

## 3 Tree Adjoining Grammars (TAG)

### 3.1 Definition of LTAG

Joshi (1985, 1987); Joshi and Schabes (1997)

**Definition 9 (Tree Adjoining Grammar)** A Tree Adjoining Grammar (TAG) is a quadruple  $\langle N, T, I, A \rangle^4$  such that

- $T$  and  $N$  are disjoint alphabets, the terminals and nonterminals,
- $I$  is a finite set of initial trees, and
- $A$  is a finite set of auxiliary trees.

The trees in  $I \cup A$  are called elementary trees.

In a *lexicalized TAG (LTAG)* each elementary tree has at least one leaf with a terminal label.

A tree is a derived initial (auxiliary) tree if it has been obtained from an initial (auxiliary) tree by a sequence of substitutions and adjunctions. The tree language  $T(G)$  of a TAG  $G$  is the set of all completed derived initial trees. The string language  $L(G)$  is the set of strings yielded by the trees in the tree language.

**Proposition 7** For each finitely ambiguous CFG  $G = \langle N, T, P, S \rangle$  that does not generate the empty string, there is a strongly equivalent LTAG  $G_{lex} = \langle N, T, I, A \rangle$ .

The proof is a constructive proof. In general, a constructive proof consists of two parts:

1. the construction algorithm for the grammar whose existence one wants to show (here the LTAG for a given CFG)
2. a proof that the constructed grammar has really the desired properties (here being lexicalized and being strongly equivalent to the original CFG) if this is not obvious; this proof is usually done by induction

Proof: Construction of an LTAG  $G_{lex}$  for a given CFG  $G$ : We consider those derivation trees of  $G$  that satisfy the following condition: there are not two nodes  $n_1, n_2$  with the same non-terminal label such that  $n_1$  is not the root,  $n_1 \neq n_2$  and  $n_1$  dominates  $n_2$ . Among these derivation trees, those with root label  $S$  and a terminal yield become the initial trees. Those having one leaf with the same non-terminal label as the root and, besides this, having only leaves with terminal labels become the auxiliary trees where the leaf with non-terminal label is the foot node.

The new grammar is lexicalized because  $A \stackrel{\dagger}{\Rightarrow} A$  and  $A \stackrel{*}{\Rightarrow} \epsilon$  is not possible in  $G$  for any  $A \in N$ .

It is rather obvious from the way the elementary trees were constructed that  $T(G_{lex}) \subseteq T(G)$ .

To show:  $T(G) \subseteq T(G_{lex})$ . In a  $G$  derivation tree, we call a *recursive pair* each pair of nodes  $\langle n_1, n_2 \rangle$  with the same non-terminal label such that  $n_1 \neq n_2$  and  $n_1$  dominates  $n_2$ .

To show: each tree  $T$  in the tree language of  $G$  is in the tree language of  $G_{lex}$ .

Induction on the number  $i$  of recursive pairs in  $T$ .

$i = 0$ :  $T$  is an initial tree in  $G_{lex}$ .

---

<sup>4</sup>Some authors define TAG as quintuple including a start symbol  $S \in N$  besides  $N, T, I$  and  $A$ , and then only derived trees with root label  $S$  are allowed in the tree language. The two definitions are strongly equivalent.

$i \rightarrow i+1$ : Let there be  $i+1$  recursive pairs in  $T$ . Then there is at least one recursive pair  $\langle n_1, n_2 \rangle$  such that: there is no recursive pair  $\langle n'_1, n'_2 \rangle$  such that  $n_1$  dominates  $n'_1$ ,  $n_2$  does not dominate  $n'_2$ ,  $n_2 \neq n'_2$  and either  $n'_2$  dominates  $n_2$  or  $n_1 \neq n'_1$ . (Otherwise take  $\langle n'_1, n'_2 \rangle$  instead.) Let  $p$  be the node address of  $n_1$ .

Let  $T_1, T_2$  be the subtrees of  $T$  with roots  $n_1$  and  $n_2$  respectively. Then there is an auxiliary tree  $\beta$  in  $G_{lex}$  one obtains by taking  $T_1$  and removing all nodes strictly dominated by  $n_2$ . Its foot node is  $n_2$ .

Let  $T'$  be the tree obtained by replacing  $T_1$  in  $T$  with  $T_2$ . Then  $T$  can be derived from  $T'$  adjoining  $\beta$  at address  $p$ .

$T'$  is a derived tree in  $G$  and it has  $\leq i$  recursive pairs. Consequently (induction claim),  $T'$  can be derived in  $G_{lex}$ .

□

Substitutions in a TAG can be precompiled (they involve only a finite number of substitution nodes and a finite number of initial trees). Therefore, the following holds:

**Proposition 8** *For each (lexicalized) TAG there is a strongly equivalent (lexicalized) TAG without substitution nodes.*

TAG as defined above are more powerful than CFG (Joshi 1985) but they cannot generate the copy language  $\{ww \mid w \in \{a, b\}^*\}$ . This is necessary however in order to describe cross-serial dependencies.

