting $t_1, \ldots, t_n$ for $x_1, \ldots, x_n$, respectively, in $t[x_1, \ldots, x_n]$. An element $t[x_1, \ldots, x_n] \in \mathcal{T}_\Delta(X)$ is an n-context over Δ if for each $i = 1, \ldots, n$, x_i occurs exactly once in $t[x_1, \ldots, x_n]$.

Definition 4 (Context-free tree grammar). *A* context-free tree grammar *(CFTG) [12, 2] is a quadruple $G = \langle N, \Sigma, P, S\rangle$, where*

1. *N is a ranked alphabet of non-terminals,*
2. *Σ an unranked alphabet of terminals,*
3. *$S \in N$ is of rank 0, and*
4. *P is a finite set of productions of the form*

$$Ax_1 \ldots x_n \to t[x_1, \ldots, x_n]$$

where $A \in N^{(n)}$ and $t[x_1, \ldots, x_n] \in \mathcal{T}_{\Sigma,N}(\{x_1, \ldots, x_n\})$.
The rank *of G is* $\max\{r \mid N^{(r)} \neq \emptyset\}$.

For every $s, s' \in \mathcal{T}_{\Sigma,N}$, $s \Rightarrow_G s'$ is defined to hold if and only if there is a 1-context $c[x_1] \in \mathcal{T}_{\Sigma,N}(\{x_1\})$, a production $Bx_1 ... x_n \to t[x_1, ..., x_n]$ in P, and trees $t_1, ..., t_n \in \mathcal{T}_{\Sigma,N}$ such that $s = c[Bt_1 \ldots t_n]$, $s' = c[t[t_1, \ldots, t_n]]$.

The relation $\Rightarrow_G^*$ is defined as the reflexive transitive closure of$\Rightarrow_G$. The tree language $L(G)$ generated by a CFTG G is defined as $\{t \in \mathcal{T}_\Sigma \mid S \Rightarrow_G^* t\}$. The string language is the set of yields of the trees in $L(G)$.

A CFTG is said to be simple if all right-hand sides of productions in the grammar are n-contexts, in other words, they contain exactly one occurrence of each of their n variables.

CFTG for $\{w^3 \mid w \in \{a, b\}^+\}$:

$N^0 = \{\overline{S}\}, N^{(3)} = \{\overline{X}\}$, $\Sigma = \{a, b, A\}$, S the start symbol.
P contains the following productions:
$\overline{S} \to \overline{X}aaa \mid \overline{X}bbb$
$\overline{X}x_1x_2x_2 \to \overline{X}(Aax_1)(Aa_2)(Aax_3) \mid \overline{X}(Abx_1)(Abx_2)(Abx_3) \mid Ax_1x_2x_3$

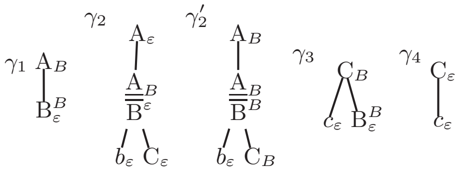
Sample derivation for the string *abaabaaba*:
$\overline{S} \Rightarrow \overline{X}aaa \Rightarrow \overline{X}(Aba)(Aba)(Aba)$
$\Rightarrow \overline{X}(Aa(Aba))(Aa(Aba))(Aa(Aba))$
$\Rightarrow A(Aa(Aba))(Aa(Aba))(Aa(Aba))$

Fig. 8. Simple CFTG for the double copy language

3.2 *k*-TWG and simple CFTG

In the following, we will show that for each k-TWG, an equivalent simple context-free tree grammar of rank k can be constructed.

