A Declarative Characterization of Different Types of Multicomponent Tree Adjoining Grammars[†]

Laura Kallmeyer (lk@sfs.uni-tuebingen.de)
Collaborative Research Center 833, University of Tübingen, Nauklerstr. 35, D-72074 Tübingen, Germany.

Abstract. Multicomponent Tree Adjoining Grammars (MCTAGs) are a formalism that has been shown to be useful for many natural language applications. The definition of non-local MCTAG however is problematic since it refers to the process of the derivation itself: a simultaneity constraint must be respected concerning the way the members of the elementary tree sets are added. Looking only at the result of a derivation (i.e., the derived tree and the derivation tree), this simultaneity is no longer visible and therefore cannot be checked. I.e., this way of characterizing MCTAG does not allow to abstract away from the concrete order of derivation. In this paper, we propose an alternative definition of MCTAG that characterizes the trees in the tree language of an MCTAG via the properties of the derivation trees (in the underlying TAG) the MCTAG licences. We provide similar characterizations for various types of MCTAG. These characterizations give a better understanding of the formalisms, they allow a more systematic comparison of different types of MCTAG, and, furthermore, they can be exploited for parsing.

Keywords: Tree Adjoining Grammar, MCTAG, multicomponent rewriting, Simple Range Concatenation Grammar

1. Introduction

1.1. Tree Adjoining Grammars

Tree Adjoining Grammar (TAG, see Joshi and Schabes, 1997) is a tree-rewriting formalism. A TAG consists of a finite set of trees (elementary trees). The nodes of these trees are labelled with nonterminals and terminals (terminals only label leaf nodes). Starting from the elementary trees, larger trees are derived using composition operations of substitution (replacing a leaf with a new tree) and adjunction (replacing an internal node with a new tree). In case of an adjunction, the tree being adjoined has exactly one leaf that is marked as the foot node (marked with an asterisk). Such a tree is called an *auxiliary* tree. When adjoining such a tree to a node n, in the resulting tree, the subtree with root n from the old tree is attached to the foot node of the auxiliary tree. Elementary trees that are not auxiliary trees are

[†] Preprint version, to appear in *Research on Language and Computation*. The original publication will be available at www.springerlink.com.

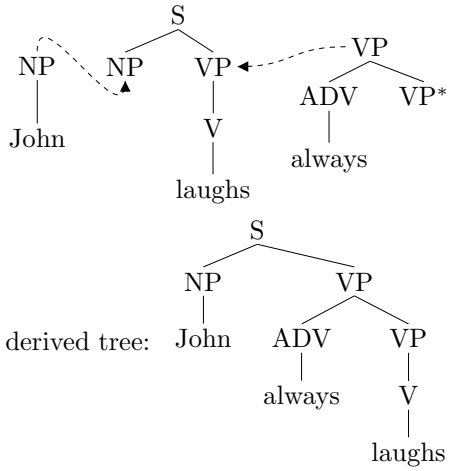


Figure 1. Sample TAG derivation

called *initial* trees. Additionally, TAG allows for each node to specify the set of auxiliary trees that can be adjoined and, furthermore, to specify whether adjunction is obligatory or not.

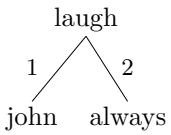
As an example, Fig. 1 shows the TAG derivation of (1).

(1) John always laughs

A derivation starts with an initial tree. In a final derived tree, all leaves must have terminal labels.

TAG derivations are represented by derivation trees that record the history of how the elementary trees are put together. A derived tree is the result of carrying out the substitutions and adjunctions. Each edge in a derivation tree stands for an adjunction or a substitution. The edges are labelled with Gorn addresses of the nodes where the substitutions/adjunctions take place.[1] E.g., the derivation tree (2) for Fig. 1 indicates that the elementary tree for *John* is substituted for the node at address 1 and *always* is adjoined at node address 2.

(2)



[1] The root has the address ε, and the jth child of the node with address p has address pj.

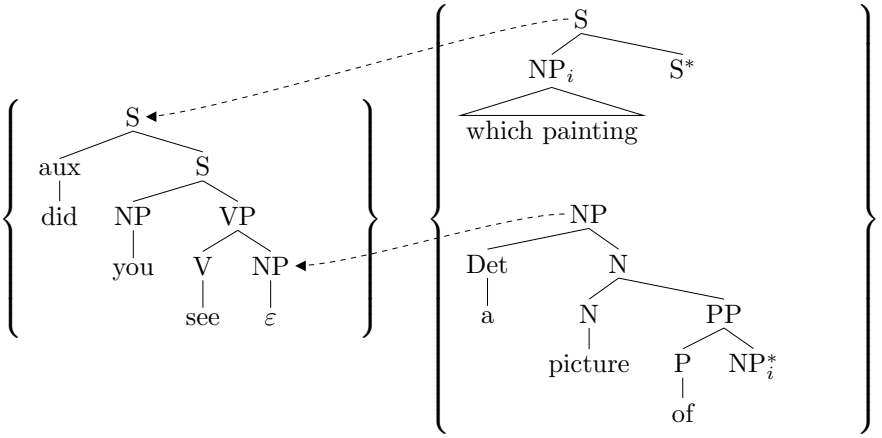


Figure 2. MCTAG elementary trees for extraction from NP

1.2. Multicomponent TAG

An extension of TAG that has been shown to be useful for several linguistic applications is multicomponent TAG (MCTAG, Joshi, 1987; Weir, 1988). An MCTAG contains sets of elementary trees. In each derivation step, one of the tree sets is chosen and all trees from the set are attached simultaneously to pairwise different nodes in the already derived tree. The very first definition of MCTAG goes back to (Joshi et al., 1975) where these grammars are called *Simultaneous TAG*.

Depending on the nodes to which the different trees from the set attach, different kinds of MCTAG are distinguished: if all nodes are required to be part of the same elementary tree, the MCTAG is called *tree-local*, if all nodes are required to be part of the same tree set, the grammar is *set-local* and otherwise the grammar is *non-local*. These three types of MCTAG were introduced in (Joshi, 1987; Weir, 1988).

A derivation starts from an initial tree α being in a singleton set. In the final derived tree, all leaves must be labelled by terminals.

Tree-local MCTAG have for example been used for extraction out of complex noun phrases (Kroch and Joshi, 1987) as in the analysis of (3) in Fig. 2.

(3) which painting did you see a picture of

The term "MCTAG" without further specification is sometimes used as meaning "set-local MCTAG" (Weir, 1988) and sometimes as a term for unrestricted (non-local) MCTAG (Becker et al., 1991). In this paper, the term "MCTAG" stands for "non-local MCTAG", i.e., MCTAG without any locality requirement.

More recently, various other types of MCTAG have been proposed, such as Vector MCTAG with Dominance Links (Rambow, 1994), tree-local MCTAG with node sharing (Kallmeyer, 2005), tree-local MCTAG with flexible composition (Joshi et al., 2007), k-delayed tree-local MCTAG (Chiang and Scheffler, 2008) and tree-local MCTAG with node sharing and tree tuples (Lichte, 2007; Kallmeyer and Parmentier, 2008). We will introduce these different MCTAG formalisms in section 4. Chen-Main and Joshi (2007) provide a good overview of various linguistic phenomena and the type of MCTAG they require.

In this paper we argue that the simultaneity requirement of the standard definition for non-local MCTAG is problematic and we propose an alternative declarative definition of MCTAG. Section 2 formally introduces TAG and MCTAG and section 3 gives the new MCTAG definition. Section 4 treats different local and non-local types of MCTAG and gives similar declarative definitions for them. Section 5 sketches a parsing strategy for MCTAG based on the declarative characterizations from sections 3 and 4. Section 6 concludes this article.

2. Definition of TAG

In the following, we give a graph-based definition of TAG that differs from the way TAG is mostly presented in the literature. This new definition of TAG is, in the first place, useful for the work presented here and, moreover, gives a good understanding of the formal properties of TAG.

In order to give a formal definition of TAG, we first need some preliminary definitions about trees:

DEFINITION 1 (Tree).

1. *A* finite tree *is a directed graph* $\gamma = \langle V, E, r\rangle$ *such that*
   - γ *contains no cycles,*
   - *only the root* $r \in V$ *has in-degree 0,*
   - *every vertex* $v \in V$ *is accessible from* r*, and*
   - *all nodes* $v \in V - \{r\}$ *have in-degree 1, i.e., there is a unique* u *with* $\langle u, v\rangle \in E$*.*

   E^+ *is the transitive closure and* E^* *the reflexive transitive closure of* E*.* E^* *is called the* dominance *relation.*

   A vertex with out-degree 0 *is called a* leaf*. The vertices in a tree are also called* nodes*.*

2. *A tree is* ordered *if it has an additional* linear precedence *relation* $\prec \in V \times V$ *such that*

   - $\prec$ *is irreflexive, antisymmetric and transitive,*
   - *for all* v_1, v_2 *with* $\{\langle v_1, v_2\rangle, \langle v_2, v_1\rangle\} \cap E^* = \emptyset$*: either* $v_1 \prec v_2$ *or* $v_2 \prec v_1$*,*
   - *for all* $v_1, v_2 \in V$*: if there is either a* $\langle v_3, v_1\rangle \in E$ *with* $v_3 \prec v_2$ *or a* $\langle v_4, v_2\rangle \in E$ *with* $v_1 \prec v_4$*, then* $v_1 \prec v_2$*, and*
   - *nothing else is in* $\prec$*.*

In this paper, we are only concerned with finite trees and therefore, we use the term "tree" as denoting "finite tree".

In the case of TAG, the derived trees and also the derivation trees are labeled trees. Derived trees have nodes labeled with non-terminals or terminals, while derivation trees have nodes labeled with names of elementary trees and edges labeled with node adresses. The node labeling is given by a function l and the edge labeling by a function g.

DEFINITION 2 (Labeling). *A* labeling *of a graph* $\gamma = \langle V, E\rangle$ *over a signature* $\langle A_1, A_2\rangle$ *is a pair of functions* $l : V \to A_1$ *and* $g : E \to A_2$ *with* A_1, A_2 *being disjoint alphabets.*

For TAG we need, specifically, the definition of initial and auxiliary trees. In the following, we assume an alphabet N of non-terminal symbols and an alphabet T of terminal symbols.

DEFINITION 3 (Auxiliary and initial trees).

1. *A* syntactic tree *is an ordered labeled tree such that* $l(v) \in N$ *for each vertex* v *with out-degree at least 1 and* $l(v) \in (N \cup T \cup \{\varepsilon\})$ *for each leaf* v*.*

2. *An* auxiliary tree *is a syntactic tree* $\langle V, E, r\rangle$ *such that there is a unique leaf* f *marked as foot node with* $l(r) = l(f)$*. We write this tree as* $\langle V, E, r, f\rangle$*.*

3. *An* initial tree *is a non-auxiliary syntactic tree.*

Now we can introduce TAG.

DEFINITION 4 (Tree Adjoining Grammar).
A Tree Adjoining Grammar *(TAG) is a tuple* $G = \langle N, T, S, I, A, f_{OA}, f_{SA}\rangle$ *where*

– N, T *are disjoint alphabets of non-terminal and terminal symbols,*

– $S \in N$ *is the start symbol,*

– I *is a finite set of initial trees, and* A *a finite set of auxiliary trees.*

– $f_{OA} : \{v \mid v$ *is a node in some* $\gamma \in I \cup A\} \to \{0, 1\}$ *and* $f_{SA} : \{v \mid v$ *is a node in some* $\gamma \in I \cup A\} \to P(A)$ *where* $P(A)$ *is the set of subsets of* A *are functions such that* $f_{OA}(v) = 0$ *and* $f_{SA}(v) = \emptyset$ *for every leaf* v.

Every tree in $I \cup A$ *is called an* elementary tree.

For a given node, the function f_{OA} specifies whether adjunction is obligatory (value 1) or not (value 0) and f_{SA} gives the set of auxiliary trees that can be adjoined. Only internal nodes can allow for adjunction, adjunction at leaves is not possible.[2] As a notational convention, we often omit the function f_{OA} and f_{SA} in the tuple notation, i.e., we write TAGs as quintuples $\langle N, T, S, I, A\rangle$.

In TAG, larger trees are derived from $I \cup A$ by subsequent applications of the operations substitution and adjunction. The substitution operation combines a syntactic tree and an initial tree into a new syntactic tree while adjunction combines a syntactic tree and an auxiliary tree into a new syntactic tree.

DEFINITION 5 (Substitution).
Let $\gamma = \langle V, E, r\rangle$ *be a syntactic tree,* $\gamma' = \langle V', E', r'\rangle$ *an initial tree and* $v \in V$. $\gamma[v, \gamma']$, *the result of* substituting γ' *into* γ *at node* v *is defined as follows:*

– *if* v *is not a leaf or* v *is a foot node or* $l(v) \neq l(r')$, *then* $\gamma[v, \gamma']$ *is undefined;*

– *otherwise,* $\gamma[v, \gamma'] = \langle V'', E'', r''\rangle$ *with* $V'' = V \cup V' \setminus \{v\}$ *and* $E'' = (E \setminus \{\langle v_1, v_2\rangle \mid v_2 = v\}) \cup E' \cup \{\langle v_1, r'\rangle \mid \langle v_1, v\rangle \in E\}$.

A leaf that has a non-terminal label and that is no foot node is called a substitution node.

DEFINITION 6 (Adjunction).
Let $\gamma = \langle V, E, r\rangle$ *be a syntactic tree,* $\gamma' = \langle V', E', r', f\rangle$ *an auxiliary tree and* $v \in V$. $\gamma[v, \gamma']$, *the result of* adjoining γ' *into* γ *at node* v *is defined as follows:*

[2] We adopt the standard assumption that adjunction at foot nodes is not allowed.

– *if* $l(v) \neq l(r')$, *then* $\gamma[v, \gamma']$ *is undefined;*

– *otherwise,* $\gamma[v, \gamma'] = \langle V'', E'', r'' \rangle$ *with* $V'' = V \cup V' \setminus \{v\}$ *and* $E'' = (E \setminus \{\langle v_1, v_2 \rangle \mid v_1 = v \text{ or } v_2 = v\}) \cup E' \cup \{\langle v_1, r' \rangle \mid \langle v_1, v \rangle \in E\} \cup \{\langle f, v_2 \rangle \mid \langle v, v_2 \rangle \in E\}$.

Note that we define adjunctions and substitutions at a node v in a tree γ not with respect to the address of v in γ as usually done in TAG (see for example Vijay-Shanker and Weir, 1994). This way, we avoid the problem of updating node addresses after each adjunction or substitution.

Now we can introduce derived trees together with their corresponding derivation trees. Derived trees are syntactic trees while derivation trees are unordered trees.[3]

In the following, we call a tree an instance of a tree γ if it is isomorphic to γ while preserving the labeling. Furthermore, we say that two trees are disjoint if their node sets are disjoint.

DEFINITION 7 (Derived tree and derivation tree).
Let $G = \langle N, T, S, I, A \rangle$ *be a TAG.*

1. *Every instance* γ *of a* $\gamma_e \in I \cup A$ *is a derived tree in* G.[4]
   The corresponding derivation tree is $\langle \{v\}, \emptyset, v \rangle$ *with* $l(v) = \gamma_e$.[5]

2. *For pairwise disjoint* $\gamma_1, \ldots, \gamma_n, \gamma$ *such that* $\gamma_1, \ldots, \gamma_n$ *are derived trees in* G *with derivation trees* $D_i = \langle V_i, E_i, r_i \rangle$ $(1 \leq i \leq n)$ *and* $\gamma = \langle V, E, r \rangle$ *is an instance of a* $\gamma_e \in I \cup A$ *such that* $v_1, \ldots, v_n \in V$ *are pairwise different with Gorn addresses* $p_1, \ldots, p_n$*: if*

   – $\gamma' = \gamma[v_1, \gamma_1] \ldots [v_n, \gamma_n]$ *is defined and*
   – $l(r_i) \in f_{SA}(v_i)$ *for all* $\gamma_i \in A$,

[3] According to our definitions of substitution and adjunction, once an adjunction or substitution is performed on a node, the node disappears (this is the general assumption in standard TAG). This means that multiple adjunctions are not possible. Consequently, the order of adjunctions and substitutions on a tree does not matter for the resulting derived tree.

[4] We use instances of elementary trees to avoid using twice exactly the same tree, i.e., the same set of nodes and edges, in a derivation. Instead, the tree instances involved in one derivation must be pairwise disjoint.

[5] While in this definition the labels of nodes in derivation trees are elementary trees, when depicting a derivation tree, we usually use names of elementary trees as node labels. These names serve only to avoid drawing the entire tree as label. Formally, they are not needed, i.e, whenever we talk about an elementary tree γ, this γ denotes the tree, not its name.

then γ' is a derived tree in G with a corresponding derivation tree $D = \langle V_D, E_D, r_D\rangle$ such that $V_D = \bigcup_{i=1}^{n} V_i \cup \{r_D\}$ where $r_D \notin \bigcup_{i=1}^{n} V_i$ is a new node, $E_D = \bigcup_{i=1}^{n} E_i \cup \{\langle r_D, r_1\rangle, \ldots, \langle r_D, r_n\rangle\}$ and $l(r_D) = \gamma_e$ and $g(\langle r_D, r_i\rangle) = p_i$ for $1 \leq i \leq n$.

3. These are all derived trees and derivation trees in G.

We call a derived tree that does not contain substitution nodes or nodes v with $f_{OA}(v) = 1$ a saturated *derived tree and the corresponding derivation tree a* saturated *derivation tree.*

Note that we have not defined the process of TAG derivation here. A derivation tree describes the set of adjunctions and substitutions that are performed in a derivation. However, it does not specify the order in which these adjunctions have to be performed. We could traverse a derivation tree in any order, e.g., start from the root and proceed top-down or compute first the derived trees of the daughter nodes of the root and then attach them to the root elementary tree.

TAG derivation trees are important structures. They describe uniquely the derived tree and they are often the output of parsing. Furthermore, they are usually used as the central structure of the syntax-semantics interface, i.e., TAG semantics is computed on the derivation tree (Kallmeyer and Joshi, 2003; Nesson and Shieber, 2006; Kallmeyer and Romero, 2008). Derivation trees are context-free since the set of derivation trees of a given TAG can be described by a CFG.