In order to increase the expressive power, adjunction constraints were introduced that specify for each node 1. whether adjunction is mandatory and 2. which trees can be adjoined.

**Definition 10 (TAG with adjunction constraints)** *A TAG with adjunction constraints is a tuple  $\langle N, T, I, A, O, C \rangle$  such that*

- $\langle N, T, I, A \rangle$  is a TAG,
- $O : \{\mu \mid \mu \text{ is an internal node or a foot node in a tree in } I \cup A\} \rightarrow \{1, 0\}$  is a function, and
- $C : \{\mu \mid \mu \text{ is an internal node or a foot node in a tree in } I \cup A\} \rightarrow P(A)$  is a function.<sup>5</sup>

If a derived initial tree is substituted for a node, in the new tree the root of the initial tree maintains its adjunction constraints.

In a TAG with adjunction constraints, adjunction of an auxiliary tree derived from the elementary tree  $\beta$  at a node  $\mu$  is allowed iff  $\beta \in C(\mu)$ . In this case, the node  $\mu$  disappears together with its constraints while the root and foot node of the auxiliary tree maintain their adjunction constraints.

A node  $\mu$  with  $O(\mu) = 1$  is said to carry a *obligatory adjunction (OA)* constraint. A node  $\mu$  with  $O(\mu) = 0$  and  $C(\mu) = \emptyset$  is said to carry a *null adjunction (NA)* constraint. A node  $\mu$  with  $O(\mu) = 0$  and  $C(\mu) \neq \emptyset$  and  $C(\mu) \neq A$  is said to carry a *selective adjunction (SA)* constraint.

The tree language is defined as the set of completed derived initial trees containing no node with an OA constraint.

Henceforth we are always using TAGs with adjunction constraints, i.e., whenever the term ‘‘TAG’’ is used it means ‘‘TAG with adjunction constraints’’. Fig. 7 shows a TAG for the copy language.

For each TAG, there is a strongly equivalent TAG with NA constraints on foot nodes. This is easy to see: whenever adjoining  $\beta_1$  to the foot node of  $\beta_2$ , one can as well adjoin  $\beta_2$  to the root of  $\beta_1$ . The derived auxiliary tree is the same.

TAG derivations are described by derivation trees: For each derivation in a TAG there is a corresponding *derivation tree*. This tree contains nodes for all elementary trees used in the derivation. Whenever a

---

<sup>5</sup>For a set  $X$ , we use  $P(X)$  as a notation for the set of all subsets of  $X$ .

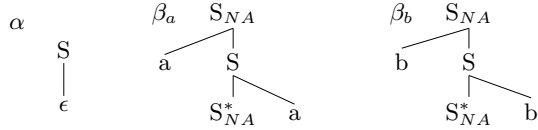


Figure 7: TAG for the copy language

derived tree, derived from some elementary tree  $\gamma$ , was attached (by substitution or adjunction) to the node at address  $p$  in the elementary tree  $\gamma'$ , there is an edge from  $\gamma'$  to  $\gamma$  labeled with  $p$ . We use Gorn addresses  $\mathbb{N}^*$  for nodes:  $\epsilon$  is the address of the root,  $p \cdot i$  the address of the  $i$ th daughter of the node at address  $p$ .

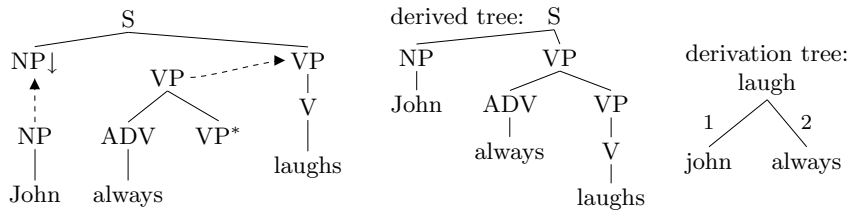


Figure 8: TAG derivation for *John always laughs*

**Exercise 4**  $L_3 := \{a^n b^n c^n \mid n \geq 0\}$

1. Give a TAG (with adjunction constraints) that generates  $L_3$ .
2. Show that TAG without adjunction constraints cannot generate  $L_3$ .  
(Hint: Any elementary tree must contain equal numbers of  $a$ 's,  $b$ 's and  $c$ 's. And each auxiliary tree can be adjoined at its own root.)

### 3.2 Feature Structure Based TAG

Vijay-Shanker (1987); Vijay-Shanker and Joshi (1988)

*Feature-structure based TAG (FTAG)*: each node has a top and a bottom feature structure (except substitution nodes that have only a top). Nodes in the same elementary tree can share features (extended domain of locality).

A FTAG does not have adjunction constraints. Instead, the adjunctions constraints can be expressed using the features.