Let us explain the construction while going through the simple example in Fig. 9. The CFTG non-terminals have the form $[A, A_1 A_2 \ldots A_n]$ with $n \leq k$, TWG for $\{(bc)^n \mid n \geq 1\} \cup \{c\}$:



Equivalent simple CFTG:
$N^{(0)} = \{S, [A], [C]\}, N^{(1)} = \{[A, B], [C, B]\}$, start symbol S, productions:
$S \to [A]$, $S \to [C]$
γ_1: $[A, B]x_1 \to Ax_1$
γ_2: $[A] \to A([A, B](Bb[C]))$
γ_2': $[A, B]x_1 \to A([A, B](Bb([C, B]x_1)))$
γ_3: $[C, B]x_1 \to Ccx_1$
γ_4: $[C] \to Cc$

Fig. 9. Sample TWG and equivalent simple CFTG

$A \in N$ and $A_i \in N$ for $1 \leq i \leq n$ where the intuition is the A is the root category of the tree this nonterminal expands to and $A_1A_2 \ldots A_n$ are the categories of pending gaps from wrappings that stretch across this tree. In other words, in the final decorated derived tree, they are the categories of the gap set nodes of this root, in linear order. Note that, since we assume a k-TWG, there cannot be more than k such categories. The gap trees that are to be inserted in the gap nodes (which are substitution nodes) are the arguments of this non-terminal. In other words, such a non-terminal tells us that we have to find an A-tree and there are pending lower parts of split trees with categories $A_1, \ldots, A_n$, which have to be inserted into that A-tree. For instance, the category $[A, B]$ in our example expands to A-trees that need a B-substitution node at some point (maybe after some further substitution), in order to insert the B-gap tree to which this category applies. The γ_1-rule in the TWG for instance encodes that one way to find such a tree is to create an A-node with a single daughter, where this daughter is the pending B-tree.

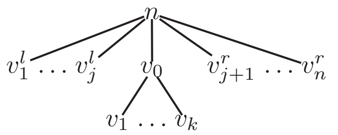
The construction does not go directly from an elementary TWG tree to a single production. Instead, it yields a single production for each possible decoration of the TWG tree with category sequences corresponding to possible gap set node labels in linear order that can arise within a derivation. An example where we have more than one possibility for a single TWG tree is the tree γ_2 in Fig. 9 where the γ_2 and γ_2' indicate the two cases. Accordingly, there are two productions. One $[A]$-production where $[A]$ is of rank 0. This is the case where nothing else is wrapped around γ_2 and, consequently, there is no pending gap at its root node. The second production (the γ_2' case) is the possibility to have something wrapped around the γ_2 tree. In this case, the gap category of the outer tree is B, and this gap must be placed somewhere below the C node, hence the non-terminal $[C, B]$ with the pending gap as argument for this node.

In order to keep track of these sequences of gap labels, we first define possible mappings f_1, f_2 for every elementary tree γ that assign to every node x in γ either a single sequence $f_1(x) = f_2(x)$ of non-terminals (= gap node labels) or, if the node is a split node or a substitution node that is used for a wrapping substitution, a pair of two possibly different such subsequences $\langle f_1(x), f_2(x)\rangle$. Intuitively, a split node starts a new gap, which is then filled by the lower part of the split node. Any gaps in the tree below the gap are accessible at the mother node of the split node.

Fig. 9 gives the assignments f_1 and f_2 for each node as a super- and a subscript. In cases where $f_1 = f_2$, there is just one subscript, while for $f_1 \neq f_2$ (split nodes and wrapping substitution nodes), we have both. For γ_2, we have two possible assignments. The mapping of γ_1 tells us that this tree is used in a wrapping configuration where a split tree with some lower category B is wrapped around it. Furthermore, according to the B-node annotation, this leaf is filled by the wrapping substitution ($f_2 = \varepsilon$). The first assignment for γ_2 tells us that this tree is used without wrapping anything around it. At its split node, we wrap it around something that has to contain a B-gap ($f_1 = B$), which will be filled by the lower part of the split tree, therefore at that point, no more gaps are pending ($f_2 = \varepsilon$). In contrast to this, the second γ_2 is used in a wrapping configuration where a split tree with some lower category B is wrapped around it ($f_1 = f_2 = B$ at the root). The B gap arising from the split node in the middle is filled by the lower B node. However, the overall B gap is still pending, therefore we have $f_2 = B$ at the split node. This pending gap is not inserted at the substitution node (category C), instead, the information about the B-gap is passed ($f_1 = f_2 = B$). It can be inserted in γ_3. There the substitution node has $f_1 = B$ (which means that we need a B-substitution node to be filled with some pending B-tree) and $f_2 = \varepsilon$ (signifying that this is the substitution node we were looking for, no more pending gaps below).