DEFINITION 8 (Tree language).
Let $G = \langle N, T, S, I, A\rangle$ be a TAG. The tree language *of G is $L_T(G) = \{\gamma \mid \gamma$ is a saturated derived initial tree in G with root label $S\}$. The* string language *of G is the set of yields of trees in $L_T(G)$.*

In an MCTAG, the elementary trees are grouped into elementary tree sets:

DEFINITION 9 (MCTAG).
A multicomponent TAG *(MCTAG) is a tuple $G = \langle N, T, S, I, A, \mathcal{A}\rangle$ where $G_{TAG} = \langle N, T, S, I, A\rangle$ is a TAG with elementary trees $I \cup A$, and $\mathcal{A}$ is a partition of $I \cup A$. $\mathcal{A}$ is called the set of elementary tree sets.*[6]

[6] Note that this definition does not exclude that the same tree occurs in different sets or even several times in the same set. In this case, we consider that there are different trees that look exactly the same (i.e., that are isomorphic while having identical labels).

Since we assume that every elementary tree occurs in exactly one of the elementary tree sets, we can uniquely determine the tree set a given tree belongs to. This is crucial for several of the definitions we give for different MCTAG variants and also for the way the conditions imposed for these MCTAG variants can be checked during parsing. We formulate for instance the condition that whenever a tree from an elementary tree set is used, all trees from this tree set must be used by imposing that every two elementary trees γ_1, γ_2 belonging to the same tree set must be used the same number of times. Such a formalization of this condition is only possible if elementary trees belong to unique tree sets. A disadvantage is of course that in cases where isomorphic trees appear in different sets, we cannot capture this generalization. Allowing a tree to appear in different sets would lead to a more compact representation of the grammar.

For non-local MCTAG we require that in each derivation step, a new instance of an elementary tree set is chosen and all elements from this set are adjoined/substituted simultaneously to the already derived tree:

DEFINITION 10 (MCTAG derivation).
Let $G = \langle N, T, S, I, A, \mathcal{A} \rangle$ be an MCTAG.

1. *Every instance γ of an elementary tree $\gamma_e \in I \cup A$ such that $\{\gamma_e\} \in \mathcal{A}$ is a derived tree in G.*

2. *For pairwise disjoint $\gamma_1, \ldots, \gamma_n, \gamma$ such that γ is a derived tree in G and $\gamma_1, \ldots, \gamma_n$ are instances of elementary trees $\gamma'_1, \ldots, \gamma'_n$ with $\{\gamma'_1, \ldots, \gamma'_n\} \in \mathcal{A}$ and for pairwise different nodes $v_1, \ldots, v_n$ in γ: if $\gamma' = \gamma[v_1, \gamma_1] \cdots [v_n, \gamma_n]$ is defined and if $\gamma'_i \in f_{SA}(v_i)$ for all v_i with out-degree at least 1 $(1 \leq i \leq n)$, then $\gamma \Rightarrow \gamma'$ and γ' is a derived tree in G.*

   $\overset{*}{\Rightarrow}$ is the reflexive transitive closure of $\Rightarrow$.

We call the sequence of adjunctions $\gamma[v_1, \gamma_1] \cdots [v_n, \gamma_n]$ corresponding to a sequence of derivation steps $\gamma \overset{*}{\Rightarrow} \gamma'$ a derivation.

In this definition, the elements from the new set instance are added one after the other. However, since they are added to pairwise different nodes in γ, the order does not matter and we consider them as being

Furthermore, this definition differs from the definition in (Weir, 1988) in the sense that Weir defines elementary tree sets as sequences of elementary trees. However, the usual practice in more recent MCTAG publications is a definition as sets, which was actually already adopted by the first introduction of MCTAG under the name of simultaneous TAG in (Joshi et al., 1975).

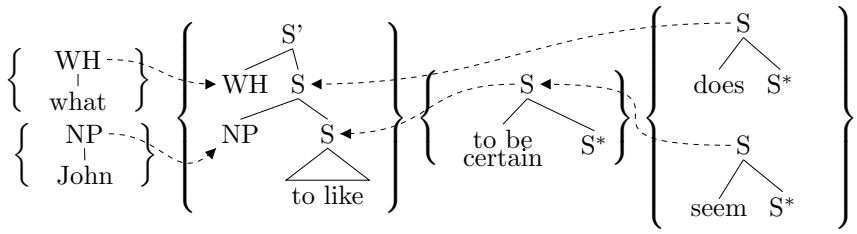


Figure 3. MCTAG derivation for (4)

added simultaneously. For this, it is crucial that none of the attachment sites belongs to any of the new elementary tree instances.

As in TAG, derived trees that do not contain substitution nodes or nodes v with $f_{OA}(v) = 1$ are called *saturated* derived trees and the tree language of a non-local MCTAG is the set of its saturated derived initial trees with root label S.

3. A declarative characterization of MCTAG

3.1. Motivation

To illustrate the idea of this paper let us consider the non-local MCTAG derivation of (4) in Fig. 3.

(4) what does John seem to be certain to like

Here, the tree for *to be certain* adjoins to the lower S node of the *like* tree, the WH and NP nodes of the *like* tree are substituted for *what* and *John* respectively, and the trees for *does* and *seem* are adjoined simultaneously to the upper S node of *like* and the root node of *to be certain* respectively. These last two operations cannot be performed before having added *to be certain* to *like*, otherwise the simultaneity requirement cannot be satisfied.

Intuitively, the requirement of attaching all elements of an elementary set instance simultaneously to pairwise different nodes in a derived tree is easy to understand and the definition of MCTAG based on this simultaneity seems clear. However, for the single adjunctions and substitutions performed in an MCTAG derivation, the simultaneity requirement imposes certain derivation orders even though a different order might lead to the same adjunctions and substitutions and to the same derived tree. E.g., in Fig. 3 one might as well start by adding *does* to *like* (at the higher S node), then adjoin *to be certain* to *like* (at the lower S node) and then adjoin *seem* to *to be certain*. This yields

the same derived tree, and the same adjunctions and substitutions are performed. But the simultaneity requirement is not respected since *does* is adjoined at a moment where the adjunctions site for *seem* is not yet available. Consequently, in order to check whether a given tree is part of the tree language of a given MCTAG, one really has to check the possible derivations of this tree including the different derivation orders.

In contrast to this, in a TAG it is sufficient to check whether there is a derivation tree that yields the tree in question; one can abstract away from the order of the derivation steps. E.g. in Fig. 1 it does not matter in which order *John* and *always* are added to *laughs*. The derivation tree and consequently the derived tree are the same.

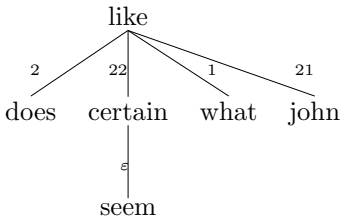
For MCTAG as well one would like to abstract away from differences with respect to derivation order that do not yield different derived trees and that do not make any difference concerning the substitutions and adjunctions that are performed. One way to achieve this is to consider an MCTAG as a TAG G where the elementary trees are grouped into pairwise disjoint sets. Because of this grouping, only some of the derivation trees in G are licensed by the MCTAG. Namely those that satisfy the constraints following from the multicomponent sets and the simultaneity requirement. This is the idea we will pursue in this paper.

Each MCTAG derivation step is a sequence of substitutions and/or adjunctions. Consequently, each derivation in an MCTAG G with $G = \langle N, T, S, I, A, \mathcal{A}\rangle$ corresponds to a derivation tree in the underlying TAG $G_{TAG} = \langle N, T, S, I, A\rangle$ with a node for every elementary tree component of every instance of an element of $\mathcal{A}$ used and an edge for every adjunction/substitution. Let us define the *TAG derivation tree* of such a multicomponent derivation as the corresponding derivation tree in G_{TAG}.[7] We can then define different variants of MCTAG by putting different constraints on this derivation tree.

Consider for example the derivation trees in Fig. 4. They are both possible in a TAG with the elementary trees from the MCTAG in Fig. 3. The first derivation tree is the one for the derivation from Fig. 3. Since we know that only *does* and *seem* are in one set (all other trees are in singletons) and since there is no dominance relation between *does* and *seem*, this is a possible TAG derivation tree in the MCTAG from Fig. 3. The second derivation tree in Fig. 4 which is possible in the underlying TAG (while generating a string different from (4)), should not be possible in the MCTAG: Since *seem* adjoins into *does*, it is not possible to order the adjunctions and substitutions in such a way as

[7] Note that this TAG derivation tree is not the *MCTAG derivation tree* defined in (Weir, 1988). We will discuss the difference in Section 4.

TAG derivation tree for the derivation of (4) in Fig. 3:



TAG derivation tree that does not correspond to
a possible MCTAG derivation:

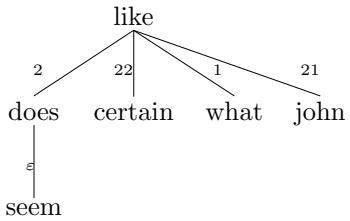


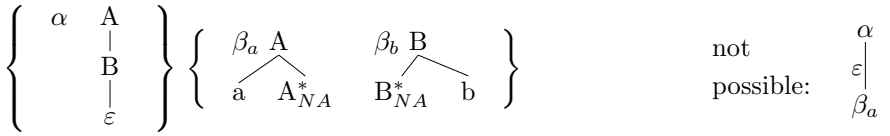
Figure 4. TAG derivation trees for the TAG underlying Fig. 3

to add *does* and *seem* simultaneously to different nodes in an already derived tree.

3.2. Declarative MCTAG definition

In general, the TAG derivation trees for MCTAG derivations must have the two properties described in the following.

Firstly, if an instance of an elementary tree set is used, then all trees from this set must occur in the derivation tree. This is the multicomponent condition that we will call (MC). Fig. 5 shows a sample derivation that does not satisfy (MC).



(only one tree from the tree set used)

Figure 5. A derivation that does not satisfy (MC)

In other words, trees γ_1 and γ_2 from the same elementary tree set must be used the same number of times in the course of a derivation. This is expressed by the condition (MC):

DEFINITION 11 (Multicomponent condition).

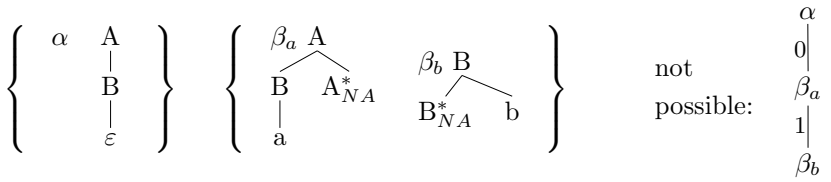
Let $G = \langle N, T, S, I, A, \mathcal{A}\rangle$ be an MCTAG, $G_{TAG} = \langle N, T, S, I, A\rangle$. Let $D = \langle V, E, r\rangle$ be the derivation tree of a saturated derived initial tree in G_{TAG}.

D is a multicomponent TAG derivation tree in G iff

(MC) *If γ_1, γ_2 are in the same $\Gamma \in \mathcal{A}$, then*
$|\{v \,|\, v \in V \text{ and } l(v) = \gamma_1\}| = |\{v \,|\, v \in V \text{ and } l(v) = \gamma_2\}|$.

Note that, in order to check this definition on a given derivation tree, it is not necessary to know which occurrences γ_1 and γ_2 come from the same instance of Γ, i.e., it is not necessary to group the elementary trees involved in the derivation into tree set instances.

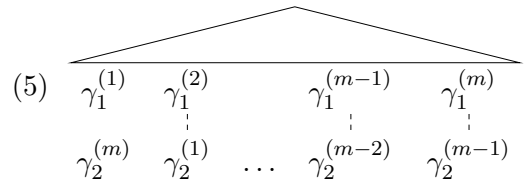
The second property is the simultaneity requirement (SIM). It can be split into two conditions: On the one hand, one tree from an instance of an elementary tree set cannot be substituted or adjoined into another tree from the same set instance. An example is shown in Fig. 6.



(trees from the same tree set adjoined into each other)

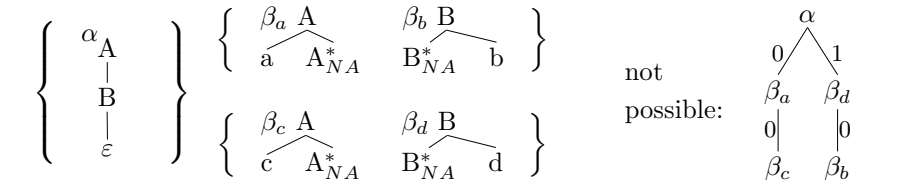
Figure 6. A derivation that does not satisfy the first part of (SIM)

On the other hand, two tree sets cannot be interleaved, i.e., they must be added one after the other. Consequently, derivation trees as (5) are not allowed with $\gamma_1^{(i)}$ and $\gamma_2^{(i)}$ being in the same tree set instance Γ_i because (5) indicates that there is a cycle in the order of adding the tree sets: Γ_2 must be added before Γ_1, Γ_3 before Γ_2 etc., Γ_m before Γ_{m-1} and Γ_1 before Γ_m. By transitive closure, Γ_1 must be added before Γ_m and Γ_m before Γ_1.



An example where this second condition is not satisfied is shown in Fig. 7.

For non-local MCTAG, these are all constraints the TAG derivation tree needs to satisfy. If the simultaneity requirement is dropped, only the multicomponent condition, (MC), must be satisfied. In order to



(β_a added before β_c but β_d added before β_b ⇒ neither $\{\beta_a, \beta_b\}$ nor $\{\beta_c, \beta_d\}$ can be added first)

Figure 7. A derivation that does not satisfy the second part of (SIM)

formalize the simultaneity constraint, we introduce the notion of $\mathcal{A}$-partition of the set V of nodes in a MCTAG derivation tree:

DEFINITION 12 ($\mathcal{A}$-partition).

Let $G = \langle N, T, S, I, A, \mathcal{A}\rangle$ be an MCTAG, $G_{TAG} = \langle N, T, S, I, A\rangle$. Let $D = \langle V, E, r\rangle$ be the derivation tree of a saturated derived initial tree in G_{TAG}.

A partition $V_1, \ldots, V_n$ of V is called an $\mathcal{A}$-partition *if for each V_i ($1 \leq i \leq n$) there is a $\Gamma_i \in \mathcal{A}$ such that $\Gamma_i = \{\gamma \mid l(v) = \gamma$ for some $v \in V_i\}$ and $|V_i| = |\Gamma_i|$.*

The simultaneity constraint (SIM) is then defined as follows:

DEFINITION 13 (Simultaneity condition).

Let $G = \langle N, T, S, I, A, \mathcal{A}\rangle$ be an MCTAG, $G_{TAG} = \langle N, T, S, I, A\rangle$. Let $D = \langle V, E, r\rangle$ be the derivation tree of a saturated derived initial tree in G_{TAG}.

D is a simultaneous TAG derivation tree in G iff

(SIM) *There is an $\mathcal{A}$-partition $V_1, \ldots, V_n$ of V such that*

(a) For all V_i ($1 \leq i \leq n$) and $v, v' \in V_i$ with $v \neq v'$: $\langle v, v'\rangle \notin E^*$.

(b) For all pairwise different $V^{(1)}, V^{(2)}, \ldots, V^{(m)} \in \{V_1, \ldots, V_n\}$ with $m \geq 2$:

There are no $n_1^{(i)}, n_2^{(i)} \in V^{(i)}$ ($1 \leq i \leq m$) such that $\langle n_1^{(1)}, n_2^{(m)}\rangle \in E^*$ and $\langle n_1^{(i)}, n_2^{(i-1)}\rangle \in E^*$ for $2 \leq i \leq m$.

(SIM) implies (MC) because of the partition requirement: If we can partition the nodes into sets V_i such that the labels of each V_i form an elementary set with no elementary tree occurring more than once, (MC) is necessarily satisfied.

Now we can show the following lemma:

LEMMA 1.

In a non-local MCTAG $G = \langle N, T, S, I, A, \mathcal{A}\rangle$ with $G_{TAG} = \langle N, T, S, I, A\rangle$, the following holds:

A derivation tree D of a saturated derived initial tree t in G_{TAG} is a possible TAG derivation tree in G iff D satisfies (SIM).

This lemma corresponds to the Lemma 1 shown in (Kallmeyer, 2005) except for the following differences in the definition and notation of (MC1)–(MC3) from (Kallmeyer, 2005) and the condition (SIM) defined here: In (Kallmeyer, 2005), the nodes of the derivation tree are defined as instances of elementary trees (not as nodes labeled with elementary trees). (MC1) from (Kallmeyer, 2005) states that for every instance of an elementary tree set, if one of its elements occurs in the derivation tree D, then all of them must occur in D. This is equivalent to the condition on the partition of the nodes into sets V_i required in (SIM) if we take a set V_i in (SIM) as a tree set instance Γ in (MC1). Conditions (MC2) and (MC3) from (Kallmeyer, 2005) are the parts a) and b) of the (SIM) condition. The adaptation of the proof from (Kallmeyer, 2005) to the notations used here is given in the appendix.

Lemma 1 gives us a way to characterize non-local MCTAG via the properties of the TAG derivation trees the grammar licenses. With this characterization we get rid of the original simultaneity requirement: The corresponding properties are now captured in the constraints (MC) and (SIM). But, since these constraints need to hold only for the TAG derivation trees that correspond to derived trees in the tree language, sub-derivation trees need not satisfy them. In other words, γ_1 and γ_2 from the same elementary tree set can be added at different moments of the derivation as long as the final complete TAG derivation tree satisfies (SIM).

Note, however, that in order to check (SIM), we need to group the elementary tree instances of a derivation into set instances. This is why non-local MCTAG parsing is NP complete. In the following, we therefore try to find characterizations of the TAG derivation trees of different types of MCTAG that can be checked without performing this grouping. A first example was (MC) that only requires the number of occurrences in a derivation to be the same for trees from the same elementary tree set.

