

11 Other extensions/variants of TAG

11.1 Synchronous TAG

11.1.1 First definition of synchronous TAG

Shieber and Schabes (1990, 1991)

Idea: pair two TAGs, for example for machine translation (Abeillé et al. 1990) or for the syntax-semantics interface (Shieber and Schabes 1990).

A synchronous TAG consists of triples $\langle \gamma_l, \gamma_r, link \rangle$ where

- γ_l is an elementary tree from some TAG G_l ,
- γ_r is an elementary tree from some TAG G_r , and
- $link$ is a relation between the nodes in γ_l and the nodes in γ_r .

A derivation in a synchronous TAG starts with a pair of two initial trees. In each derivation step:

1. Choose a link between two nodes at positions p_1 and p_2 in the current derived tree pair $\langle \overline{\alpha}_1, \overline{\alpha}_2, L_\alpha \rangle$.
2. Choose a new $\langle \gamma_l, \gamma_r, L_\gamma \rangle$ from the grammar such that γ_l can be adjoined at p_1 in $\overline{\alpha}_1$ and γ_r can be adjoined at p_2 in $\overline{\alpha}_2$.
3. The new derived tree pair is then $\langle \overline{\alpha}_1[p_1, \gamma_l], \overline{\alpha}_2[p_2, \gamma_r], L' \rangle$ where L' is defined as follows:
 L' contains all links from L_α and L_γ except the link that has just been used. Links involving the nodes at positions p_1 and p_2 resp. in the old trees now involve instead the root node of the new elementary trees γ_l and γ_r respectively.

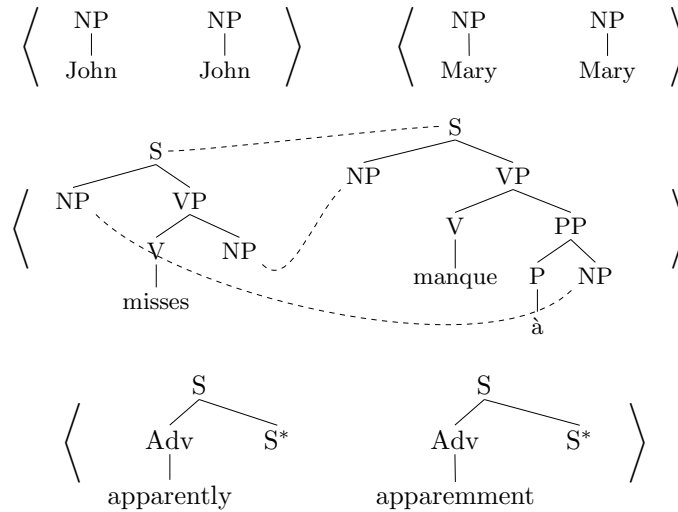


Figure 20: sample synchronous TAG for English-French translation

See for example the synchronous TAG in Fig. 20. With this we can translate (57a.) into (57b.).

- (57) a. Apparently, John misses Mary.
 b. Apparemment, Mary manque à John.

A synchronous TAG generates pairs of trees. The set of the left/right trees from these pairs is called its *left projection language/right projection language*.

11.1.2 Natural definition of synchronous TAG

Shieber (1994)

Intuition behind synchronous TAG: pair two TALs. But with the definition above, the weak generative capacity is increased, i.e., the projection languages of synchronous TAGs are a true superset of TAL (see Fig. 21).

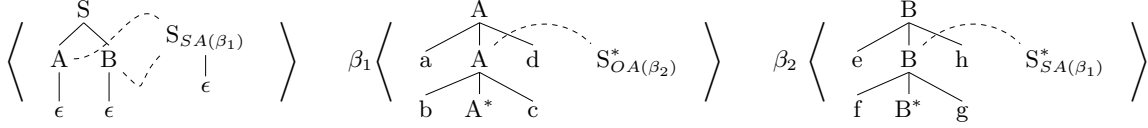


Figure 21: Synchronous TAG for $L_8 = \{a^n b^n c^n d^n e^n f^n g^n h^n \mid n \geq 0\}$ as left projection language

Problem: derivation trees are not isomorphic. See for example derivation trees for derivation of $aabbccddeeffgghh$ in the left and right TAG in Fig. 22.