The definition of these assignments is such that we guess the pending gap categories for the leaves, we guess whether a substitution node is used for wrapping, and for split nodes, we guess the pending gap categories that arise out of the tree that this node is wrapped around. The rest is calculated in a bottom-up way as follows:

- For a substitution node v with category A: either v is not used for a wrapping substitution and we have $f_1(v) = f_2(v) = A_1 \dots A_i$ $(0 \leq i)$ or v is used for a wrapping substitution and we have $f_1(v) = A$ and $f_2(v) = \varepsilon$.
- $f_2(v_0) = f_1(v_1) \dots f_1(v_j)$ for every node v_0 with $v_1, \dots v_j$ being all daughters of v_0 in linear precedence order such that none of the daughters is a split node.
- For every split node v_0 with top label X and bottom label Y with $v_1, \dots v_k$ being all daughters of v_0 in linear precedence order, n being the mother of v_0 and $v^l_1, \dots, v^l_j$ and $v^r_{j+1}, \dots, v^r_n$ being the sisters of v_0 to the left and right in linear precedence order:



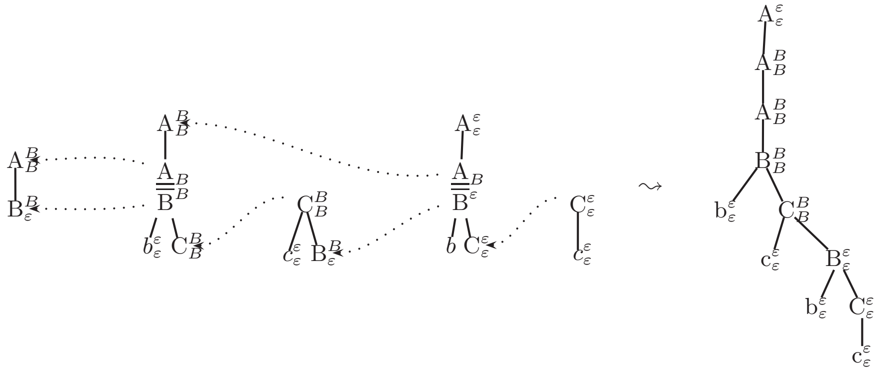
$f_2(v_0) = f_1(v_1) \dots f_1(v_k)$ and there are $B_1, \dots, B_j \in N$ such that
$f_1(v_0) = B_1 \dots B_i Y B_{i+1} \dots B_j$ and
$f_2(n) = f_1(v_1^l) \dots f_1(v_j^l) B_1 \dots B_i f_2(v_0) B_{i+1} \dots B_j f_1(v_{j+1}^r) \dots f_1(v_n^r)$.
We call the Y in this step the split category.

- For every node v that is neither a split node nor a non-terminal leaf, we have $f_1(v) = f_2(v)$.
- For every leaf v with a terminal label, we have $f_1(v) = f_2(v) = \varepsilon$.
- The length of the assigned sequences is limited to k.
- For every node v with a non-terminal label from $N \setminus C$ (C is the set of categories allowed on wrapping spines), it holds that $f_1 = f_2 = \varepsilon$.

Instead of using the original TWG, we can also use the trees with annotations f_1, f_2 in TWG derivations. For these derivations, let us make the following assumptions: The conditions for wrapping are that the f_1 value of the split node must be the f_1 value of the root of the target tree while the bottom category of the split node must be the f_1 of the target substitution node and the f_2 of this substitution node must be ε. The annotation of the root of the target tree remains while the annotation of the substitution node is the f_2 value of the split node. Furthermore, annotations of substitution nodes that are not used for wrapping have to be equal to the ones of the root of the tree that substitutes in.

An example of such a TWG derivation can be found in Fig. 10.