4. Different types of MCTAG

4.1. Local MCTAG variants

4.1.1. *Tree-locality and set-locality*

We can define tree-local and set-local TAG derivation trees by imposing further conditions. To satisfy tree-locality, in the TAG derivation tree, the root label must be in a singleton elementary tree set and,

furthermore, for every instance of a tree set used in the derivation, all trees from this set instance must be added to the same elementary tree. Alternatively, we can capture this condition by stating that for every γ_1 and γ_2 from the same elementary tree set, every node must have equal numbers of γ_1-daughters and γ_2-daughters:

DEFINITION 14 (Tree-locality condition).
Let $G = \langle N, T, S, I, A, \mathcal{A}\rangle$ *be an MCTAG,* $G_{TAG} = \langle N, T, S, I, A\rangle$. *Let* $D = \langle V, E, r\rangle$ *be the derivation tree of a saturated derived initial tree in* G_{TAG}.
D is a tree-local TAG derivation tree in G iff
(TL) $\{l(r)\} \in \mathcal{A}$ *and for every* $\Gamma \in \mathcal{A}$, $\gamma_1, \gamma_2 \in \Gamma$ *and* $v \in V$*:*
$|\{v' \mid \langle v, v'\rangle \in E \text{ and } l(v') = \gamma_1\}| = |\{v' \mid \langle v, v'\rangle \in E \text{ and } l(v') = \gamma_2\}|$.

From (TL), (MC) and (SIM) follow immediately. Therefore, we can define tree-local MCTAG by saying that an MCTAG G is *tree-local* iff the TAG derivation trees licensed by it satisfy (TL).

The (TL) condition can be encoded in the functions f_{OA} and f_{SA} of the underlying TAG. This is why tree-local MCTAG generate exactly Tree Adjoining Languages as observed in (Joshi, 1987).

DEFINITION 15 (Set-locality condition).
Let $G = \langle N, T, S, I, A, \mathcal{A}\rangle$ *be an MCTAG. Let* $D = \langle V, E, r\rangle$ *be the derivation tree of a saturated derived initial tree in* G_{TAG}.
D is called set-local *if*
(SL) *there is an* $\mathcal{A}$*-partition* $V_1, \ldots, V_n$ *of* V *such that for all* V_i $(1 \leq i \leq n)$, *either* $V_i = \{l(r)\}$ *or there is a* V_j $(1 \leq j \leq n)$ *such that for every* $v \in V_i$ *there is a* $v' \in V_j$ *with* $\langle v', v\rangle \in E$.

Note that (SL) implies (MC) because of the possibility to partition V into sets labeled with elementary tree sets required in (SL).

Actually, not only (TL) but even (SL) implies (SIM):

LEMMA 2. *Let* $G = \langle N, T, S, I, A, \mathcal{A}\rangle$ *be an MCTAG. Let* $D = \langle V, E, r\rangle$ *be the derivation tree of a saturated derived initial tree in* G_{TAG}.
If D satisfies (SL) then it satisfies (SIM).

The proof is given in the appendix.

We can therefore define set-local MCTAG imposing only (SL): An MCTAG G is a *set-local MCTAG* if the TAG derivation trees licensed by it satisfy (SL).

For tree-local and set-local MCTAG, Weir (1988) introduces a special MCTAG derivation structure called *MCTAG derivation tree*. This is a derivation tree where the node labels are elementary tree sets

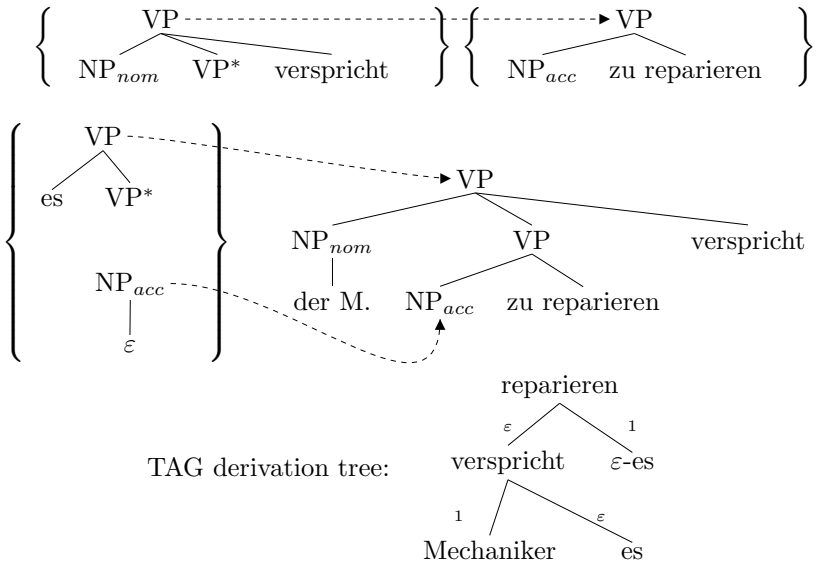


Figure 8. SN-MCTAG derivation of (6)

and the edge labels are tuples of node addresses. Each edge stands for adjunctions/substitutions of the trees from the daughter set into the trees of the mother sets at the nodes whose addresses are specified in the edge label. However, such derivation trees can be obtained only for set-local (and then also for tree-local) MCTAG. They are not available for non-local MCTAG or for the other types of MCTAG that we will describe in this paper.

4.1.2. *SN-MCTAG*

Another variant that also involves a locality constraint is tree-local MCTAG with shared nodes (SN-MCTAG, Kallmeyer, 2005). See the derivation of (6) in Fig. 8 to illustrate the idea of SN-MCTAG:

(6) ... dass [es]$_1$ der Mechaniker [t_1 zu reparieren] verspricht
... that it the mechanic to repair promises
'... that the mechanic promises to repair it'

Consider the root of the *reparieren* tree. The *verspricht* tree adjoins to this node. In standard TAG, in the derived tree (see Fig. 8), the root node belongs only to *verspricht*, i.e., further adjunctions at that node are adjunctions to *verspricht*. In contrast to this, in SN-MCTAG, the node in question is considered as being shared by *reparieren* and *verspricht* since it is a merging or unification of the root nodes of

reparieren and *verspricht*. Further adjunctions at that node can be considered being either adjunctions to *reparieren* or to *verspricht*. This node sharing, combined with tree-locality, gives additional expressive power (Kallmeyer, 2005).

SN-MCTAG are defined via the properties of the underlying TAG derivation trees: In the TAG derivation trees in SN-MCTAG, for each elementary tree set there must be a γ such that all elements from the tree set are either daughters of γ or linked to a daughter of γ by a chain of adjunctions at root nodes. For example, in Fig. 8 ε-*es* is substituted into *reparieren* while *es* is adjoined to the root of *verspricht* which is adjoined to *reparieren*.

In the following, we adapt the definition from (Kallmeyer, 2005) to the formalization of TAG and TAG derivation trees used here.

DEFINITION 16 (SN-tree-locality). *Let $G = \langle N, T, S, I, A, \mathcal{A}\rangle$ be an MCTAG. Let $D = \langle V, E, r\rangle$ be the derivation tree of a saturated derived initial tree in G_{TAG}.*

D is called SN-tree-local *if*

(SN-TL) *there is an $\mathcal{A}$-partition $V_1, \ldots, V_n$ of V such that for every $V_i = \{n_1, \ldots, n_m\}$ $(1 \leq i \leq n)$ there is a node $n_0 \in V$ such that: for all n_j $(1 \leq j \leq m)$ either $\langle n_0, n_j\rangle \in E$ or there are $n_{j,1}, \ldots, n_{j,k} \in V$ for some $k > 1$ with $n_j = n_{j,k}$, $\langle n_0, n_{j,1}\rangle \in E$ and for $1 \leq l \leq k-1$: $\langle n_{j,l}, n_{j,l+1}\rangle \in E$ and $g(\langle n_{j,l}, n_{j,l+1}\rangle) = \varepsilon$ (i.e. root adjunction).*

G is a *tree-local MCTAG with Shared Nodes (SN-MCTAG)* if the TAG derivation trees licensed by it satisfy (SIM) and (SN-TL). ((MC) follows from (SN-TL).)

Here (in contrast to set-local MCTAG), the simultaneity constraint cannot be omitted. As an example consider the MCTAG in Fig. 9. If we impose (SIM) and (SN-TL), we obtain the copy language $\{ww \mid w \in \{a, b\}^*\}$: In each derivation step, one of the tree sets is picked and, simultaneously, its first element adjoins to the A-daughter of the S-node, its second element to the B-daughter. However, if we impose only (MC) and (SN-TL), we obtain the language $\{ww' \mid w$ and w' contain equal numbers of as and equal numbers of bs$\}$ since we can delay adjunctions: The only condition is that all the A-auxiliary trees involved in the derivation adjoin at some point to the A-daughter of the S-node and similarly for the B-auxiliary trees. (Except the first, all of them adjoin to the root, so SN-tree locality is trivially satisfied.)

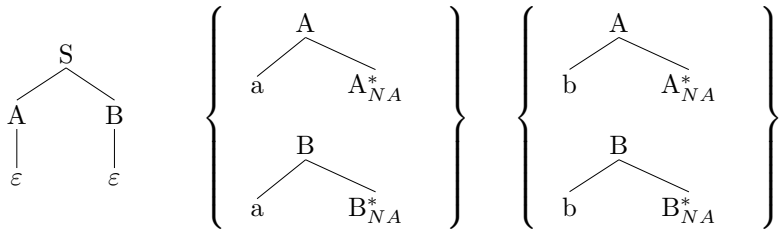


Figure 9. SN-MCTAG for the copy language

4.1.3. *TT-MCTAG*

A more recent MCTAG variant proposed in the context of anlyzing German scrambling data is SN-MCTAG with Tree Tuples (TT-MCTAG, Lichte, 2007). The grammar itself is slightly different from standard MCTAG since the elementary tree sets contain two parts: 1. one lexicalized tree γ, the unique *head tree*, and 2. a set of auxiliary trees, the so-called *argument trees*. Such a pair is called a tree tuple. During derivation, the auxiliary trees must either adjoin directly to their head tree or they must be linked by a chain of adjunctions at root nodes to a tree that attaches to the head tree. In other words, in the corresponding TAG derivation tree, the head tree must dominate the auxiliary trees such that all positions on the path between them, except the first one, must be root node addresses. This is again the notion of adjunction under node sharing.

Fig. 10 shows the TT-MCTAG analysis of (6). In this derivation, the NP_{nom} auxiliary tree adjoins directly to *verspricht* (its head tree) while the NP_{acc} tree adjoins to the root of a tree that adjoins to the root of a tree that adjoins to *reparieren*.

In contrast to SN-MCTAG, TT-MCTAG does not require simultaneity. The simultaneity requirement (SIM) formulated for standard MCTAG (without tuples) is actually in contradiction to the TT-MCTAG requirement of adjoining the argument trees of a tuple to their head tree or to trees attached to their head tree. Even simultaneity of the adjunctions of the auxiliary trees of a tuple to their head tree is not required.

DEFINITION 17 (TT-MCTAG).

Let $G = \langle N, T, S, I, A, \mathcal{A} \rangle$ *be an MCTAG. G is a* Tree-Tuple MCTAG with Shared Nodes (TT-MCTAG) *iff every* $\Gamma \in \mathcal{A}$ *has the form* $\{\gamma, \beta_1, \ldots, \beta_n\}$ *where* γ *is a tree with at least one leaf with terminal label, the* head tree*, and* $\beta_1, \ldots, \beta_n$ *are auxiliary trees, the* argument trees*. We write such a set as a tuple* $\langle \gamma, \{\beta_1, \ldots, \beta_n\} \rangle$.

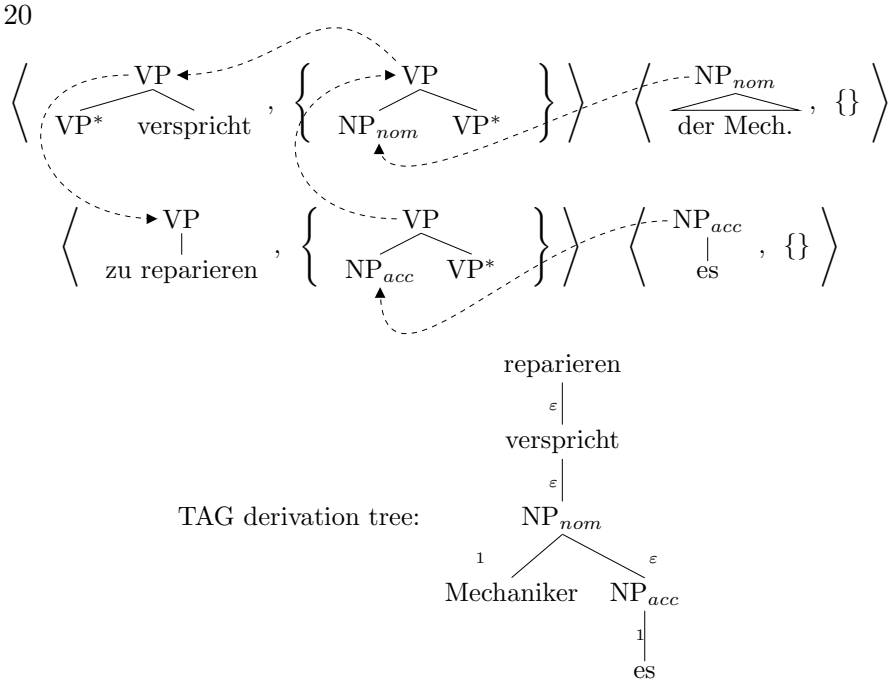


Figure 10. TT-MCTAG derivation of (6)

As a notation, for a given argument tree β, $h(\beta)$ denotes the head of β. For a given TT-MCTAG G, $H(G)$ is the set of head trees and $A(G)$ the set of argument trees.

DEFINITION 18 (SN-tree-tuple-locality). *Let $G = \langle N, T, S, I, A, \mathcal{A}\rangle$ be a TT-MCTAG, $G_{TAG} = \langle N, T, S, I, A\rangle$. Let $D = \langle V, E, r\rangle$ be the derivation tree of a saturated derived initial tree in G_{TAG}.*

D is a SN-tree-tuple-local TAG derivation tree in G iff

(SN-TTL) *For all argument trees $\beta \in A(G)$:*

If $v_1, \ldots, v_n \in V$ are pairwise different nodes with $l(v_i) = h(\beta)$ for $1 \leq i \leq n$, then there are pairwise different $u_1, \ldots, u_n \in V$ with $l(u_i) = \beta$ for $1 \leq i \leq n$ such that for all $i, 1 \leq i \leq n$: either $\langle v_i, u_i\rangle \in E$ or there are nodes $u_{i,1}, \ldots, u_{i,k} \in V$ with $k > 1$ such that $u_i = u_{i,k}$, $\langle v_i, u_{i,1}\rangle \in E$ and for $1 \leq j \leq k-1$: $\langle u_{i,j}, u_{i,j+1}\rangle \in E$ and $g(\langle u_{i,j}, u_{i,j+1}\rangle) = \varepsilon$ (i.e. root adjunction).

Note that a derivation tree satisfying (SN-TTL) does not necessarily satisfy (MC) since it only requires that for every $h(\beta)$-node there is a corresponding β-node but not vice versa.

We define that for a TT-MCTAG G, a derivation tree D in the underlying TAG G_{TAG} is licensed in G iff it satisfies (MC) and (SN-TTL).

Checking the condition (SN-TTL) as it is provided here requires to find an appropriate partition of the derivation tree nodes. In other words, one has to group the tree instances into set instances. In order to avoid this, Kallmeyer and Satta (2009) provide a characterization of (SN-TTL) based on a counting of elementary trees.

For a node v in a derivation tree D, we write D_v to represent the subtree of D rooted at v. For $\gamma \in (I \cup A)$, we define $Dom(v, \gamma)$ as the set of nodes of D_v that are labeled by γ. Furthermore, for an argument tree $\beta \in A(G)$, we let $\pi(v, \beta) = |Dom(v, \beta)| - |Dom(v, h(\beta))|$. Intuitively, $\pi(v, \beta)$ gives us the number of pending β-nodes below v, i.e., the number of β instances below v whose heads are higher.

DEFINITION 19 (SN-tree-tuple-locality, revised).

Let G be an TT-MCTAG, $G_{TAG} = \langle N, T, S, I, A \rangle$. Let $D = \langle V, E, r \rangle$ be the derivation tree of a saturated derived initial tree in G_{TAG}.

D is a SN-tree-tuple-local TAG derivation tree in G iff

(SN-TTL-counting) *For every $v \in V$ and every $\beta \in A(G)$, the following conditions both hold.*

1. $\pi(v, \beta) \geq 0$.
2. *If $\pi(v, \beta) > 0$, then one of the following conditions must be satisfied:*
   - *a) $l(v) = \beta$ and $\pi(v, \beta) = 1$;*
   - *b) $l(v) = \beta$ and $\pi(v, \beta) > 1$, and there is some $\langle v, v_\varepsilon \rangle \in E$ with $g(\langle v, v_\varepsilon \rangle) = \varepsilon$ and $\pi(v_\varepsilon, \beta) + 1 = \pi(v, \beta)$;*
   - *c) $l(v) \notin \{\beta, h(\beta)\}$ and there is some $\langle v, v_\varepsilon \rangle \in E$ with $g(\langle v, v_\varepsilon \rangle) = \varepsilon$ and $\pi(v_\varepsilon, \beta) = \pi(v, \beta)$;*
   - *d) $l(v) = h(\beta)$ and there is some $\langle v, v_\varepsilon \rangle \in E$ with $g(\langle v, v_\varepsilon \rangle) = \varepsilon$ and $\pi(v, \beta) \leq \pi(v_\varepsilon, \beta) \leq \pi(v, \beta) + 1$.*

(Kallmeyer and Satta, 2009) have shown the following lemma:

LEMMA 3. *Let G be a TT-MCTAG with underlying TAG G_T, and let $D = \langle V, E, r \rangle$ be a derivation tree in G_T that satisfies (MC). D satisfies (SN-TTL) if and only if it satisfies (SN-TTL-counting).*

Furthermore, they show that the characterization of TT-MCTAG derivation trees using (SN-TTL-counting) can be used to do polynomial parsing of TT-MCTAG since, due to the lexicalization of the grammar, the number of pending arguments is always limited, depending on the length of the input.

Besides this general definition of TT-MCTAG, a limitation for TT-MCTAG was introduced (Kallmeyer and Parmentier, 2008): TT-MCTAG are of rank k if, at any time during the derivation, at most k argument trees depending on higher heads in the derivation tree are still waiting for adjunction.

DEFINITION 20 (TT-MCTAG derivation of rank k).

Let G be an TT-MCTAG, $G_{TAG} = \langle N, T, S, I, A\rangle$. Let $D = \langle V, E, r\rangle$ be the derivation tree of a saturated derived initial tree in G_{TAG}.