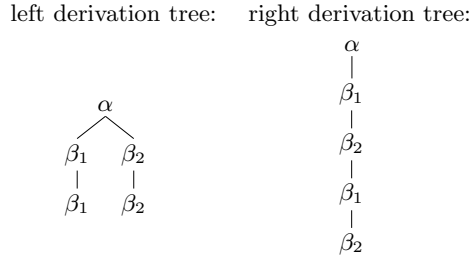


Figure 22: Derivation trees in left and right TAG from Fig. 21 for synchronous derivation of $aabbccddeeffgghh$

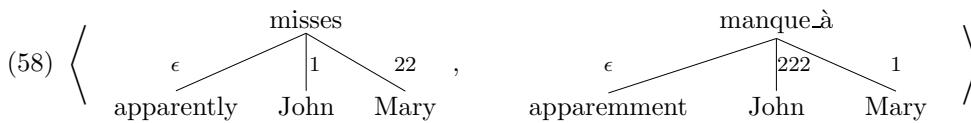
Therefore *natural definition* (Shieber 1994) of synchronous TAG: one derivation tree with pairs of trees as nodes and pairs of addresses as edge labels.

In other words the derivation trees in the left and right TAG must be isomorphic:

Definition of the set of pairs of derivation trees $\langle D_l, D_r \rangle$ licenced by a synchronous TAG:

- D_l is a possible derivation tree in the left TAG,
- D_r is a possible derivation tree in the right TAG,
- D_l and D_r are isomorphic, i.e., there is a bijection f from the nodes of D_l to the nodes of D_r that preserves dominance,
- the isomorphic substitutions/adjunctions are sanctioned by links in tree pairs, i.e.,
 1. if $f(\mu_1) = \mu_2$ and t_1, t_2 are the trees of μ_1 and μ_2 resp., then there is a triple $\langle t_1, t_2, L \rangle$ in the synchronous TAG,
 2. furthermore, if both μ_1 and μ_2 have parents such that t'_1, t'_2 are the trees of the parents and p_1, p_2 are the labels of the corresponding edges resp., then there is a triple $\langle t'_1, t'_2, L' \rangle$ in the synchronous TAG with $\langle p_1, p_2 \rangle \in L'$.

Example: isomorphic derivation trees for the derivation of $\langle (57a.), (57b.) \rangle$:



Proposition 28 *Under the natural definition of synchronous TAG, each left (right) projection language of a synchronous TAG is a TAL.*

Note that for a syntax-semantics interface with synchronous TAGs, the isomorphism requirement means that the derivation tree is the structure that links syntax to semantics, i.e., semantics is computed on the derivation tree.

11.2 Description-based TAG variants

So far, the elements of our grammars were trees, i.e., objects. Substitution and adjunction are operations for combining trees.

Other possibility to define TAG: use tree descriptions instead. Combining two tree descriptions then amounts to conjunction plus additional equalities between node names.

11.2.1 Quasi-trees

Vijay-Shanker (1992); Rogers (1994); Rogers and Vijay-Shanker (1994)

Motivation: non-monotonicity of TAG: in an adjunction, the structural relationships holding in the trees that are combined are not preserved.

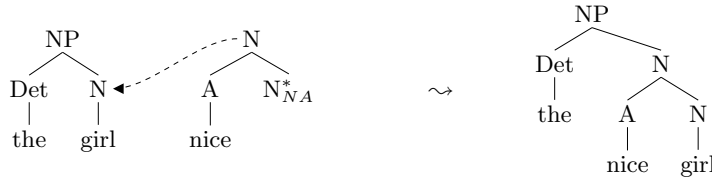
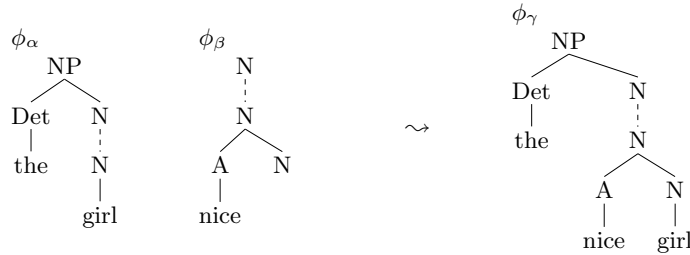


Figure 23: standard TAG adjunction

In Fig. 23 for example, on the path from the NP node to the “girl” node, there is only one N node in the “the girl” tree while there are two N nodes in the “the nice girl” tree. Furthermore, the N node from “the girl” is even deleted in the derivation, i.e., replaced with two new N nodes.

Idea of Vijay-Shanker: replace nodes that allow adjunction with *quasi-nodes*. A quasi-node is a pair of two node variables with a dominance link in between. A tree with quasi-nodes is a tree description that does not fully specify the parent relation between the nodes described in the description. See Fig. 24 for the quasi-trees corresponding to Fig. 23.