Sample derivations of $w = bcbc$:
TWG:


A_B^B
B_ε^B
A_B^B
A_B
B^B
$b_\varepsilon^\varepsilon$
C_B^B
C_B^B
$c_\varepsilon^\varepsilon$
B_ε^B
$\mathrm{A}_\varepsilon^\varepsilon$
A_B
B^ε
b
$\mathrm{C}_\varepsilon^\varepsilon$
$\mathrm{C}_\varepsilon^\varepsilon$
$c_\varepsilon^\varepsilon$
$\rightsquigarrow$
$\mathrm{A}_\varepsilon^\varepsilon$
A_B^B
A_B^B
B_B^B
$b_\varepsilon^\varepsilon$
C_B^B
$c_\varepsilon^\varepsilon$
$\mathrm{B}_\varepsilon^\varepsilon$
$b_\varepsilon^\varepsilon$
$\mathrm{C}_\varepsilon^\varepsilon$
$c_\varepsilon^\varepsilon$

Corresponding CFTG derivation:
$S \Rightarrow [A] \Rightarrow A([A,B](Bb[C])) \Rightarrow A([A,B](Bb(Cc)))$
$\Rightarrow A(A([A,B](Bb[C,B](Bb(Cc))))) \Rightarrow A(A(A(Bb[C,B](Bb(Cc)))))$
$\Rightarrow A(A(A(Bb(Cc(Bb(Cc))))))$

Fig. 10. Sample derivations in the grammars from Fig. 9

For these TWG derivations, the following lemma holds:

Lemma 2. *With this annotated TWG we obtain exactly the set of derived trees of the original TWG including for each node in a derived tree, obtained with a specific derivation, a decoration with the labels of the nodes from its gap set in linear precedence order.*

This lemma holds since all possible combinations of pending gaps below substitution nodes and to the left and right of split nodes are considered in the f_1, f_2 annotations. Furthermore, gaps are passed upwards. The only way to get rid of a gap in the f_1, f_2 value of a root node is to wrap a tree filling this gap around it.

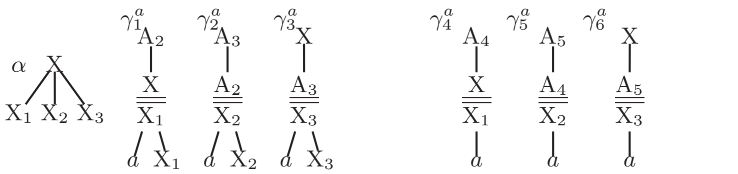
Given the gap assignment definition, we can now specify the set of productions in our CFTG that we obtain for each elementary tree.

1. For every non-terminal category X in our TWG, we add a production
$$S \rightarrow [X]$$
which is used for derived trees with root category X.
2. For every tree γ in the k-TWG with root r and root category A and for every assignment $f = \langle f_1, f_2 \rangle$ for γ as defined above, we have productions
$$[A, f_1(r)]y_1 \dots y_{|f_1(r)|} \rightarrow \tau(\gamma, f)$$
where $\tau(\beta, f)$ for any subtree β of a TWG tree with gap assignment f is defined as follows:
   - If β has only a single node v with non-terminal category B and $f_1(v) = f_2(v)$, then $\tau(\beta, f) = [B, f_1(v)]x_1 \dots x_{|f_1(v)|}$.[1]
   - If β has only a single node v with non-terminal category B and $f_1(v) = A \in N$, $f_2 = \varepsilon$, then $\tau(\beta, f) = x_1$.
   - If the root v of β is not a split node, its root category is A, and if $\beta_1 \dots, \beta_n$ are the daughter trees of v, then $\tau(\beta, f) = (A\tau(\beta_1, f) \dots \tau(\beta_n, f))$.
   - If the root v of β is a split node with top category A and bottom category B, and if $\beta_1 \dots, \beta_n$ are the daughter trees of v, then $\tau(\beta, f) = ([A, f_1(v)]x_1 \dots x_i(B\tau(\beta_1, f) \dots \tau(\beta_n, f))x_{i+1} \dots x_j)$. where $f_1(v) = A_1 \dots A_iBA_{i+1} \dots A_j$ and B is the split category from the construction of f.