D is of rank k iff

(TT-k) *There is no $v \in V$ such that*

$$\sum_{\beta \in A(G)} (|\{v' \,|\, l(v') = \beta \text{ and } \langle v, v'\rangle \in E^+\}| - |\{v' \,|\, l(v') = h(\beta) \text{ and } \langle v, v'\rangle \in E^*\}|) > k$$

A TT-MCTAG G is of rank k (or a *k-TT-MCTAG* for short) iff each TAG derivation tree $D = \langle V, E, r\rangle$ licenced in G satisfies (MC), (SN-TTL) and (TT-k).

Note that, since k is limited, the list of pending argument trees at every node in the derivation tree is limited as well. Consequently, we can encode this list as a condition in the names of elementary trees and model the adjunctions of pending arguments with the f_{OA} and f_{SA} functions. In other words, for a given k-TT-MCTAG we can construct an equivalent TAG that generates the same derived trees but that does not capture the relations between heads and arguments. This idea is followed in the construction of an equivalent simple 2-RCG (Range Concatenation Grammar) for a given k-TT-MCTAG in (Kallmeyer and Parmentier, 2008). We therefore obtain that k-TT-MCTAG generate exactly all Tree Adjoining Languages.

In general, the universal recognition problem for TT-MCTAG and probably also for k-TT-MCTAG is NP-hard (Søgaard et al., 2007). But parsing can be done in an amount of time polynomial in the length of the input string (Kallmeyer and Parmentier, 2008; Kallmeyer and Satta, 2009).

4.1.4. *Delayed tree-locality*

Another recent proposal for extending the expressive power of TAG with multicomponents while keeping the idea of tree-locality is *delayed tree-locality* (Chiang and Scheffler, 2008). This idea arouse from the desire to provide a precise formal definition of *flexible composition*.

Let us briefly sketch the idea of flexible composition. Flexible composition (Joshi et al., 2007) is a way of viewing TAG derivation so that the operation of adjoining a tree β into a tree γ can be alternatively

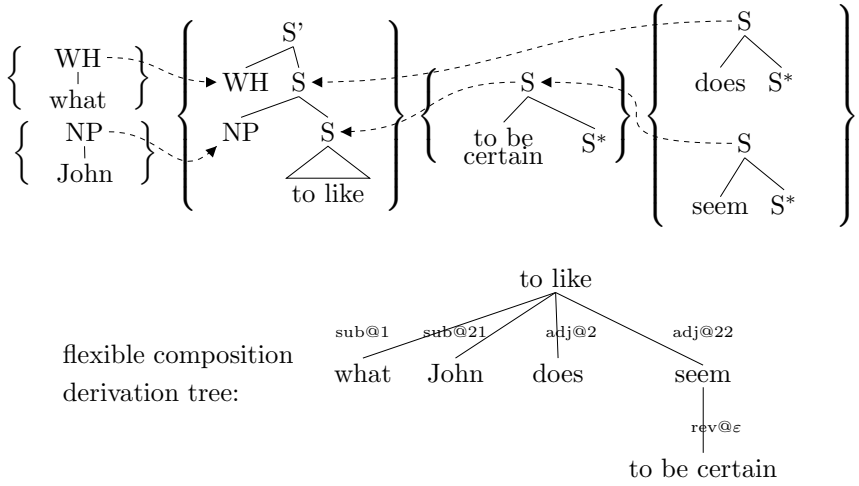


Figure 11. Tree-local MCTAG with flexible composition

viewed as attachment of γ to β. Applied to MCTAG, this "flexible" view allows to turn non-local derivations into tree-local derivations.

Formalizing flexible composition leads to operations different from adjunction and substitution, namely to reverse-adjunctions as defined in (Chiang and Scheffler, 2008). As an example consider the non-local derivation from Fig.3, repeated in Fig.11 together with its flexible composition derivation tree. If we adopt flexible composition , we can say that *to be certain* reverse-adjoins to *seem* which in turn adjoins to *like*. The reverse adjunction is a splitting at a node (here the root of *to be certain*) and a wrapping around an auxiliary tree. As can be seen in Fig.11, the corresponding derivation tree satisfies (TL).

However, this new derivation tree is not the underlying TAG derivation tree since it has reverse adjunction edges. Instead of using this new operation, one can also think of flexible composition as delayed adjunction as introduecd in (Chiang and Scheffler, 2008): All trees from one set must be adjoined (or substituted) at some point of the derivation but, instead of adding all to the same elementary tree, some of them can be delayed. Chiang and Scheffler restrict this mechanism of delayed adjunction by requiring that elementary trees from at most k different elementary tree set instances can be delayed at every particular node in the underlying TAG derivation tree.

In order to formulate the corresponding condition, we need the definition of the delay of a set of nodes in the TAG derivation tree.

DEFINITION 21 (delay).
Let $D = \langle V, E, r\rangle$ be a TAG derivation tree and $U \subseteq V$.

1. *The* destination *of U (in D) is the least common ancestor $v \in V$ of all $u \in U$.*

2. *The* delay *of U is then $\bigcup_{u \in U}\{u' \mid \langle v, u'\rangle, \langle u', u\rangle \in E^*, u' \neq v\}$ where v is the destination of U.*

DEFINITION 22 (k-delayed-tree-locality).

Let G be an MCTAG, $G_{TAG} = \langle N, T, S, I, A\rangle$. Let $D = \langle V, E, r\rangle$ be the derivation tree of a saturated derived initial tree in G_{TAG}.

D is a k-delayed tree-local TAG derivation tree with multicomponents in G iff

(k-DTL) *there is an $\mathcal{A}$-partition $V_1, \ldots, V_n$ of V such that for every $v \in V$: $|\{V_i' \mid V_i'$ delay of a $V_i, 1 \leq i \leq n$ and $v \in V_i'\}| \leq k$.*

This is a formalization of the condition for a k-delayed tree-local MCTAG deriation trees from (Chiang and Scheffler, 2008). The condition (k-DTL) makes use of the partition of the derivation tree into elementary tree set instances. As already mentioned, if possible, we want to avoid this since it increases the complexity of parsing. For k-delayed tree-locality, we can avoid the grouping into tree set instances by, instead, requiring that the TAG derivation tree satisfies (MC) and that the number of incomplete elementary tree sets below any given node v is limited to k. This number can be obtained as follows. For every elementary tree set Γ and every $\gamma_1, \gamma_2 \in \Gamma$, the following holds: If there are n_1 nodes with label γ_1 and only $n_2 < n_1$ nodes with label γ_2 below v, then this means that when visiting v, there are at least $n_1 - n_2$ incomplete sets Γ. In general, the maximal number of γ-nodes for any $\gamma \in \Gamma$ minus the minimal number of γ-nodes for any $\gamma \in \Gamma$ provides the number of incomplete Γ-sets. This is formulated in the following alternative definition of (k-DTL-counting):

DEFINITION 23 (k-delayed-tree-locality, revised).

Let G be an MCTAG, $G_{TAG} = \langle N, T, S, I, A\rangle$. Let $D = \langle V, E, r\rangle$ be the derivation tree of a saturated derived initial tree in G_{TAG}.

For any $\gamma \in I \cup A, v \in V$ we define $dom_\gamma(v) = |\{v' \mid \langle v, v'\rangle \in E^*, l(v') = \gamma\}|$ as the number of γ-nodes dominated by v.

D is a k-delayed tree-local TAG derivation tree in G iff

(k-DTL-counting) *for every $v \in V$:*

$$\sum_{\Gamma \in \mathcal{A}} (\max_{\gamma \in \Gamma} |Dom_\gamma(v)| - \min_{\gamma \in \Gamma} |Dom_\gamma(v)|) \leq k.$$

The following lemma holds:

LEMMA 4. *Let G be a MCTAG with underlying TAG G_T, and let $D = \langle V, E, r\rangle$ be a derivation tree in G_T.*

D satisfies (k-DTL) if and only if it satisfies (MC) and (k-DTL-counting).

This lemma follows immediately from the fact that for any node v in the derivation tree and any set $\Gamma \in \mathcal{A}$, the maximal number of γ-nodes for any $\gamma \in \Gamma$ dominated by v minus the minimal number of γ-nodes for any $\gamma \in \Gamma$ dominated by v provides the number of incomplete instances of the set Γ below v, i.e., the number of Γ instances with v being part of their delay.

Note that (MC) follows from (k-DTL) but not from (k-DTL-counting). (SIM) is not required for delayed tree-local MCTAG: An MCTAG G is a *k-delayed tree-local MCTAG* iff the TAG derivation trees licensed by it satisfy (MC) and (k-DTL-counting).

We will show in section 5 how this characterization can be exploited for parsing lexicalized k-delayed tree-local MCTAG in polynomial time.

Chiang and Scheffler (2008) show that for every tree-local MCTAG with flexible composition, there is a weakly equivalent 2-delayed tree-local MCTAG that even has the same elementary tree sets.

Note that k-TT-MCTAG is a special case of k-delayed tree-locality where the head of an elementary tree set instance is its destination. Therefore, the following lemma holds:

LEMMA 5. *Let $G = \langle N, T, S, I, A, \mathcal{A}\rangle$ be an TT-MCTAG. Let $D = \langle V, E, r\rangle$ be the derivation tree of a saturated derived initial tree in G_{TAG}.*

If D satisfies (TT-k) and (SN-TTL) then it satisfies (k-DTL-counting).

Similar to tree-local MCTAG and k-TT-MCTAG, k-delayed tree-local MCTAG are also weakly equivalent to TAG (Chiang and Scheffler, 2008): The set of delayed trees can be encoded in the trees and thereby constrain the derivation tree.

4.2. Non-local MCTAG variants

Other restrictions sometimes imposed on MCTAG derivations come from so-called *dominance links*. Examples are MCTAG with dominance links (Becker et al., 1991) and Vector MCTAG with Dominance Links (V-TAG, Rambow, 1994). Vector MCTAG, defined in (Rambow, 1994), are like non-local MCTAG except that there is no simultaneity requirement for the derivations.

Dominance links are constraints of the form $f \geq n$ where f is a foot node in some tree β, n an internal node in some tree γ such that β and γ belong to the same elementary tree set. A dominance link $f \geq n$ means that whenever an instance of this set is used in a derivation,

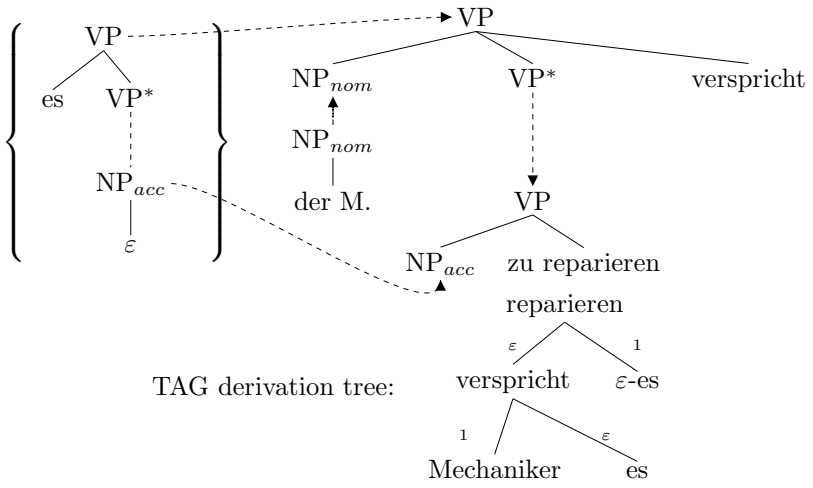


Figure 12. Multicomponent derivation with dominance constraints

in the final derived tree, the node corresponding to f in this instance must dominate the node corresponding to n (or anything adjoined to this node).

As an example for dominance constraints see the analysis of (6) in Fig. 12. Here again the scrambled *es* and the empty trace it leaves behind are two elements of the same elementary tree set. Furthermore, the dominance link between the foot node of the *es* auxiliary tree and the root of the empty NP means that in the resulting derived tree the moved element must c-command its trace. A derivation as in Fig. 12 is therefore possible. The crucial property of the corresponding TAG derivation tree (see Fig. 8) is that *es* is adjoined to the spine[8] of *verspricht* which is adjoined to *reparieren* at a node that dominates the substitution site of ε-*es* (the tree with the empty trace).

Dominance constraints are restrictions on the derived trees. We can formulate a corresponding constraint (Dom) for the possible TAG derivation trees.[9]

We first define the notion of *spine-dominance*. For this we need spine addresses: a spine address is a an address of a node on the spine of an auxiliary tree, i.e., a prefix of the foot node address. A node in a derivation tree spine-dominates another node if all edge labels on the path that links them are spine addresses (see Fig. 13).

[8] The spine is the path from the root to the foot node.

[9] Note that dominance constraints are particularly of interest in non-local MC-TAG variants. In most local variants they can be simulated choosing appropriate node labels.

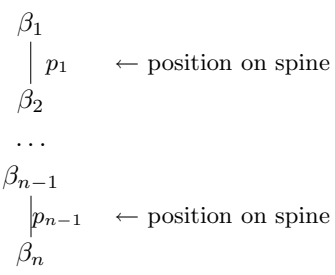


Figure 13. Spine-dominance

DEFINITION 24. *Let $G = \langle N, T, S, I, A\rangle$ be a TAG, $D = \langle V, E, r\rangle$ a derivation tree in G.*

A node $n \in V$ spine-dominates a node $n' \in V$ iff

- *either $n = n'$*
- *or there are $n_1, \ldots, n_m \in V$ $(m > 1)$ with $n = n_1, n' = n_m$ and for all $1 \leq i < m$: $\langle n_i, n_{i+1}\rangle \in E$ with $g(\langle n_i, n_{i+1}\rangle)$ a spine address in $l(n_i)$.*

As a notation, we write E^*_{Spine} for the spine-dominance relation.

If we have a chain of adjunctions such that γ_1 adjoins to some γ_0, γ_2 adjoins to γ_1, γ_3 to γ_2 etc. up to some γ_n, then in the derived tree the following holds: the foot node of γ_n dominates a node k from γ_0 iff (i) this node is below the node γ_1 adjoins to and (ii) in the corresponding derivation tree γ_1 spine-dominates γ_n. This is what the next lemma tells us.

In order to formulate the lemma, we need to specify the nodes in a derived tree that correspond to nodes in the elementary trees that have been used in the derivation. For this purpose, we define a mapping δ from nodes in elementary trees that label nodes in the derivation tree to nodes in the corresponding derived tree. An example for a δ-function for a given pair of derivation and derived tree is shown in Fig. 14.

In the following, as a notation, we write $node(\tau, p)$ for the node at position p in a tree τ.

DEFINITION 25 (δ-function).

Let $G = \langle N, T, S, I, A\rangle$ be a TAG. Let $D = \langle V_D, E_D, r_D\rangle$ be a derivation tree with $t = \langle V_t, E_t, r_t\rangle$ being a corresponding derived tree such that for every node $v \in V_D$, t_v is the instance of $l(v)$ used to obtain t.

Then we define a (partial) function $\delta_{D,t} : \{n \mid n$ is a node in a $\gamma \in I \cup A\} \times V_D \rightarrow V_t$ as follows:

1. *If $V_D = \{r_D\}$ then for every node position p in $l(r_D)$: $\delta_{D,t}(node(l(r_D), p), r_D) = node(t_{r_D}, p)$.*

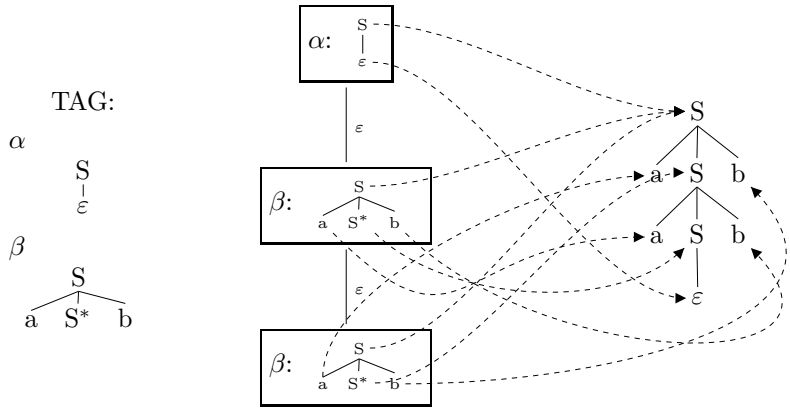


Figure 14. Sample δ-function for the derivation of *aabb*

2. *If* $|V_D| > 1$ *such that* $v_1, \ldots, v_n \in V_D$ *are all nodes with* $\langle r_D, v_i \rangle \in E_D$ *and* $g(\langle r_D, v_i \rangle) = p_i$ *for* $1 \leq i \leq n$*, then for all addresses* p *in* $l(r_D)$*:*

   If $p \notin \{p_1, \ldots, p_n\}$ *then* $\delta_{D,t}(node(l(r_D), p), r_D) = node(t_{r_D}, p)$.

   If $p = p_i \in \{p_1, \ldots, p_n\}$ *then* $\delta_{D,t}(node(l(r_D), p_i), r_D) = \delta_{D,t}(node(l(v_i), \varepsilon), v_i)$.

3. *For all pairs* $\langle n, v \rangle$ *with* $v \in V_D$ *and* n *is a node in an elementary tree* $\gamma \neq l(v)$*,* $\delta_{D,t}(node(l(v), p), v)$ *is undefined.*

We can now formulate our lemma:

LEMMA 6. *Let* $D = \langle V_D, E_D, r_D \rangle$ *be a TAG derivation tree with* $t = \langle V_t, E_t, r_t \rangle$ *being a corresponding derived tree.*

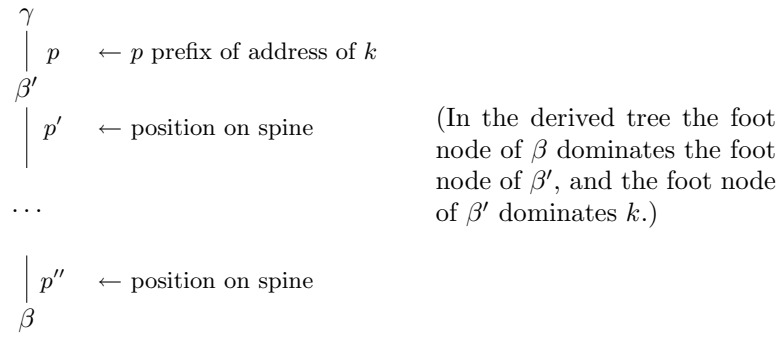
Let $n_0, \ldots, n_m \in V_D$ *with* $\langle n_{i-1}, n_i \rangle \in E_D$ *and* $g(\langle n_{i-1}, n_i \rangle) = p_i$ *for all* $1 \leq i \leq m$*. Let* $l(n_m)$ *be an auxiliary tree with foot node* f *and, furthermore, let* k *be a node in* $l(n_0)$ *with address* p*. Then the following holds:*

$\langle \delta_{D,t}(f, n_m), \delta_{D,t}(k, n_0) \rangle \in E_t^*$ *iff* $\langle n_1, n_m \rangle \in E_{D_{Spine}}^*$ *and* p_1 *is a prefix of* p*.*

The proof of this lemma is given in the appendix.