Unification during derivation (see Fig. 9):

- Substitution: the top of the root of the new initial tree unifies with the top of the substitution node
- Adjunction: the top of the root of the new auxiliary tree unifies with the top of the adjunction site and the bottom of the foot of the new tree unifies with the bottom of the adjunction site.
- In the final derived tree, top and bottom must unify for all nodes.

Important: There is only a finite set of possible feature structures and feature structures are acyclic.

Since nodes in the same elementary tree can share features, constraints among dependent nodes can be more easily expressed. E.g., agreement between subject and verb (see Fig. 10).

**Proposition 9** For each FTAG there exists a weakly equivalent TAG with adjunction constraints and vice versa. The two TAGs generate even the same sets of trees, only with different node labels.

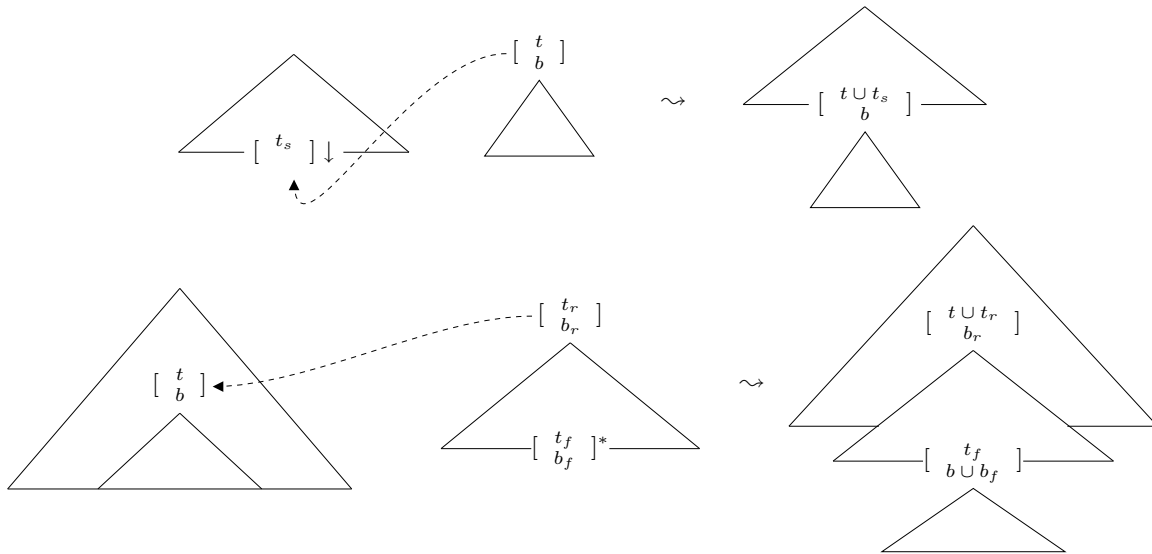


Figure 9: Feature structure unifications in FTAG

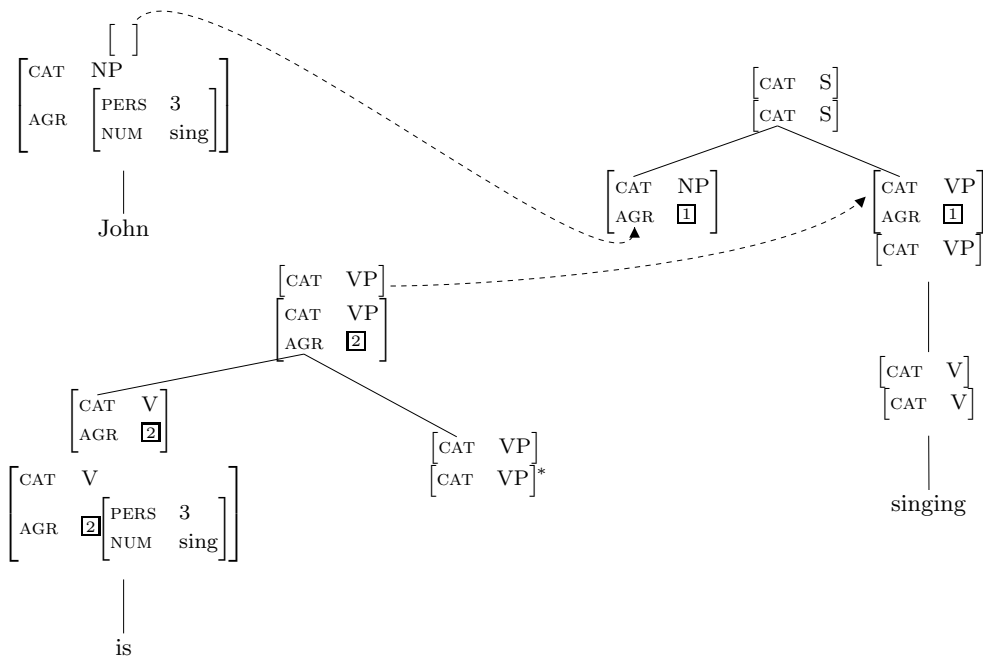


Figure 10: Agreement with feature structures