   The variables $y_1, \dots, y_{|f_1(r)|}$ in the lefthand side of the production are exactly the ones from the righthand side in linear precedence order.
3. These are all the productions in the CFTG.

[1] We assume that fresh variables are used each time a new variable is needed.

TWG for the double copy language $\{w^3 \mid w \in \{a,b\}^+\}$:



(same for b and B_2, B_3) (same for b and B_4, B_5)

Equivalent simple CFTG:
Start symbol S, productions:

$S \to [X]$
γ_6^a: $[X] \to X([A_5, X_3](X_3 a))$
γ_5^a: $[A_5, X_3]x_1 \to A_5([A_4, X_2X_3](X_2 a)x_1)$
γ_4^a: $[A_4, X_2X_3]x_1x_2 \to A_4([X, X_1X_2X_3](X_1 a)x_1x_2)$
γ_3^a: $[X, X_1X_2X_3]x_1x_2x_3 \to X([A_3, X_1X_2X_3]x_1x_2(X_3 a x_3))$
γ_2^a: $[A_3, X_1X_2X_3]x_1x_2x_3 \to A_3([A_2, X_1X_2X_3]x_1(X_2 a x_2)x_3)$
γ_1^a: $[A_2, X_1X_2X_3]x_1x_2x_3 \to A_2([X, X_1X_2X_3](X_1 a x_1)x_2x_3)$
(same with b and B_2, B_3, B_4, B_5)
α: $[X, X_1X_2X_3]x_1x_2x_3 \to X x_1x_2x_3$

Fig. 11. Sample 3-TWG and equivalent simple CFTG

An example of this construction can be found in Fig. 9, and a sample derivation in the TWG and the corresponding CFTG is given in Fig. 10. The TWG derivation involves two wrappings of γ_2 around γ_1, the first (inner one) with an additional substitution of γ_3 into the C substitution node, the second, outer one with a substitution of γ_4 into this slot. The corresponding CFTG derivation starts by expanding $[A]$ to the tree corresponding to the outer wrapping of γ_2, with a non-terminal $[C]$ for γ_4. Inside the resulting tree, we have a non-terminal $[A,B]$ of rank 1 whose argument is the tree $Bb(Cc))$ which has to fill a B-gap. This is then expanded to a γ_2 tree that assumes that there is a B-gap below its C-substitution node. This second use of γ_2 creates again the request for an A-tree with a B-gap (non-terminal $[A,B]$), which is now filled by γ_1, and, below its substitution node, it needs a C-tree with a B-substitution node (non-terminal $[C,B]$) which can then be filled by the pending B-tree $Bb(Cc))$ from the outer use of γ_2. Such a tree is provided by γ_3.

As a further example consider the TWG and corresponding CFTG in Fig.11.

The crucial part of the construction is actually the definition of the f_1, f_2 gap category annotations. Once we have this, the following holds:

Lemma 3. *There is a derived tree γ in the TWG with pending gap category sequence annotations $f = \langle f_1, f_2\rangle$ (written $\langle \gamma, f\rangle$) as described above iff there is a corresponding derivation in the CFTG.*

Here, “corresponding derivation” means the following: if γ has gap sequence annotations f_1, f_2 and a root node v with node label A, then the correponding derivation is of the form $[A, f_1(v)]y_1 \dots y_{|f_1(v)|} \Rightarrow_G^* \tau(\gamma, f)$ where $\tau(\gamma, f)$ is as defined above in the construction for the case of elementary trees.