Below we define the condition (Dom) as a condition on the TAG derivation tree. For a dominance constraint $f \geq k$ with f foot node of β and k internal node in γ, (Dom) requires that k is not part of a derived tree γ' that is attached to β (i.e., in the derivation tree, the β-node does not dominate the γ-node) since adjunction at foot nodes is not allowed. Furthermore, (Dom) distinguishes two cases (a) and (b)

Case (a):



Case (b):

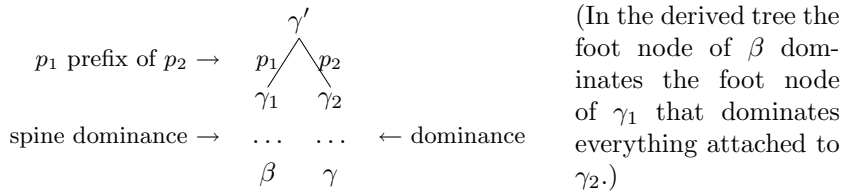


Figure 15. Cases (a) and (b) of (Dom)

that correspond to the configuration of Lemma 6. The two cases are depicted in Fig. 15. Condition (a) makes sure that if $\langle \gamma, \beta \rangle \in E^*$ then the foot node f of β dominates the foot node of β', and the foot node of β' dominates k. (b) guarantees that if $\langle \gamma, \beta \rangle \notin E^*$ then f dominates the foot node of γ_1 that dominates everything attached (by adjunction or substitution) to γ_2, including the node k.

DEFINITION 26. *Let $G = \langle N, T, S, I, A, \mathcal{A} \rangle$ be an MCTAG, Dom_G a set of dominance links for G.*

Let $D = \langle V, E, r \rangle$ be the derivation tree of a saturated derived initial tree in G_{TAG}.

D respects the dominance links in Dom_G iff it satisfies the following:

(Dom) *there is an $\mathcal{A}$-partition $V_1, \ldots, V_n$ of V such that for each V_i, $1 \leq i \leq n$ and each $n_\beta, n_\gamma \in V_i$ such that $l(n_\beta) = \beta \in A$, $l(n_\gamma) = \gamma \in I \cup A$ and there is a dominance constraint $f \geq k$ with f foot node of β, k an internal node in γ:*

$\langle n_\beta, n_\gamma \rangle \notin E^*$ and

(a) if $\langle n_\gamma, n_\beta \rangle \in E^*$, then there is a $n' \in V$ with $\langle n_\gamma, n' \rangle \in E$, $g(\langle n_\gamma, n' \rangle)$ a prefix of the address of k in γ and $\langle n', n_\beta \rangle \in E^*_{Spine}$.

(b) if $\langle n_\gamma, n_\beta\rangle \notin E^*$ *and* n *is the least common ancestor of* n_γ *and* n_β *in* D*, then there are* $n_1, n_2 \in V$ *with* $\langle n, n_1\rangle, \langle n, n_2\rangle \in E$*,* $g(\langle n, n_1\rangle) = p_1, g(\langle n, n_2\rangle) = p_2$ *where* p_1 *is a prefix of* p_2 *and* $\langle n_1, n_\beta\rangle \in E^*_{Spine}$ *and* $\langle n_2, n_\gamma\rangle \in E^*$*.*

In MCTAG with dominance links the TAG derivation trees licensed by the grammar must satisfy (SIM) and (Dom) while in V-TAG they must satisfy only (Dom). ((Dom) implies (MC).)

With Lemma 6, this is almost immediate: In the case (a) where the γ-node dominates the β-node in the derivation tree, Lemma 6 applies immediately. In the second case, (b), there must be a lowest γ'-node that dominates both, the γ-node and the β-node. Then, necessarily, for a dominance relation in the derived tree between any node from β and any node from γ, the immediate daughter n_1 (with label γ_1) of the γ'-node that dominates β must adjoin higher than the immediate daughter n_2 of the γ'-node that dominates γ. Futhermore, according to Lemma 6, for the foot node of β to dominate anything below the adjunction site of γ_1, there must be a spine-dominance relation between n_1 and the β-node.

Fig. 16 gives an overview of the different combinations of the constraints on TAG derivation trees and of the corresponding grammar formalisms. (For a combination of constraints, "–" means that it does not exist while "??" means that it exists but has not been defined in the literature.)

5. Parsing lexicalized MCTAG

The proposed declarative definition of different types of MCTAG can be exploited for parsing. It allows us to consider the underlying TAG derivation tree and check whether this derivation tree satisfies (MC), (SIM), (Dom) or one of the locality conditions. In other words, parsing can be done in two steps: 1. parsing in the underlying TAG G_{TAG} and 2. check of conditions on derivation trees for all derivation trees obtained in the first step. Of course the two steps need not be done separately, they could be combined. In the following we will briefly sketch how this could be done.

We concentrate on lexicalized MCTAG where lexicalization means that every tree set in the grammar contains at least one lexicalized tree. It does not require all trees in the grammar to be lexicalized.

DEFINITION 27 (Lexicalized MCTAG).

Local MCTAGs weakly equivalent to TAG:

	(TL)	(SN-TTL), (TT-k)	(k-DTL-counting)
(MC)	tree-local MCTAG	k-TT-MCTAG	k-delayed tree-local MCTAG
(MC), (SIM)	tree-local MCTAG	–	??

Other local MCTAG variants:

	(SL)	(SN-TL)	(SN-TTL)
(MC)	set-local MCTAG	??	TT-MCTAG
(MC), (SIM)	set-local MCTAG	SN-MCTAG	–

Non-local MCTAG variants:

	(Dom)	–
(MC)	Vector MCTAG with Dominance Links (V-TAG)	Vector MCTAG
(MC), (SIM)	non-local MCTAG with dominance links	non-local MCTAG

Figure 16. Summary of different MCTAG variants

Let $G = \langle N, T, S, I, A, \mathcal{A} \rangle$ *be an MCTAG.* G *is* lexicalized *if for every* $\Gamma \in \mathcal{A}$*, there is at least one* $\gamma = \langle V, E, r \rangle \in \Gamma$ *such that there is a* $v \in V$ *with* $l(v) \in T$*.*

The advantage of using lexicalized MCTAG is that the number of trees used in a derivation is always linear in the input length. If k is the maximal number of trees per tree set in the grammar, then every derivation tree for an input of length n has no more than kn nodes.

5.1. TAG PARSING WITH G_{TAG}

Since we want to check conditions on derivation trees, we choose to do TAG parsing using an algorithm that traverses the derivation tree.[10] In order to achieve this, we transform our TAG into a specific grammar that encodes its derivation trees. One way to do this is to transform the TAG into an equivalent 2-Linear Context-Free Rewriting System (LCFRS, Vijay-Shanker et al., 1987; Weir, 1988) or an ordered simple 2-Range Concatenation Grammar (SRCG, Boullier, 2000). Simple RCG can be seen as a notation of LCFRS where the functions for yield computation are defined inside the rules.[11] We will use simple RCG to describe TAG derivation trees.

5.1.1. *Range Concatenation Grammars*

The idea underlying SRCG is that non-terminals are considered as predicates that are true for certain string tuples. The productions in an RCG (called *clauses*) rewrite predicates ranging over parts of the input by other predicates. The clause $S(aXb) \rightarrow S(X)$ for instance signifies that S is true for a part of the input if this part starts with an a, ends with a b, and if, furthermore, S is also true for the part between a and b.

DEFINITION 28 (Range Concatenation Grammar).

A range concatenation grammar (RCG) *is a 5-tuple* $G = (N, T, V, P, S)$ *where*

1. N *is a finite set of predicate names with an arity function dim:* $N \rightarrow \mathbb{N}^+$,
2. T *and* V *are disjoint finite sets of terminals and variables.*
3. P *is a finite set of clauses of the form* $\psi_0 \rightarrow \psi_1 \dots \psi_m$, *where* $m \geq 0$ *and each of the* $\psi_i, 0 \leq i \leq m$, *is a predicate of the form* $A_i(\alpha^i_1, \dots, \alpha^i_{dim(A)})$. *Each* $\alpha^i_j \in (T \cup V)^*$, $1 \leq j \leq dim(A)$ *and* $0 \leq i \leq k$, *is an argument.*
4. $S \in N$ *is the start predicate name with* $dim(S) = 1$.

As a shorthand notation for $A_i(\alpha_1, \dots, \alpha_{dim(A)})$, we use $A_i(\vec{\alpha})$.

[10] All TAG parsing algorithm (see for instance Vijay-Shanker and Weir, 1993) produce derivation trees but in most cases, parsing is done on the derived tree, i.e., the operations of the parser correspond to a traversal of the derived tree.

[11] For instance, the LCFRS with rules $S \rightarrow f(A)$, $A \rightarrow g(A)$, $A \rightarrow h()$ and functions $f(\langle x, y\rangle) := \langle xy\rangle$, $g(\langle x, y\rangle) := \langle ax, by\rangle$, $h() := \langle a, b\rangle$ can be written as the SRCG with clauses $S(XY) \rightarrow A(X, Y)$, $A(aX, bY) \rightarrow A(X, Y)$, $A(a, b) \rightarrow \varepsilon$.

DEFINITION 29 (Simple Range Concatenation Grammar).

1. *A RCG* $G = (N, T, V, P, S)$ *is a* k-RCG *if for all* $A \in N, dim(A) \leq k$.
2. *A RCG* $G = (N, T, V, P, S)$ *is* simple *if for all* $c \in P$*, every* $X \in V$ *occurring in* c *occurs exactly once in the lefthand side and exactly once in the righthand side, and each argument in the righthand side of* c *contains exactly one* $X \in V$.
3. *A simple RCG* $G = (N, T, V, P, S)$ *is* ordered *if for all* $\psi_0 \rightarrow \psi_1 \cdots \psi_m \in P$*, it holds that if a* $X_1 \in V$ *precedes a* $X_2 \in V$ *in one of the* ψ_i*,* $1 \leq i \leq m$*, then* X_1 *also precedes* X_2 *in* ψ_0.

The ordering requirement does not change the expressive power, i.e., ordered simple RCG is equivalent to simple RCG (Villemonte de La Clergerie, 2002).

In order to be able to talk about specific parts of the input, i.e., in order to distinguish different occurrences of the same string in the input, we introduce *ranges*.

DEFINITION 30 (Range).

Let w *be the input word,* $w = w_1 \ldots w_n$ *(*$n \geq 0$*) with* $w_i \in T$ *for* $1 \leq i \leq n$.

- $Pos(w) = \{0, \ldots, n\}$.
- *We call a pair* $\langle l, r \rangle \in Pos(w) \times Pos(w)$ *with* $l \leq r$ *a* range *in* w*. Its* yield $\langle l, r \rangle(w)$ *is the substring* $w_{l+1} \ldots w_r$.
- *For two ranges* $\rho_1 = \langle l_1, r_1 \rangle, \rho_2 = \langle l_2, r_2 \rangle$*: if* $r_1 = l_2$*, then* $\rho_1 \cdot \rho_2 = \langle l_1, r_2 \rangle$*; otherwise* $\rho_1 \cdot \rho_2$ *is undefined.*

In order to apply a clause, its variables and terminals are instantiated with substrings of the input, or, rather with ranges $\langle i, j \rangle$ that denote the substring between positions i and j in the input string. Then the lefthand side of the clause can be replaced with the righthand side.

DEFINITION 31 (Clause instantiation).

Let $G = (N, T, V, P, S)$ *be a RCG. For a given clause* $c = A_0(\vec{\alpha_0}) \rightarrow A_1(\vec{\alpha_1}) \cdots A_m(\vec{\alpha_m})$ *(*$0 \leq m$*) and a string* $w = t_1 \ldots t_n$

1. *an* instantiation *of* c *with respect to* w *consists of a function* $f : \{t' \mid t'$ *is an occurrence of some* $t \in T$ *in the clause*$\} \cup V \cup \{Eps_{i,j} \mid 1 \leq i \leq m, 1 \leq j \leq dim(A_i), \vec{\alpha_i}(j) = \varepsilon\} \rightarrow \{\langle i, j \rangle \mid i \leq j, i, j \in \mathbb{N}\}$ *such that*

Simple RCG:

$S(XYZ) \to A(X,Y,Z)$
$A(aX,aY,aZ) \to A(X,Y,Z)$
$A(bX,bY,bZ) \to A(X,Y,Z)$
$A(a,a,a) \to \varepsilon$
$A(b,b,b) \to \varepsilon$

derivation for *ababab*:

$$\begin{aligned} S(\langle 0,6\rangle) &\Rightarrow A(\langle 0,2\rangle,\langle 2,4\rangle\langle 4,6\rangle) \\ &\Rightarrow A(\langle 1,2\rangle,\langle 3,4\rangle\langle 5,6\rangle) \\ &\Rightarrow \varepsilon \end{aligned}$$

Figure 17. A sample simple RCG for $\{www \mid w \in \{a,b\}^+\}$

a) for all occurrences t' of a $t \in T$ in the clause: $f(t') = \langle i, i+1\rangle$ for some $i, 0 \le i < n$ such that $t_i = t$,

b) for all $X \in V$: $f(X) = \langle j,k\rangle$ for some $0 \le j \le k \le n$,

c) for all x, y adjacent in one of the elements of $\vec{\alpha_i}$ $(0 \le i \le m)$, there are l, j, r with $f(x) = \langle l, j\rangle, f(y) = \langle j, r\rangle$, and
We define then $f(xy) = \langle l, r\rangle$.

d) for all $Eps \in \{Eps_{i,j} \mid 1 \le i \le m, 1 \le j \le dim(A_i), \vec{\alpha_i}(j) = \varepsilon\}$, there is a k, $0 \le k \le n$ with $f(Eps) = \langle k,k\rangle$.
We define then for every ε-argument $\vec{\alpha_i}(j)$ that $f(\vec{\alpha_i}(j)) = f(Eps_{i,j})$.

2. if f is an instantiation of c, then $A_0(f(\vec{\alpha_0})) \to A_1(f(\vec{\alpha_1}))\cdots A_m(f(\vec{\alpha_m}))$ is an instantiated clause *where $f(\langle x_1,\ldots,x_k\rangle) = \langle f(x_1),\ldots,f(x_k)\rangle$.*

The derivation relation is defined as follows:

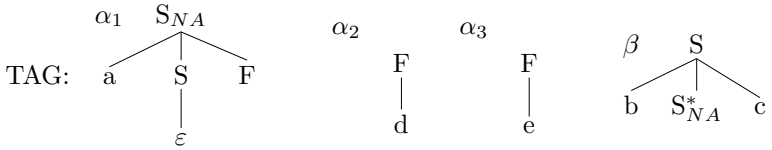
DEFINITION 32 (RCG derivation and string language).

– *For a predicate A of arity k, a clause $A(\ldots) \to \ldots$, and ranges $\langle i_1, j_1\rangle, \ldots, \langle i_k, j_k\rangle$ with respect to a given w: if there is a instantiation of this clause with lefthand side $A(\langle i_1, j_1\rangle, \ldots, \langle i_k, j_k\rangle)$, then in one derivation step $(\ldots \Rightarrow \ldots)$ $A(\langle i_1, j_1\rangle, \ldots, \langle i_i, j_k\rangle)$ can be replaced with the righthand side of this instantiation. $\overset{*}{\Rightarrow}$ is the reflexive transitive closure of $\Rightarrow$.*

– *The language of an RCG G is*
$L(G) = \{w \mid S(\langle 0, |w|\rangle) \overset{*}{\Rightarrow} \varepsilon$ *with respect to* $w\}$.

Fig. 17 shows a sample RCG generating the double copy language with the derivation for the input word *ababab*.

5.1.2. *Constructing an equivalent 2-SRCG for a given TAG*
Now let us give the transformation from TAG to RCG, following (Boullier, 1998; Boullier, 1999): The RCG contains predicates $\langle\alpha\rangle(X)$ and $\langle\beta\rangle(L,R)$ for initial and auxiliary trees respectively. X covers the yield of α and all trees added to α by adjunction or substitution, while L and R cover those parts of the yield of β (including all trees added to β by adjunction or substitution) that are to the left and the right of the foot node of β. The clauses in the RCG reduce the argument(s) of these predicates by identifying those parts that come from the elementary tree α/β itself and those parts that come from one of the elementary trees added by substitution or adjunction. A sample TAG with an equivalent RCG is shown in Fig. 18.



Equivalent RCG:

- $S(X) \rightarrow \langle\alpha_1\rangle(X)$ (every word in the language is the yield of a tree derived from α_1)
- $\langle\alpha_1\rangle(aF) \rightarrow \langle\alpha_2\rangle(F) \mid \langle\alpha_3\rangle(F)$ (either the yield of α_1 is a followed by the yield of tree that substitutes at F)
- $\langle\alpha_1\rangle(aB_1B_2F) \rightarrow \langle\beta\rangle(B_1,B_2)\langle\alpha_2\rangle(F) \mid \langle\beta\rangle(B_1,B_2)\langle\alpha_3\rangle(F)$ (or β adjoins to S in α; then the yield is a followed by the left part of β, the right part of β and the tree substituted at F)
- $\langle\beta\rangle(B_1b,cB_2) \rightarrow \langle\beta\rangle(B_1,B_2)$ (β can adjoin to its root; then the left part is the left part of the adjoined β follwed by b; the right part is c followed by the right part of the adjoined β)
- $\langle\alpha_2\rangle(d) \rightarrow \varepsilon \quad \langle\alpha_3\rangle(e) \rightarrow \varepsilon \quad \langle\beta\rangle(b,c) \rightarrow \varepsilon$ (the yields of α_2, α_3 and β can be d, e and the pair b (left) and c (right) resp.)