Graphical representation:



Corresponding tree descriptions:

$$\begin{aligned}
 \phi_\alpha &= NP(k_1) \wedge Det(k_2) \wedge N(k_3) \wedge k_1 \triangleleft k_2 \wedge k_1 \triangleleft k_3 \wedge k_2 \prec k_3 \wedge N(k_4) \wedge k_3 \triangleleft^* k_4 \wedge \dots \\
 \phi_\beta &= N(k_7) \wedge N(k_8) \wedge k_7 \triangleleft k_8 \wedge A(k_9) \wedge N(k_{10}) \wedge k_8 \triangleleft k_9 \wedge k_8 \triangleleft k_{10} \wedge k_9 \prec k_{10} \dots \\
 \phi_\gamma &= \phi_\alpha \wedge \phi_\beta \wedge k_3 \approx k_7 \wedge k_4 \approx k_{10}
 \end{aligned}$$

Figure 24: quasi-tree TAG adjunction

Semantics of tree descriptions can be defined

- either in a standard predicate logic. The models are then sets of nodes with unary labelling predicates and binary predicates immediate dominance, dominance and linear precedence denoted by \triangleleft , \triangleleft^* and \prec . \approx denotes equality.
- or in a modal logic with the nodes being the worlds and the binary relations being the accessibility relations leading from one world to another.

For TAG, one needs the existential fragment of first order logic with an additional operation for putting tree descriptions together. This operation builds the conjunction while adding further equivalences between node variables.

11.2.2 D-Tree Substitution Grammars (DSG)

Rambow et al. (2001)

D-Tree Substitution Grammars (DSG) are roughly a description-based formalization of a kind of non-local MCTAG with dominance links. However, in contrast to MCTAG, they do not allow adjunction-like operations.

Elements of the grammar: special tree descriptions called *d-trees*. A d-tree consists of n fully-specified tree descriptions called *components* plus dominance links between the leaf of one of these tree descriptions and the root of another one. A d-tree is always connected. Different components denote different subtrees.

Sample d-trees are ϕ_α and ϕ_β in Fig. 24.

The operation for combining two d-trees is such that a leaf in one of the d-trees is identified with the root of a component in the other d-tree. I.e., the operations are substitutions.

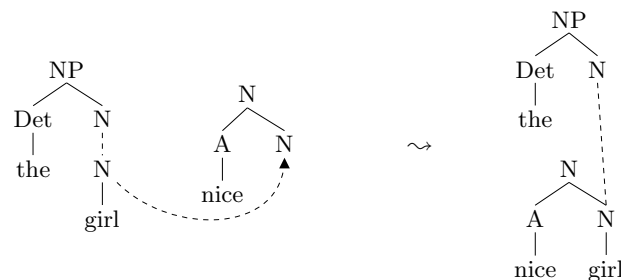


Figure 25: Sample d-tree substitution

DSGs are polynomially parsable.

DSG allow to deal with scrambling in a way similar to V-TAG. But since they are non-local they also need an additional mechanism for locality constraints (so-called *path constraints*).

For cross-serial dependencies in TAG, i.e., for the copy language, it was crucial to adjoin to the spine of auxiliary trees. In DSG, introducing a dominance link leads to much more freedom in the order of the terminals. This is why Rambow et al. (2001) suspect that DSG might not be able to generate the copy language.

11.2.3 Tree Description Grammars (TDG)

Kallmeyer (1999)

Goal: allow to derive tree descriptions that are underspecified wrt. dominance (not only immediate dominance). This can then be used for PP-attachment ambiguities or for scope ambiguities.

In the grammar, the tree descriptions look like d-trees. Different components denote also different subtrees.

But: adjunction-like operations possible: in a derivation step

- either the root variable of a new tree description is identified with a leaf variable (corresponds to substitution),
- or the root variable and a leaf variable of the new description are identified with internal node variables (corresponds to adjunction)

⇒ the copy language can be generated.

The definition of a locality constraint leads to *local TDG*. Local TDGs are semilinear. Whether they are polynomially parsable has not yet been shown. But it might be the case because

- the derivation of the tree descriptions has an underlying context-free backbone
- the tree descriptions generated by the grammar could be transformed into *normal dominance constraints* (Koller et al. 2003). These are known to be polynomially solvable.

Local TDGs can generate the copy language and also the word orders of scrambling.