We can show this by an induction over the derivation structure, proving that

- The claim holds for elementary trees. This follows immediately from the construction.
- We assume that the claim holds for $\langle \gamma, f \rangle$ and $\langle \alpha, f_\alpha \rangle$ with corresponding CFTG derivations $[A_\gamma, \mathit{gaps}_\gamma]\boldsymbol{x}_\gamma \Rightarrow_G^* \tau(\gamma, f)$ and $[A_\alpha, \mathit{gaps}_\alpha]\boldsymbol{x}_\alpha \Rightarrow_G^* \tau(\alpha, f_\alpha)$.
  Then:
  $\langle \gamma', f' \rangle$ can be derived from $\langle \gamma, f \rangle$ in the TWG via substitution of $\langle \alpha, f_\alpha \rangle$ into one of the non-termial leaves
  $\Leftrightarrow$ this non-terminal leaf in $\langle \gamma', f' \rangle$ has category A_α and gap sequence gaps_α
  $\Leftrightarrow$ there is a corresponding non-terminal $[A_\alpha, \mathit{gaps}_\alpha]$ in $\tau(\gamma, f)$ that can be expanded using the derivation $[A_\alpha, \mathit{gaps}_\alpha]\boldsymbol{x} \Rightarrow_G^* \tau(\alpha, f_\alpha)$ (induction assumption)
  $\Leftrightarrow$ $[A_\gamma, \mathit{gaps}_\gamma]\boldsymbol{x} \Rightarrow_G^* \tau(\gamma', f')$ where, in this derivation, we have one part $[A_\gamma, \mathit{gaps}_\gamma]\boldsymbol{x}_\gamma \Rightarrow_G^* \tau(\gamma, f)$ and a second part consisting of an application of $[A_\alpha, \mathit{gaps}_\alpha]\boldsymbol{x}_\alpha \Rightarrow_G^* \tau(\alpha, f_\alpha)$.
- We assume that the claim holds for $\langle \gamma, f \rangle$ and $\langle \beta, f_\beta \rangle$ with corresponding CFTG derivations $[A_\gamma, \mathit{gaps}_\gamma]\boldsymbol{x}_\gamma \Rightarrow_G^* \tau(\gamma, f)$ and $[A_\beta, \mathit{gaps}_\beta]\boldsymbol{x}_\beta \Rightarrow_G^* \tau(\beta, f_\beta)$.
  Then:
  $\langle \gamma', f' \rangle$ is derived from $\langle \gamma, f_\gamma \rangle$ in the TWG by wrapping $\langle \beta, f_\beta \rangle$ around $\langle \gamma, f_\gamma \rangle$
  $\Leftrightarrow$ there is a split node in $\tau(\beta, f_\beta)$ with category $\langle A_\gamma, Y \rangle$ and with an f_1 value gaps_γ and there is a non-terminal leaf in $\langle \gamma, f_\gamma \rangle$ with category Y and with $f_1 = Y$ and $f_2 = \varepsilon$
  $\Leftrightarrow$ there is a non-terminal $[A_\gamma, \mathit{gaps}_\gamma]$ corresponding to the split node in $\tau(\beta, f_\beta)$ that can be expanded by the derivation $[A_\gamma, \mathit{gaps}_\gamma]\boldsymbol{x}_\gamma \Rightarrow_G^* \tau(\gamma, f)$
  $\Leftrightarrow$ there is a derivation $[A_\beta, \mathit{gaps}_\beta]\boldsymbol{x}_\beta \Rightarrow_G^* \tau(\gamma', f')$ in the CFTG consisting of $[A_\beta, \mathit{gaps}_\beta]\boldsymbol{x}_\beta \Rightarrow_G^* \tau(\beta, f_\beta)$ and then an application of $[A_\gamma, \mathit{gaps}_\gamma]\boldsymbol{x}_\gamma \Rightarrow_G^* \tau(\gamma, f)$.

With this lemma, we obtain the following theorem:

Theorem 1. *For every k-TWG there is an equivalent simple CFTG of rank k.*

As a consequence, we obtain that the languages of k-TWGs are in the class of well-nested linear context-free rewriting languages[2] and therefore mildly context-sensitive [16]. This term, introduced by [4], characterizes formalisms beyond CFG that can describe cross-serial dependencies, that are polynomially parsable and that generate languages of constant growth. Joshi's conjecture is that mildly context-sensitive grammar formalisms describe the appropriate grammar class for dealing with natural languages.