Figure 18. A sample TAG and an equivalent RCG

In order to make the choice of an adjoined/substituted tree locally for every node and not at once for the entire elementary tree, Boullier introduces additional so-called *branching predicates* $\langle adj,\gamma,p\rangle$ and $\langle subst,\gamma,p\rangle$ that correspond to the edges in derivation trees. E.g., in the example in Fig. 18, the clauses $\langle\alpha_1\rangle(aB_1B_2F) \rightarrow \langle\beta\rangle(B_1,B_2)\langle\alpha_2\rangle(F) \mid \langle\beta\rangle(B_1,B_2)\langle\alpha_3\rangle(F)$ would be replaced with $\langle\alpha_1\rangle(aB_1B_2F) \rightarrow \langle adj,\alpha_1,2\rangle(B_1,B_2)\langle sub,\alpha_1,3\rangle(F)$, $\langle adj,\alpha_1,2\rangle(X,Y) \rightarrow \langle\beta\rangle(X,Y)$ and $\langle sub,\alpha_1,3\rangle(X) \rightarrow \langle\alpha_2\rangle(X) \mid \langle\alpha_3\rangle(X)$.

Since the yields of initial trees require unary predicates while the yields of auxiliary trees require binary predicates, the maximal predicate arity in the resulting simple RCG is 2. Furthermore, since we encode the yields of elementary trees always from left to right, we obtain an ordered simple 2-RCG.

More precisely, the construction goes as follows:

We define the decoration string σ_γ of an elementary tree γ as in (Boullier, 1999): each internal node has two variables L and R and each substitution node has one variable X (L and R represent the left and right parts of the yield of the adjoined tree and X represents the yield of a substituted tree). In a top-down-left-to-right traversal the left variables are collected during the top-down traversal, the terminals and variables of substitution nodes are collected while visiting the leaves and the right variables are collected during bottom-up traversal. Furthermore, while visiting a foot node, a separating "," is inserted. The string obtained in this way is the decoration string.

1. We add a start predicate S and clauses $S(X) \rightarrow \langle\alpha\rangle(X)$ for all $\alpha \in I$ with root label S.

2. For every $\gamma \in I \cup A$: Let L_p, R_p be the left and right symbols in σ_γ for the node at position p if this is not a substitution node. Let X_p be the symbol for the node at position p if this is a substitution node.

   We assume that $p_1, \ldots, p_k$ are the possible adjunction sites, $p_{k+1}, \ldots, p_l$ the substitution sites in γ. Then the RCG contains all clauses

   $$\langle\gamma\rangle(\sigma_\gamma) \rightarrow \langle adj, \gamma, p_1\rangle(L_{p_1}, R_{p_1}) \ldots \langle adj, \gamma, p_k\rangle(L_{p_k}, R_{p_k}) \\ \langle sub, \gamma, p_{k+1}\rangle(X_{p_{k+1}}) \ldots \langle sub, \gamma, p_l\rangle(X_{p_l})$$

3. For all predicates $\langle adj, \gamma, p\rangle$ the RCG contains all clauses

   $\langle adj, \gamma, p\rangle(L, R) \rightarrow \langle\gamma'\rangle(L, R)$ such that γ' can be adjoined at position p in γ.

4. For all predicates $\langle adj, \gamma, p\rangle$ where $f_{OA}(node(\gamma, p)) = 0$ (adjunction not obligatory), the RCG contains a clause $\langle adj, \gamma, p\rangle(\varepsilon, \varepsilon) \rightarrow \varepsilon$.

5. For all predicates $\langle sub, \gamma, p\rangle$ and all γ' that can be substituted into position p in γ the RCG contains a clause $\langle sub, \gamma, p\rangle(X) \rightarrow \langle\gamma'\rangle(X)$.

5.1.3. *Parsing ordered simple RCG*

There are various parsing algorithms for simple ordered RCG: the top-down parsing algorithm from (Boullier, 2000), the CYK algorithms

introduced for LCFRS in (Burden and Ljunglöf, 2005) that could be adapted to RCG and the Thread-Automaton implementation for ordered simple RCG presented in (Villemonte de La Clergerie, 2002). The latter amounts to an automaton-based implementation of an incremental Earley-style algorithm. Based on these ideas we will present an incremental Earley-style chart parser for ordered simple RCG using the framework of parsing as deduction (Shieber et al., 1995).

The general idea is that we process the arguments of the lefthand sides of clauses incrementally, starting from an S-clause. Whenever we reach a variable, we move into the clause of the corresponding rhs predicate (**predict** or **resume**). Whenever we reach the end of an argument, we **suspend** this clause and move into the parent clause that has called the current one.

Our items have the form

$$[A(\vec{\phi}) \rightarrow A_1(\vec{\phi_1}) \dots A_m(\vec{\phi_m}), pos, \langle i, j\rangle, \vec{\rho}]$$

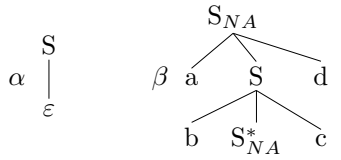
where

- $A(\vec{\phi}) \rightarrow A_1(\vec{\phi_1}) \dots A_m(\vec{\phi_m})$ is a clause;
- $pos \in \{0, \dots, n\}$ is the position up to which we have processed the input;
- $\langle i, j\rangle \in \mathbb{N}^2$ marks the position of our dot in the arguments of the predicate A: $\langle i, j\rangle$ indicates that we have processed the arguments up to the jth element of the ith argument;
- $\vec{\rho}$ is an range vector containing the bindings of the variables occurring in the clause. $\vec{\rho}(i)$ is the range the i-th variable in the lefthand side is bound to.) When first predicting a clause, $\vec{\rho}$ is initialized with a vector containing only symbols "?" for "unknown". We call such a vector (of appropriate arity) $\vec{\rho}_{init}$. We write $\vec{\rho}(X)$ for the range bound to the variable X in $\vec{\rho}$.

Applying a range vector $\vec{\rho}$ containing variable bindings to a $\alpha \in (T \cup V)^*$ means mapping every variable X to $\vec{\rho}(X)$ and concatenating adjacent ranges.

We say that two $\alpha_1, \alpha_2 \in (T \cup R)^*$ where R is the set of ranges over w are compatible iff we can find instantiations $f_1, f_2 : (T \cup R) \rightarrow R$ such that

- $f_i(r) = r$ for every $r \in R$ and for $1 \leq i \leq 2$,
- $f_i(t) = r$ with $r(w) = t$ for $1 \leq i \leq 2$,
- $f_i(xy) = f_i(x)f_i(y)$ for all $x, y \in T \cup R$ for $1 \leq i \leq 2$ and
- $f_1(\alpha_1) = f_2(\alpha_2)$.



$$S(X) \to \langle\alpha\rangle(X)$$
$$\langle\alpha\rangle(LR) \to \langle adj, \alpha, \varepsilon\rangle(L, R)$$
$$\langle\beta\rangle(aLb, cRd) \to \langle adj, \beta, 2\rangle(L, R)$$
$$\langle adj, \alpha, \varepsilon\rangle(L, R) \to \langle\beta\rangle(L, R)$$
$$\langle adj, \beta, 2\rangle(L, R) \to \langle\beta\rangle(L, R)$$
$$\langle adj, \alpha, \varepsilon\rangle(\varepsilon, \varepsilon) \to \varepsilon$$
$$\langle adj, \beta, 2\rangle(\varepsilon, \varepsilon) \to \varepsilon$$

Successful items obtained from parsing $w = abcd$:

	dotted clause	*pos*	bindings $\vec{\rho}$	operation
1	$S(\bullet X) \to \langle\alpha\rangle(X)$	0	$[?]$	
2	$\langle\alpha\rangle(\bullet LR) \to \langle adj, \alpha, \varepsilon\rangle(L, R)$	0	$[?, ?]$	predict
3	$\langle adj, \alpha, \varepsilon\rangle(\bullet L, R) \to \langle\beta\rangle(L, R)$	0	$[?, ?]$	predict
4	$\langle\beta\rangle(\bullet aLb, cRd) \to \langle adj, \beta, 2\rangle(L, R)$	0	$[?, ?]$	predict
5	$\langle\beta\rangle(a \bullet Lb, cRd) \to \langle adj, \beta, 2\rangle(L, R)$	1	$[?, ?]$	scan
6	$\langle adj, \beta, 2\rangle(\bullet, \varepsilon) \to \varepsilon$	1	$[]$	predict
7	$\langle\beta\rangle(aL \bullet b, cRd) \to \langle adj, \beta, 2\rangle(L, R)$	1	$[\langle 1, 1\rangle, ?]$	suspend
8	$\langle\beta\rangle(aLb\bullet, cRd) \to \langle adj, \beta, 2\rangle(L, R)$	2	$[\langle 1, 1\rangle, ?]$	scan
9	$\langle adj, \alpha, \varepsilon\rangle(L\bullet, R) \to \langle\beta\rangle(L, R)$	2	$[\langle 0, 2\rangle, ?]$	suspend
10	$\langle\alpha\rangle(L \bullet R) \to \langle adj, \alpha, \varepsilon\rangle(L, R)$	2	$[\langle 0, 2\rangle, ?]$	suspend
11	$\langle adj, \alpha, \varepsilon\rangle(L, \bullet R) \to \langle\beta\rangle(L, R)$	2	$[\langle 0, 2\rangle, ?]$	resume
12	$\langle\beta\rangle(aLb, \bullet cRd) \to \langle adj, \beta, 2\rangle(L, R)$	2	$[\langle 1, 1\rangle, ?]$	resume
13	$\langle\beta\rangle(aLb, c \bullet Rd) \to \langle adj, \beta, 2\rangle(L, R)$	3	$[\langle 1, 1\rangle, ?]$	scan
14	$\langle adj, \beta, 2\rangle(\varepsilon, \bullet) \to \varepsilon$	3	$[]$	resume
15	$\langle\beta\rangle(aLb, cR \bullet d) \to \langle adj, \beta, 2\rangle(L, R)$	3	$[\langle 1, 1\rangle, \langle 3, 3\rangle]$	suspend
16	$\langle\beta\rangle(aLb, cRd\bullet) \to \langle adj, \beta, 2\rangle(L, R)$	3	$[\langle 1, 1\rangle, \langle 3, 3\rangle]$	scan
17	$\langle adj, \alpha, \varepsilon\rangle(L, R\bullet) \to \langle\beta\rangle(L, R)$	4	$[\langle 0, 2\rangle, \langle 2, 4\rangle]$	suspend
18	$\langle\alpha\rangle(LR\bullet) \to \langle adj, \alpha, \varepsilon\rangle(L, R)$	4	$[\langle 0, 2\rangle, \langle 2, 4\rangle]$	suspend
19	$S(X\bullet) \to \langle\alpha\rangle(X)$	4	$[\langle 0, 4\rangle]$	

Figure 19. A sample parsing trace

Note that in all the compatibility checks needed below, one of the α_1, α_2 is a range, i.e., does not contain terminals.

We start by predicting the S-predicate:

$$\frac{}{[S(\vec{\phi}) \to \vec{\Phi}, 0, \langle 1, 0\rangle, \vec{\rho}_{init}]} \; S(\vec{\phi}) \to \vec{\Phi} \text{ a clause}$$

Scan: Whenever the next symbol after the dot is the next terminal in the input, we can scan it:

$$\frac{[A(\vec{\phi}) \to \vec{\Phi}, pos, \langle i, j\rangle, \vec{\rho}]}{[A(\vec{\phi}) \to \vec{\Phi}, pos+1, \langle i, j+1\rangle, \vec{\rho}]} \quad \vec{\phi}(i, j+1) = w_{pos+1}$$

Predict: Whenever our dot is left of a variable that is the first argument of some righthand side predicate B, we predict new B-clauses:

$$\frac{[A(\vec{\phi}) \to \ldots B(X, \ldots) \ldots, pos, \langle i, j\rangle, \vec{\rho}_A]}{[B(\vec{\psi}) \to \vec{\Psi}, pos, \langle 1, 0\rangle, \vec{\rho}_{init}]}$$

where $\vec{\phi}(i, j+1) = X$ and $B(\vec{\psi}) \to \vec{\Psi}$ is a clause.

Suspend: Whenever we arrive at the end of an argument, we suspend the processing of this clause and we go back to the item that was used to predict it.

$$\frac{[B(\vec{\psi}) \to \vec{\Psi}, pos', \langle i, j\rangle, \vec{\rho}_B], [A(\vec{\phi}) \to \ldots B(\vec{\xi}) \ldots, pos, \langle k, l\rangle, \vec{\rho}_A]}{[A(\vec{\phi}) \to \ldots B(\vec{\xi}) \ldots, pos', \langle k, l+1\rangle, \vec{\rho}]}$$

where

- $|\vec{\psi}(i)| = j$ (the ith argument has length j and has therefore been completely processed),
- $\vec{\rho}_B(\vec{\psi}(i))$ (contains ranges and terminals) compatible with the range $\langle pos, pos'\rangle$,
- and for all $1 \leq h < i$: $\vec{\rho}_B(\vec{\psi}(h))$ (contains ranges and terminals) compatible with $\vec{\rho}_A(\vec{\xi}(h))$ (a range).

$\vec{\rho}$ is $\vec{\rho}_A$ updated with $\vec{\rho}_A(\vec{\xi}(i)) = \langle pos, pos'\rangle$.

Resume: Whenever we are left of a variable that is not the first argument of one of the righthand side predicates, we resume the clause of the righthand side predicate.

$$\frac{[A(\vec{\phi}) \to \ldots B(\vec{\xi}) \ldots, pos, \langle i, j\rangle, \vec{\rho}_A], [B(\vec{\psi}) \to \vec{\Psi}, pos', \langle k-1, l\rangle, \vec{\rho}_B]}{[B(\vec{\psi}) \to \vec{\Psi}, pos, \langle k, 0\rangle, \vec{\rho}_B]}$$

where

- $\vec{\phi}(i)(j+1) = \vec{\xi}(k), k > 1$ (the next element is a variable that is the kth element in $\vec{\xi}$, i.e., the kth argument of B),
- $|\vec{\psi}(k-1)| = l$, and
- $\vec{\rho}_A(\vec{\xi}(h))$ (a range) and $\vec{\rho}_B(\vec{\psi})(h)$ (a sequence of ranges and terminals) are compatible for all $1 \leq h \leq k-1$.

The goal items have the form $[S(\vec{\phi}) \rightarrow \vec{\Phi}, n, \langle 1, j\rangle, \psi]$ with $|\vec{\phi}(1)| = j$ (i.e., the dot is at the end of the lefthand side arguments).

With a dynamic programming interpretation, i.e., implemented as a chart parser, this algorithm can run in polynomial time (Villemonte de La Clergerie, 2002).

Fig. 19 shows a sample TAG, the corresponding RCG and a sample parse trace.

So far, we have described a recognizer. We can extend it to a parser if we add backpointers to the items in the resulting chart. The possible RCG derivation trees can then be obtained by starting from a goal item and following the backpointers. We can immediately obtain TAG derivation trees if, while following the backpointers, every $\langle\gamma\rangle$-predicate for some elementary tree γ with the dot at the right of the last argument is considered representing a completely recognized tree γ and every $\langle adj, \gamma, p\rangle$-predicate with the dot at the right of the last argument represents a completed adjunction at the node at address p in γ. In other words, while traversing the chart following the backpointers, the $\langle\gamma\rangle$-predicates translate into derivation tree nodes while the branching predicates $\langle adj, \gamma, p\rangle$ and $\langle sub, \gamma, p\rangle$ translate into derivation tree edges.

5.2. Checking constraints on MCTAG derivations

In this section, we will sketch how to check the conditions on the derivation tree imposed by the various types of MCTAG seen in this paper. In particular, the conditions characterized via counts of elementary trees used in a derivation can be checked during RCG-parsing if the items are extended with corresponding counters.

5.2.1. *Multicomponent condition*

In order to check for the multicomponent condition, we only need to make sure that different elementary trees from the same elementary tree set in the grammar occur the same number of times in the derivation. Crucially, we do not need to determine which of the different tree instances belong to the same set instance. If we have a lexicalized MCTAG, which means that each elementary tree set contains at least one tree with a leaf that has a terminal label, the number of nodes in the derivation tree is limited by cn where n is the length of the input and c is the maximal cardinality of elementary tree sets in the grammar. Furthermore, we know that at most n different tree set instances are used.

One way to add this additional check is to extend our items during parsing with a counter of the elementary trees already used. This idea is

pursued in (Kallmeyer and Satta, 2009) in order to check the conditions for TT-MCTAG.

Let $c_{\mathcal{A}} = |\mathcal{A}|$, $c_\Gamma = max_{\Gamma \in \mathcal{A}}|\Gamma|$. We assume that $\mathcal{A}$ is ordered and also each of the $\Gamma \in \mathcal{A}$ is ordered. In other words, we treat the element of $\mathcal{A}$ as vectors rather than unordered sets. The same view on elementary tree sets is used in (Rambow, 1994; Kallmeyer and Satta, 2009; Nesson et al., 2008). Let Γ_i denote the ith tree set in $\mathcal{A}$ and $\gamma_{i,j}$ the jth tree in Γ_i. Our counter is then a $c_{\mathcal{A}} \times c_\Gamma$-array T. $T_1 + T_2$ for two counters is defined in the usual componentwise way.

We impose as a general condition on our items that

$$(7) \quad \sum_{1 \leq i \leq c_{\mathcal{A}}} max_{1 \leq j \leq |\Gamma_i|} T(i,j) \leq n$$

where n is the length of the input. This is the condition resulting from lexicalization which checks that the number of tree set instances used in the derivation is not greater than n. (For every set Γ_i, $max_{1 \leq j \leq |\Gamma_i|} T(i,j)$ gives us the number of Γ_i instances used so far.)

Let us call T^0 the counter with $T(m,n) = 0$ for all $1 \leq m \leq c_{\mathcal{A}}$ and $1 \leq n \leq c_\Gamma$, and for every pair i,j with $1 \leq i \leq c_{\mathcal{A}}$ and $1 \leq j \leq |\Gamma_i|$, $T^{i,j}$ denotes the counter with value 1 for $T(i,j)$ and value 0 in all other cases.

During parsing, the counters of items introduced for new predicted clauses get initialized with values 0 for all trees. The counter of an item gets modified whenever we are at the end of the last lefthand side argument in a clause and perform a **suspend**. At that moment, in the corresponding TAG derivation tree, we have completed a subtree and we move up. We then have to add the counter of this subtree to the counter of the mother item.

For our deduction rules, we obtain the following: The start rule and the rule **predict** introduce initial counters T^0 for their consequent items. The rules **scan** and **resume** just pass the counters unchanged. Concerning **suspend**, we have to distinguish two cases: if $|\psi| > i$ (the argument we have finished is not the last), we pass the counter unchanged. Otherwise, the consequent item receives the sum of the two counters of the antecedent items. Furthermore, if the completed clause is a $\langle\gamma\rangle$-clause, we increment the count of γ in the counter of the consequent item. Let us call the second case **suspend-completed**. For **suspend**, we require that $|\psi| > i$. The new deduction rule for the case of the completed clause is as follows:

Suspend-completed:

$$\frac{[B(\vec{\psi}) \rightarrow \vec{\Psi}, pos', \langle i,j\rangle, \vec{\rho}_B, T_B], [A(\vec{\phi}) \rightarrow \ldots B(\vec{\xi}) \ldots, pos, \langle k,l\rangle, \vec{\rho}_A, T_A]}{[A(\vec{\phi}) \rightarrow \ldots B(\vec{\xi}) \ldots, pos', \langle k, l+1\rangle, \vec{\rho}, T'_A]}$$

where in addition to the side condition for **suspend**, we require that $|\psi| = i$ and T'_A is defined as follows:

(a) If B is of the form $\langle \gamma_{i,j} \rangle$ for some $\gamma_{i,j} \in I \cup A$, then $T'_A = T_B + T_A + T^{i,j}$.

(b) If B is of the form $\langle adj \ldots \rangle$ or $\langle sub \ldots \rangle$, then $T'_A = T_B + T_A$.

The multicomponent condition (MC) can then be implemented as an additional check on the goal item, requiring that

(8) for all i, $1 \leq i \leq c_{\mathcal{A}}$, $T(i,j) = T(i,k)$ for all $1 \leq j < k \leq |\Gamma_i|$.

The result is still a polynomial parsing algorithm since the number of possible counters is polynomial in the input length. More precisely, it is $\mathcal{O}(n^{c_G})$ where $c_G = \sum_{i=1}^{|\mathcal{A}|} |\Gamma_i|$.

This confirms the fact that lexicalized Vector MCTAG are polynomially parsable, a result that was already conjectured in (Rambow, 1994).

5.2.2. *Tree-locality*

Concerning tree-locality (TL), we do not need a counter of all elementary trees used throughout the derivation. It is enough to count the daughters for every node in the derivation tree (i.e., every completed $\langle \gamma \rangle$ predicate) and then check whether trees from the same tree set have the same counts. This check is the multicomponent check (8) except that it is performed only on the daughter counters.

We use again a counter T but now it counts only the daughter elementary trees. As above, the start rule and the rule **predict** introduce initial counters T^0 for their consequent items. The rules **scan** **resume** and **suspend** (with the additional condition of $|\psi| > i$) pass the counters unchanged.

The deduction rule **suspend-completed** is now as follows:

$$\frac{[B(\vec{\psi}) \rightarrow \vec{\Psi}, pos', \langle i,j \rangle, \vec{\rho}_B, T_B], [A(\vec{\phi}) \rightarrow \ldots B(\vec{\xi}) \ldots, pos, \langle k,l \rangle, \vec{\rho}_A, T_A]}{[A(\vec{\phi}) \rightarrow \ldots B(\vec{\xi}) \ldots, pos', \langle k, l+1 \rangle, \vec{\rho}, T'_A]}$$

where, as above, in addition to the side condition for **suspend**, we require that $|\psi| = i$. T'_A is now defined as follows:

(a) If B is of the form $\langle \gamma_{i,j} \rangle$ for some $\gamma_{i,j} \in I \cup A$, then $T'_A = T_A + T^{i,j}$.

Furthermore, we require in this case that T_B satisfies the multicomponent check (8).

(b) If B is of the form $\langle adj \ldots \rangle$ or $\langle sub \ldots \rangle$, then $T'_A = T_A + T_B$.

This extension of our RCG-based TAG parser gives us a way to exploit the characterization via TAG derivation trees and the conditions based on counting for parsing tree-local MCTAG. The parsing algorithm we obtain for lexicalized tree-local MCTAG is polynomial in the length of the input since, with our lexicalization condition (7) imposed on the daughter counts, the number of possible daughter counts is polynomial in the length n of the input.

In the unlexicalized case, the size of the daughter counts depends on the number of adjunction and substitution sites in an elementary tree. Therefore, if we limit the number of attachment sites (adjunction sites and substitution nodes) in the elementary trees or, more generally, the number of nodes in the elementary trees of our tree-local MCTAG to some k, then we obtain a polynomial parsing algorithm. This would also limit the number of elementary trees in a tree set to k since tree-sets with more than k elements could not be adjoined to different nodes in a single elementary tree and consequently would be useless.

Nesson et al. (2008) have investigated tree-local MCTAG parsing extensively, showing that the complexity depends crucially on the maximal number of children of a node in the elementary trees. In the approach from (Nesson et al., 2008) that performs parsing on the derived tree, nodes (in a single elementary tree) that require adjunctions or substitutions of trees from the same set instance are linked explicitely. In contrast to this, in our approach, we do not need to know which adjunction or substitution sites (i.e., which daughters of a node in the derivation tree) belong to each other. We only make sure that the daughter count is balanced in the sense of (8).

Note however that the general universal recognition problem for tree-local MCTAG has been shown to be NP-hard (Søgaard et al., 2007; Nesson et al., 2008).

5.2.3. *Delayed tree-locality*

Derivation trees in a k-delayed tree-local MCTAG have to satisfy two conditions, (MC) and (k-DTL-counting). The condition (k-DTL-counting) is formulated in terms of the counts of the elementary trees below every node v in the derivation tree. This can be immediately checked on our counters T where the counters are now again as in the case of checking for the multicomponent condition, i.e., they count the elementary trees occurring in the entire sub-derivation tree below a certain node and they are obtained as shown in section 5.2.1.

The (k-DTL-counting) condition requires that for every node $v \in V$:

$$(9)\ \sum_{\Gamma\in\mathcal{A}}(\max_{\gamma\in\Gamma}|Dom_\gamma(v)|-\min_{\gamma\in\Gamma}|Dom_\gamma(v)|)\leq k$$

Since for every elementary tree $\gamma_{i,j}$, $T(i,j)$ at a specific node gives us $|Dom_\gamma(v)|$, checking the condition on T amounts to requiring that

$$(10)\ \sum_{1\leq i\leq c_\mathcal{A}}(\max_{1\leq j\leq|\Gamma_i|}T(i,j)-\min_{1\leq j\leq|\Gamma_i|}T(i,j))\leq k$$

We have to perform this check whenever we have completely processed a sub-tree in our derivation tree. This is the case when having reached the end of the lefthand side arguments, i.e., when performing a **suspend-completed**. In case (a) (having completed a $\langle\gamma_{i,j}\rangle$-clause), we impose the check (10) on $T_B+T^{i,j}$.

In addition, we have to require that the counter of the goal item satisfies our (MC) check (8).

5.2.4. *SN-tree-tuple locality*
In (Kallmeyer and Satta, 2009), (SN-TTL-counting) is checked during TAG parsing on the derived tree, using a similar counter T as we do. In the following, we will transfer this check to a check on the derivation tree which is actually closer to the characterization given in (SN-TTL-counting).

The first part of the condition captures the fact that heads are higher than their arguments in the derivation tree. We assume that for each $\Gamma_i\in\mathcal{A}$, the first element $\gamma_{i,1}$ is the head tree. Then the first condition on the counters is

$$(11)\ \text{for all } 1\leq i\leq c_\mathcal{A} \text{ and all } 2\leq j\leq|\Gamma_i|\text{: } T_{i,j}\geq T_{i,1}$$

This condition can again be checked in **suspend-completed**, requiring it to hold for $T_B+T^{i,j}$ in the case (a).

The second part of (SN-TTL-counting) treats different cases, one of which must hold if there is a pending argument. Some of these cases refer to the root-adjoining sub-tree. Therefore, the corresponding condition applies when finishing a root adjunction. Since our RCG is ordered and our parser is incremental, the root adjunction is the last adjunction to be finished in an elementary tree. Consequently, when performing a **suspend-completed** whith a consequent $\langle\gamma\rangle$-item that is again a completed item, we can be sure that, if there was an adjunction at the root node, then the B predicate in this **suspend-completed** is of the form $\langle adj,\gamma,\varepsilon\rangle$. Within this rule application, we can therefore acces the counter of the node itself and the counter of the root-adjoining subtree. This is all we need to check the second part of (SN-TTL-counting).

Besides the side condition we already have and the calculation of the counters for the multicomponent check, we add the following additional side conditions to our deduction rule **suspend-completed**:

$$\frac{[B(\vec{\psi}) \to \vec{\Psi}, pos', \langle i,j\rangle, \vec{\rho}_B, T_B], [A(\vec{\phi}) \to \ldots B(\vec{\xi}) \ldots, pos, \langle k,l\rangle, \vec{\rho}_A, T_A]}{[A(\vec{\phi}) \to \ldots B(\vec{\xi}) \ldots, pos', \langle k, l+1\rangle, \vec{\rho}, T'_A]}$$

1. If B is of the form $\langle \gamma_{i,j} \rangle$, then $T_B + T^{i,j}$ must satisfy (11).
2. If the consequent item has a predicate A of the form $\langle \gamma_{i,j} \rangle$ and a clause whose lefthand side has been completely recognized (i.e., the dot is at the end which means $|\phi| = k, |phi(k)| = l+1$), then we require the following. Let T''_A be $T'_A + T^{i,j}$.

   (12) For every $1 \leq l \leq c_{\mathcal{A}}$ and $2 \leq m \leq |\Gamma_l|$ with $T''_A(l,m) > T''_A(l,1)$
   - a) either $\gamma_{l,m} = \gamma_{i,j}$ and $T''_A(l,m) = T''_A(l,1) + 1$;
   - b) or $\gamma_{l,m} = \gamma_{i,j}$ and $T''_A(l,m) > T''_A(l,1) + 1$ and B is a predicate of the form $\langle adj, \gamma_{i,j}, \varepsilon \rangle$ and it holds that $T_B(l,m) - T_B(l,1) + 1 = T''_A(l,m) - T''_A(l,1)$;
   - c) or neither $\gamma_{l,m} = \gamma_{i,j}$ nor $\gamma_{l,1} = \gamma_{i,j}$ and B is a predicate $\langle adj, \gamma_{i,j}, \varepsilon \rangle$ and it holds that $T_B(l,m) - T_B(l,1) = T''_A(l,m) - T''_A(l,1)$;
   - d) or $\gamma_{l,1} = \gamma_{i,j}$ and B is a predicate $\langle adj, \gamma_{i,j}, \varepsilon \rangle$ and it holds that $T''_A(l,m) - T''_A(l,1) \leq T_B(l,m) - T_B(l,1) \leq T''_A(l,m) - T''_A(l,1) + 1$.

With these additional conditions, we capture the conditions for tree-tuple locality with shared nodes, similar to the proposal in (Kallmeyer and Satta, 2009). The resulting parser is still polynomial in the input length.

The condition (TT-k) on k-TT-MCTAG derivation trees requires that there is no $v \in V$ such that

$$(13) \sum_{\beta \in A(G)} (|\{v' \mid l(v') = \beta \text{ and } \langle v, v' \rangle \in E^+\}| - |\{v' \mid l(v') = h(\beta) \text{ and } \langle v, v' \rangle \in E^*\}|) > k$$

This is again a condition we can check immediately on our counters T. It amounts to the following condition:

$$(14) \sum_{1 \leq i \leq c_{\mathcal{A}}} \sum_{2 \leq j \leq |\Gamma_i|} (T(i,j) - T(i,1)) \leq k$$

We can impose this condition again in **suspend-completed**, requiring it to hold in the case (a) for $T_B + T^{i,j}$ if $\gamma_{i,j}$ is a head, otherwise for T_B.

In contrast to this, Kallmeyer and Parmentier (2008) encode the (TT-k) constraint in the RCG that one obtains from transforming the TT-MCTAG. I.e., the RCG predicates are enriched with the list of pending arguments (that, according to (TT-k) is never larger than k). A disadvantage is that the size of the RCG is considerably (actually exponentially) larger than the original MCTAG.

5.2.5. *Checking the simultaneity condition*
It is crucial that for checking (MC), (TT-k), (SN-TTL-counting), (k-DTL-counting) and also (TL), we do not need to group the elementary tree instances into sets.

This is different for (SIM): in order to check (SIM) we have to group the tree instances into set instances. This can be done building the different derivation trees while traversing the derivation forest in the order of a simultaneous multicomponent derivation. This way both, (SIM) and (MC) are checked in once. Obviously, the number of ways to combine tree instances into set instances can explode. This was actually to be expected since the parsing of non-local MCTAG (i.e., MCTAG satisfying (MC) and (SIM)) is NP-hard, even in the lexicalized case (Rambow and Satta, 1992; Champollion, 2007). However, note that this explosion is only the worst case. In a lexicalized MCTAG it depends on the number of times a terminal occurs in the input. In natural languages it is rather rare that words (except some functional operators) occur more than once or twice in a sentence.[12]

6. Conclusion

TAG derivation trees abstract away from the concrete order of derivation steps. A similar abstraction is not possible with the classical MCTAG definition: The simultaneity constraint refers to the process of the derivation itself. Looking only at the result of a derivation (i.e., the derived tree and the derivation tree), simultaneity cannot be checked. In this respect the standard MCTAG definition is problematic. We therefore propose a new declarative definition of MCTAG that characterizes

[12] In TüBa-D/Z, a German newspaper corpus of 15260 sentences (Telljohann et al., 2003), without considering punctuation and determiners, 20% of the sentences contain twice the same word and only 2.5% contain three times the same word. Excluding only punctuation gives 32.3% for twice the same word and 6.6% for three times the same word.

the trees in the tree language via the properties of the TAG derivation trees the MCTAG licences. In this way, in MCTAG like in TAG, the TAG derivation tree can be considered being the central structure of the formalism and the desired abstraction can be obtained. We provide similar declarative definitions for a variety of MCTAG types.

The declarative MCTAG definition enables us to see more clearly the differences between classical MCTAG and variants of them where some of the constraints for the TAG derivation trees need not hold while others are added. In particular, we have shown that tree-local and set-local derivations necessarily satisfy the simultaneity constraint and that every derivation in a k-TT-MCTAG is a k-delayed tree-local derivation.

For some MCTAG conditions, we have provided a definition that does not rely on the grouping of elementary tree instances into sets but that is, instead, based only on counts of the elementary trees occurring in a derivation tree. Such counting conditions are easier to check in the sense that it is enough to add a counter of the elementary trees that label the nodes in the derivation tree and then to check the respective condition on this counter.

We have shown that this way of defining MCTAG can be exploited for parsing: The parsing of the underlying TAG can be done via a grammar that describes the derivation trees, and the additional constraint checking is either performed during the TAG parsing or done separately in a second phase. In particular, for all MCTAG variants that can be defined without the need to refer to the grouping of tree instances into set instances, the parser can be extended with a counter of elementary trees and the condition can be checked on this counter. This way, in the case of lexicalized MCTAG, we obtain polynomial algorithms for Vector TAG, tree-local MCTAG, k-delayed tree-local MCTAG, TT-MCTAG and k-TT-MCTAG.

Acknowledgements

The work presented in this paper was financed by the Deutsche Forschungsgemeinschaft (DFG) with an Emmy Noether Research Grant.

I am particularly grateful to three anonymous reviewers. The paper has considerably benefitted from their very extensive and detailed comments. Furthermore, I would like to thank Wolfgang Maier for proof-reading and for helpful discussions of various aspects of the paper.

Appendix: Proofs

LEMMA 1.

In a non-local MCTAG $G = \langle N, T, S, I, A, \mathcal{A}\rangle$ *with* $G_{TAG} = \langle N, T, S, I, A\rangle$*, the following holds:*

A derivation tree D *of a saturated derived initial tree* t *in* G_{TAG} *is a possible TAG derivation tree in* G *iff* D *satisfies (SIM).*

Proof:

Let G, G_{TAG} and D be as in the lemma.

1. First we show $\Rightarrow$ of the iff:

Let D be a TAG derivation tree licensed in G. To show: D satisfies (SIM).

It is immediate that $l(r) = \alpha \in I$ with $\{\alpha\} \in \mathcal{A}$. For all other nodes $v \in V$ it holds that $l(v) \in I \cup A$.

Furthermore, since we use a fresh instance of a tree set in every derivation step, there must be a partition of V into n sets $V_1, \ldots, V_n$ where for each V_i $(1 \leq i \leq n)$ there is a $\Gamma_i \in \mathcal{A}$ such that $\Gamma_i = \{\gamma \,|\, l(v) = \gamma \text{ for some } v \in V_i\}$ and $|V_i| = |\Gamma_i|$; V_i corresponds to an instance of Γ_i used in a single derivation step. In the following, let $\overline{l(v)}$ denote the instance of $l(v)$ that v corresponds to and $\overline{\Gamma_i}$ the tree set instance of V_i.

We now assume that D with this partition does not satisfy (SIM) and show that this leads to a contradiction.

Assume that

- there is a V_i in the partition with $v_1, v_2 \in V_i$ and $\langle v_1, v_2\rangle \in E^+$. Consequently, $\overline{l(v_2)}$ was added to a tree derived from $\overline{l(v_1)}$. Contradiction since the two trees are added simultaneously.
- or there are pairwise different $V^{(1)}, V^{(2)}, \ldots, V^{(m)} \in \{V_1, \ldots, V_n\}$ with $m \geq 2$ such that there are $n_1^{(i)}, n_2^{(i)} \in V^{(i)}$ $(1 \leq i \leq m)$ such that $\langle n_1^{(1)}, n_2^{(m)}\rangle \in E^*$ and $\langle n_1^{(i)}, n_2^{(i-1)}\rangle \in E^*$ for $2 \leq i \leq m$.

  $\Rightarrow \overline{l(n_1^{(i)})}$ was added before $\overline{l(n_2^{(i-1)})}$ for $2 \leq i \leq m$ and since all elements from $\overline{\Gamma^{(i)}}$ must be added simultaneously for $1 \leq i \leq m$, $\overline{\Gamma^{(m)}}$ was added before $\overline{\Gamma^{(1)}}$. $\Rightarrow \langle n_1^{(1)}, n_2^{(m)}\rangle \notin E^*$. Contradiction.