[2] Note that the fact that we can construct an equivalent well-nested LCFRS for a k-TWG does not mean that k-TWG (for some fixed k) cannot deal with ill-nested dependencies. The structures described by the LCFRS do not correspond to the dependency structures obtained from TWG derivations. The latter are determined only by the fillings of substitution slots.

4 Conclusion

We have shown that k-TWG is a mildly context-sensitive grammar formalism, more particular, it falls into the class of simple context-free tree languages of rank k/well-nested $(k+1)$-LCFRS. This is an interesting result, considering that TWG arose out of an attempt to formalize the syntactic inventory of RRG, a grammar theory that emerged from broad empirical linguistic studies. Therefore the formal results in this paper support in a convincing way Joshi's conjecture about the mild context-sensitivity of natural languages.

References

1. Chiang, D., Scheffler, T.: Flexible composition and delayed tree-locality. In: TAG+9 Proceedings of the Ninth International Workshop on Tree-Adjoining Grammar and Related Formalisms (TAG+9). pp. 17–24. Tübingen (June 2008)
2. Engelfriet, J., Schmidt, E.M.: IO and OI. Journal of Computer and System Sciences (15), 328–353 (1977)
3. Gómez-Rodríguez, C., Kuhlmann, M., Satta, G.: Efficient parsing of well-nested linear context-free rewriting systems. In: Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics. pp. 276–284. Association for Computational Linguistics, Los Angeles, California (June 2010), `http://www.aclweb.org/anthology/N10-1035`
4. Joshi, A.K.: Tree adjoining grammars: How much context-sensitivity is required to provide reasonable structural descriptions? In: Dowty, D., Karttunen, L., Zwicky, A. (eds.) Natural Language Parsing, pp. 206–250. Cambridge University Press (1985)
5. Joshi, A.K., Schabes, Y.: Tree-Adjoning Grammars. In: Rozenberg, G., Salomaa, A. (eds.) Handbook of Formal Languages, pp. 69–123. Springer, Berlin (1997)
6. Kallmeyer, L., Osswald, R., Van Valin, Jr., R.D.: Tree wrapping for Role and Reference Grammar. In: Morrill, G., Nederhof, M.J. (eds.) Formal Grammar 2012/2013. Lecture Notes in Computer Science, vol. 8036, pp. 175–190. Springer, Berlin, Heidelberg (2013)
7. Kanazawa, M.: The pumping lemma for well-nested Multiple Context-Free Languages. In: Diekert, V., Nowotka, D. (eds.) DLT 2009. LNCS, vol. 5583, pp. 312–325. Springer, Berlin Heidelberg (2009)
8. Kanazawa, M.: Multidimensional trees and a Chomsky-Schützenberger-Weir representation theorem for simple context-free tree grammars. Journal of Logic and Computation (Published online June 30, 2014)
9. Osswald, R., Kallmeyer, L.: Towards a formalization of Role and Reference Grammar. In: Proceedings of the 2013 Conference on Role and Reference Grammar (to appear)
10. Rambow, O., Vijay-Shanker, K., Weir, D.: D-Tree Grammars. In: Proceedings of ACL (1995)
11. Rambow, O., Vijay-Shanker, K., Weir, D.: D-Tree Substitution Grammars. Computational Linguistics (2001)
12. Rounds, W.C.: Mappings and grammars on trees. Mathematical Systems Theory (4), 257–287 (1970)
13. Seki, H., Matsumura, T., Fujii, M., Kasami, T.: On multiple context-free grammars. Theoretical Computer Science 88(2), 191–229 (1991)

14. Van Valin, Jr., R.D.: Exploring the Syntax-Semantics Interface. Cambridge University Press (2005)
15. Van Valin, Jr., R.D., Foley, W.A.: Role and reference grammar. In: Moravcsik, E.A., Wirth, J.R. (eds.) Current approaches to syntax, Syntax and semantics, vol. 13, pp. 329–352. Academic Press, New York (1980)
16. Vijay-Shanker, K., Weir, D.J., Joshi, A.K.: Characterizing structural descriptions produced by various grammatical formalisms. In: Proceedings of ACL. Stanford (1987)