Consequently, D satisfies (SIM).

2. Then show $\Leftarrow$ of the iff:

Let D be a derivation tree in G_{TAG} satisfying (SIM).

There are different orderings of the derivation steps in D possible: Let the node positions on the derived tree be pairs $\langle \gamma, p\rangle$ with γ being an instance of an elementary tree and p being a position in γ.

Every top-down order read off D (no matter whether (partly) depth first or not and whether left to right or right to left) is a possible derivation order in G_{TAG} for the derivation tree D since in order to perform the derivation step $\ldots[\langle\gamma_1, p\rangle, \gamma_2]$ corresponding to an edge $\langle\gamma_1, \gamma_2, p\rangle$ in D one only needs to make sure that γ_1 (i.e., the mother node of γ_2) was already added.

Because of (SIM) $l(r) = \alpha \in I$ with $\{\alpha\} \in \mathcal{A}$ and the nodes in D can be partitioned into pairwise different elementary tree sets as in (SIM).

To show: there is a top-down traversal of D such that the traversal starts with the root and there is always an element from the partition whose memebers are visited next in any order, i.e., simultaneously.

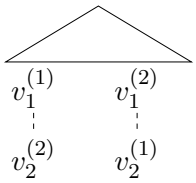
We assume that at some point of the traversal, the choice of a new V_i to be visited next is not possible.

$\Rightarrow$ for each set V_i that has not been visited yet there is at least one $v \in V_i$ whose mother node has not been visited yet (otherwise V_i could be visited next).

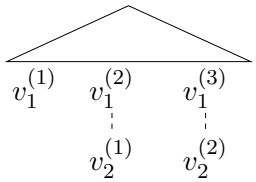
Pick an unvisited $V^{(1)}$ with at least one $v_1^{(1)} \in V^{(1)}$ whose mother node has been visited. Assume $v_2^{(1)} \in V^{(1)}$ with mother not yet visited. Suppose $v_1^{(2)}$ to be the highest unvisited node dominating $v_2^{(1)}$. Since $v_1^{(2)} \neq v_1^{(1)}$ and $\langle v_1^{(2)}, v_1^{(1)}\rangle \in E^+$, (with the first condition of (SIM)) $v_1^{(2)} \in V^{(2)}$ where $V^{(2)} \neq V^{(1)}$ is also an element of the partition.

Then there is a $v_2^{(2)} \in V^{(2)}$ with unvisited mother such that

(a) either $\langle v_1^{(1)}, v_2^{(2)}\rangle \in E^+$. Contradiction to the second condition of (SIM) with $m = 2$.

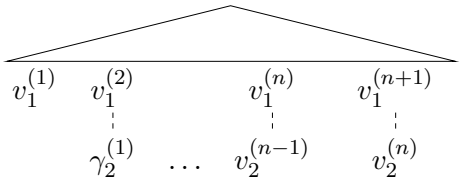


(b) or $\langle v_1^{(1)}, v_2^{(2)}\rangle \notin E^+$. Because of the first condition, we have $\langle v_1^{(2)}, v_2^{(2)}\rangle \notin E^*$. Let $v_1^{(3)} \in V^{(3)}$ be the highest unvisited node dominating $v_2^{(2)}$. ($V^{(3)}$ is another element of the partition.) Because of the first condition of (SIM) $V^{(3)} \neq V^{(2)}$ and because of the second condition $V^{(3)} \neq V^{(1)}$.



In the (b) case there is a $v_2^{(3)} \in V^{(3)}$ with unvisited mother node. Because of (SIM) $\langle v_1^{(1)}, v_2^{(3)}\rangle \notin E^*$, $\langle v_1^{(2)}, v_2^{(3)}\rangle \notin E^*$, and $\langle v_1^{(3)}, v_2^{(3)}\rangle \notin E^*$. Then there is a highest unvisited node $v_1^{(4)} \in V^{(4)}$ dominating $v_2^{(3)}$ with $V^{(4)} \neq V^{(3)}, V^{(4)} \neq V^{(2)}, V^{(4)} \neq V^{(1)}$. And there is a $v_2^{(4)} \in \Gamma_4$ with unvisited mother node.

For each of the $V^{(n)}$, $1 \leq n$ with $v_1^{(n)}, v_2^{(n)}$ as above the situation is as follows: $V^{(n)} \neq V^{(i)}$ for $1 \leq i < n$ (otherwise contradiction to the second part of (SIM)), and $\langle v_1^{(i)}, v_2^{(n)}\rangle \notin E^*$ for $i \leq n$ (otherwise contradiction to the first part os (SIM) for $i = n$ or to the second part of (SIM) for $i \neq n$). Consequently, there is always a new $V^{(n+1)}$ with a new $v_1^{(n+1)}$ being the highest unvisited node dominating $v_2^{(n)}$.



Contradiction to the finiteness of the set of nodes in D.

$\Rightarrow$ there is a top-down traversal of D that corresponds to a multi-component derivation in G in the sense that it allows us to visit the sets of nodes corresponding to the instances of elementary tree sets one after the other.

□

LEMMA 2. *Let $G = \langle N, T, S, I, A, \mathcal{A}\rangle$ be an MCTAG. Let $D = \langle V, E, r\rangle$ be the derivation tree of a saturated derived initial tree in G_{TAG}.*

If D satisfies (SL) then it satisfies (SIM).

Proof: Assume that D as above satisifes (SL) with a partition $V_1, \ldots, V_n$ of V.

To show: D satisfies (SIM) with the same partition.

We assume that this is not the case and show that this assumption leads to a contradiction.

D does not satisfy (SIM) with the partition $V_1, \ldots, V_n$. $\Rightarrow$

- either there is a V_i $(1 \leq i \leq n)$ such that there are $n_0, n_0' \in V_i$, $n_0 \neq n_0'$ with $\langle n_0, n_0' \rangle \in E^*$.
  Then, with (SL), there must be a $V^{(0)} \in \{V_1, \ldots, V_n\}$ such that there are $n_1, n_1' \in V^{(0)}$ with $\langle n_1, n_0 \rangle, \langle n_1', n_0' \rangle \in E$ (and therefore $n_1 \neq n_1'$) and $\langle n_1, n_1' \rangle \in E^*$.
  By induction, with (SL), for all $i \geq 0$ there must be a $V^{(i)} \in \{V_1, \ldots, V_n\}$ and $n_{i+1}, n_{i+1}' \in V^{(i)}$ with $\langle n_{i+1}, n_i \rangle, \langle n_{i+1}', n_i' \rangle \in E$, $\langle n_{i+1}, n_{i+1}' \rangle \in E^*$ and $n_{i+1} \neq n_{i+1}'$.
  This is in contradiction to the finiteness of V.
- or there are pairwise different $V^{(1)}, \ldots, V^{(m)} \in \{V_1, \ldots, V_n\}$ $(m \geq 2)$ such that:
  There are $n_1^{(i)}, n_2^{(i)} \in V^{(i)}$ $(1 \leq i \leq m)$ such that $\langle n_1^{(1)}, n_2^{(m)} \rangle \in E^*$ and $\langle n_1^{(i)}, n_2^{(i-1)} \rangle \in E^*$ for $2 \leq i \leq m$.
  Because of (SL), we can show: For all $V_i, V_j \in \{V_1, \ldots, V_n\}$ with $i \neq j$: if there are $n \in V_i, n' \in V_j$ with $\langle n, n' \rangle \in E^*$, then for all $n_j \in V_j$ there is a $n_i \in V_i$ with $\langle n_i, n_j \rangle \in E^*$.
  This follows immediately from (SL) by induction on the length of the path from n to n'.
  For the $V^{(1)}, \ldots, V^{(m)}$ we consequently obtain: for each $n_m \in V^{(m)}$ there is a $n_1 \in V^{(1)}$ with $\langle n_1, n_m \rangle \in E^*$ and (by induction) for each $n_1 \in V^{(1)}$ there is a $n_m \in V^{(m)}$ with $\langle n_m, n_1 \rangle \in E^*$. This is a contradiction since (with $V^{(1)} \cap V^{(m)} = \emptyset$) this would imply that dominance is not antisymmetric.

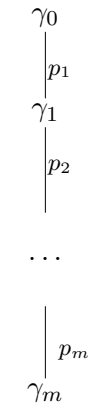
□

LEMMA 4. *Let $D = \langle V_D, E_D, r_D \rangle$ be a TAG derivation tree with $t = \langle V_t, E_t, r_t \rangle$ being a corresponding derived tree.*

Let $n_0, \ldots, n_m \in V_D$ with $\langle n_{i-1}, n_i \rangle \in E_D$ with $g(\langle n_{i-1}, n_i \rangle) = p_i$ for all $1 \leq i \leq m$. Let $l(n_m)$ be an auxiliary tree with foot node f and, furthermore, let k be a node in $l(n_0)$ with address p. Then the following holds:

$\langle \delta_{D,t}(f, n_m), \delta_{D,t}(k, n_0) \rangle \in E_t^*$ iff $\langle n_1, n_m \rangle \in E_{D_{Spine}}^*$ and p_1 is a prefix of p.

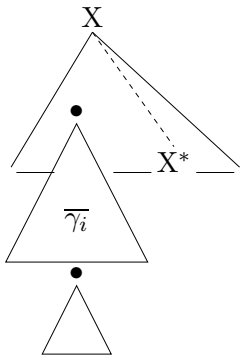
Proof: The configuration in D is as shown on the right where $\gamma_i = l(n_i)$ and $p_i = g(\langle n_{i-1}, n_i \rangle)$ for $1 \leq i \leq m$.



First we show the $\Leftarrow$ part of the iff:
We assume that $\langle n_1, n_m \rangle \in E^*_{D_{Spine}}$ and p_1 is a prefix of p.
Then in the derived tree t, $\delta_{D,t}(f_i, n_i)$ dominates $\delta_{D,t}(f_{i-1}, n_{i-1})$ where f_i is the foot node of γ_i for $1 < i \leq m$. Furthermore, since dominance is transitive, $\delta_{D,t}(f, n_m)$ dominates $\delta_{D,t}(f_1, n_1)$.
Finally, since p_1 is a prefix of p, the adjunction site of the subtree of t corresponding to the subtree of D rooted in n_1 dominates the node $\delta_{D,t}(k, n_0)$. Consequently, $\delta_{D,t}(f_1, n_1)$ dominates $\delta_{D,t}(k, n_0)$ and (by transitivity) $\delta_{D,t}(f, n_m)$ dominates $\delta_{D,t}(k, n_0)$.
Now we show the $\Rightarrow$ direction of the iff:
Assume that the left side of the iff holds. Then we assume that the right side does not hold and we show that this leads to a contradiction.
First assume that p_1 is not a prefix of p. In this case

- either p is a prefix of p_1 and $p \neq p_1$ holds. Then $\delta_{D,t}(k, n_0)$ strictly dominates anything adjoined to the node at position p_1 in γ_1, including the foot node f of γ_m. This is a contradiction to our assumption.
- or p is not a prefix of p_1. Then (since p_1 is no prefix of p either) there is no dominance relation between k and the material adjoined to the node at position p_1 in γ_1. This is also a contradiction.

Then we assume that $\langle n_1, n_m \rangle \notin E^*_{D_{Spine}}$. $\Rightarrow$ there is a $i \in \{1, \ldots, m\}$ such that p_i is not a position on the spine. Consequently, the auxiliary tree derived from γ_{i-1} is either as shown on the right or symmetric to this but with the foot node on the left of $\overline{\gamma_i}$, the tree derived from γ_i.



Adjoining this tree to a tree $\overline{\gamma_0}$ derived from γ_0 by the first $i-1$ adjunctions yields a tree where no node from $\overline{\gamma_i}$ dominates a node from $\overline{\gamma_0}$. In particular, f does not dominate k.

□

References

Becker, T., A. K. Joshi, and O. Rambow: 1991, 'Long-distance Scrambling and Tree Adjoining Grammars'. In: *Proceedings of ACL-Europe*.

Boullier, P.: 1998, 'A Generalization of Mildly Context-Sensitive Formalisms'. In: *Proceedings of the Fourth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+4)*. University of Pennsylvania, Philadelphia, pp. 17–20.

Boullier, P.: 1999, 'On TAG Parsing'. In: *TALN 99, 6^e conférence annuelle sur le Traitement Automatique des Langues Naturelles*. Cargèse, Corse, pp. 75–84.

Boullier, P.: 2000, 'Range Concatenation Grammars'. In: *Proceedings of the Sixth International Workshop on Parsing Technologies (IWPT2000)*. Trento, Italy, pp. 53–64.

Burden, H. and P. Ljunglöf: 2005, 'Parsing Linear Context-Free Rewriting Systems'. In: *IWPT'05, 9th International Workshop on Parsing Technologies*. Vancouver, Canada.

Champollion, L.: 2007, 'Lexicalized non-local MCTAG with dominance links is NP-complete'. In: G. Penn and E. Stabler (eds.): *Proceedings of Mathematics of Language (MOL) 10*.

Chen-Main, J. and A. Joshi: 2007, 'Some Observations on a Graphical Model-Theoretical Approach and Generative Models'. In: *Model Theoretic Syntax at 10. Workshop, ESSLLI 2007*. Dublin, Ireland.

Chiang, D. and T. Scheffler: 2008, 'Flexible Composition and Delayed Tree-Locality'. In: *TAG+9 Proceedings of the ninth International Workshop on Tree-Adjoining Grammar and Related Formalisms (TAG+9)*. Tübingen, pp. 17–24.

Joshi, A. K.: 1987, 'An introduction to Tree Adjoining Grammars'. In: A. Manaster-Ramer (ed.): *Mathematics of Language*. Amsterdam: John Benjamins, pp. 87–114.

Joshi, A. K., L. Kallmeyer, and M. Romero: 2007, 'Flexible Composition in LTAG: Quantifier Scope and Inverse Linking'. In: R. Muskens and H. Bunt (eds.): *Computing Meaning Volume 3*. Kluwer.

Joshi, A. K., L. S. Levy, and M. Takahashi: 1975, 'Tree Adjunct Grammars'. *Journal of Computer and System Science* **10**, 136–163.

Joshi, A. K. and Y. Schabes: 1997, 'Tree-Adjoning Grammars'. In: G. Rozenberg and A. Salomaa (eds.): *Handbook of Formal Languages*. Berlin: Springer, pp. 69–123.

Kallmeyer, L.: 2005, 'Tree-local Multicomponent Tree Adjoining Grammars with Shared Nodes'. *Computational Linguistics* **31**(2), 187–225.

Kallmeyer, L. and A. K. Joshi: 2003, 'Factoring Predicate Argument and Scope Semantics: Underspecified Semantics with LTAG'. *Research on Language and Computation* **1**(1–2), 3–58.

Kallmeyer, L. and Y. Parmentier: 2008, 'On the relation between Multicomponent Tree Adjoining Grammars with Tree Tuples (TT-MCTAG) and Range Concatenation Grammars (RCG)'. In: C. Martín-Vide, F. Otto, and H. Fernaus (eds.): *Language and Automata Theory and Applications. Second International Conference, LATA 2008*, No. 5196 in Lecture Notes in Computer Science. Heidelberg Berlin: Springer-Verlag, pp. 263–274.

Kallmeyer, L. and M. Romero: 2008, 'Scope and Situation Binding in LTAG using Semantic Unification'. *Research on Language and Computation* **6**(1), 3–52.

Kallmeyer, L. and G. Satta: 2009, 'A Polynomial-Time Parsing Algorithm for TT-MCTAG'. In: *Proceedings of ACL*. Singapore.

Kroch, A. S. and A. K. Joshi: 1987, 'Analyzing extraposition in a tree adjoining grammar'. In: G. J. Huck and A. E. Ojeda (eds.): *Syntax and Semantics: Discontinuous Constituency*. Academic Press, Inc., pp. 107–149.

Lichte, T.: 2007, 'An MCTAG with Tuples for Coherent Constructions in German'. In: *Proceedings of the 12th Conference on Formal Grammar 2007*. Dublin, Ireland.

Nesson, R., G. Satta, and S. Shieber: 2008, 'Complexity, Parsing, and Factorization of Tree-Local Multi-Component Tree-Adjoining Grammar'. Technical Report TR-05-08, School of Engineering and Applied Sciences, Harvard University, Cambridge, MA.

Nesson, R. and S. M. Shieber: 2006, 'Simpler TAG Semantics Through Synchronization'. In: *Proceedings of the 11th Conference on Formal Grammar*. Malaga, Spain.

Rambow, O.: 1994, 'Formal and Computational Aspects of Natural Language Syntax'. Ph.D. thesis, University of Pennsylvania.

Rambow, O. and G. Satta: 1992, 'Formal Properties of Non-Locality'. In: *Proceedings of 1st International Workshop on Tree Adjoining Grammars*. Philadelphia.

Seki, H., T. Matsumura, M. Fujii, and T. Kasami: 1991, 'On multiple context-free grammars'. *Theoretical Computer Science* **88**(2), 191–229.

Shieber, S. M., Y. Schabes, and F. C. N. Pereira: 1995, 'Principles and implementation of deductive parsing'. *Journal of Logic Programming* **24**(1&2), 3–36.

Søgaard, A., T. Lichte, and W. Maier: 2007, 'The complexity of linguistically motivated extensions of tree-adjoining grammar'. In: *Recent Advances in Natural Language Processing 2007*. Borovets, Bulgaria.

Telljohann, H., E. W. Hinrichs, and S. Kübler: 2003, 'Stylebook for the Tübingen Treebank of Written German (TüBa-D/Z)'. Seminar für Sprachwissenschaft, Universität Tübingen, Germany.

Vijay-Shanker, K. and D. J. Weir: 1993, 'Parsing some constrained grammar formalisms'. *Computational Linguistics* **19**(4), 591–636.

Vijay-Shanker, K. and D. J. Weir: 1994, 'The Equivalence of Four Extensions of Context-Free Grammars'. *Mathematical Systems Theory* **27**(6), 511–546.

Vijay-Shanker, K., D. J. Weir, and A. K. Joshi: 1987, 'Characterizing structural descriptions produced by various grammatical formalisms'. In: *Proceedings of ACL*. Stanford.

Villemonte de La Clergerie, E.: 2002, 'Parsing Mildly Context-Sensitive Languages with Thread Automata'. In: *Proc. of COLING'02*.

Weir, D. J.: 1988, 'Characterizing mildly context-sensitive grammar formalisms'. Ph.D. thesis, University of Pennsylvania.