Grammar Formalisms
Feature Structures in LTAG

Laura Kallmeyer, Timm Lichte, Wolfgang Maier
Universität Tübingen

02.05.2007

Outline

Why feature structures? (1)

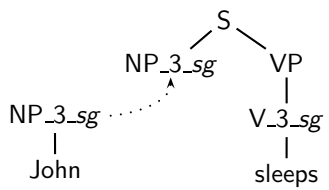
Idea: Instead of atomic categorial symbols, feature structures are used as non-terminal nodes.

Two reasons with respect to TAG:

- generalizing agreement
- modelling adjunction constraints

Why feature structures (2)

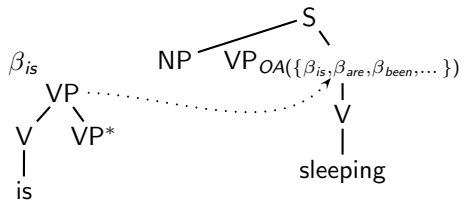
Example without feature structures:



⇒ the generalization that the finite verb and its subject agree in number and person is not captured.

Why feature structures (3)

Example without feature structures:



⇒ the generalization that some form of the auxiliary *to be* needs to be adjoined to *sleeping* is not captured.

Why feature structures (4)

Things get even worse when combining the two cases: We need different elementary trees for

- *sleeping* requiring a singular form of *to be* adjoining at the VP. In this case the subject NP must be NP_3_*sg*.
- *sleeping* requiring a plural form of *to be* adjoining at the VP. In this case the subject NP must be NP_3_*pl*.

Feature structures - Basics (1)

$$\begin{bmatrix} \text{ATTR}_1 & \text{val}_1 \\ \text{ATTR}_2 & \text{val}_2 \\ \ldots & \ldots \\ \text{ATTR}_n & \text{val}_n \end{bmatrix} \quad \begin{array}{l} \{<\text{ATTR}_1,\text{val}_1>,<\text{ATTR}_2,\text{val}_2>, \\ \quad \ldots,<\text{ATTR}_n,\text{val}_n>\} \end{array}$$

subsumption $\sqsubseteq$ **:** $\boxed{\begin{array}{l} A \sqsubseteq B, \text{ iff} \\ \text{if } t \in A, \text{ then } t \in B. \end{array}}$

unification $\sqcup$ **:** $\boxed{\begin{array}{l} A \sqcup B = C, \text{ iff} \\ C \text{ is the smallest feature structure such that} \\ A \sqsubseteq C \text{ and } B \sqsubseteq C. \end{array}}$

Note: We are using only untyped feature structures!

Feature structures - Basics (2)

Feature structures as values:

- non-recursive: $\begin{bmatrix} \text{AGR} & \begin{bmatrix} \text{PERS} & 1 \\ \text{NUMB} & \text{sg} \\ \text{CASE} & \text{nom} \end{bmatrix} \end{bmatrix}$
- recursive: $\begin{bmatrix} \text{SUBCAT} & \left\langle \begin{bmatrix} \text{SUBCAT} & [\ldots] \end{bmatrix} \right\rangle \end{bmatrix}$

FTAG uses non-recursive feature structures!

Feature structures - Basics (3)

Re-entrancies (or "links"):

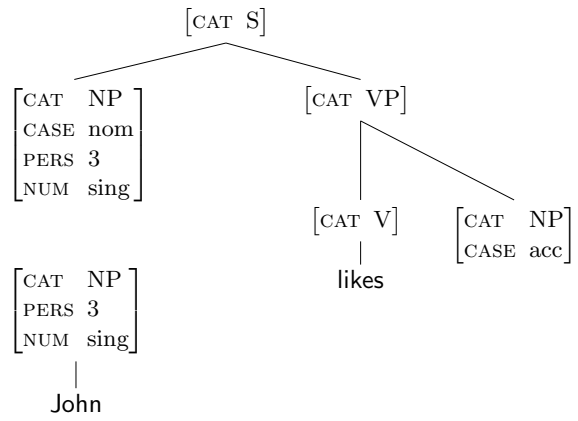
- boxed numbers ($\boxed{1}$, $\boxed{2}$, ...)
- within feature structures:

  $\begin{bmatrix} \text{ATTR}_1 & \boxed{1} \\ \text{ATTR}_2 & \boxed{1} \end{bmatrix}$ $\begin{bmatrix} \text{ATTR}_1 & \boxed{1}\text{val}_1 \\ \text{ATTR}_2 & \boxed{1} \end{bmatrix}$ $\begin{bmatrix} \text{ATTR}_1 & \boxed{1}\begin{bmatrix} \text{ATTR}_2 & \boxed{1} \end{bmatrix} \end{bmatrix}$

  FTAG uses acyclic re-entrancies!
- between feature structures (in a tree):

  ⇒ see FTAG ...

Feature structures - as tree nodes



Agreement properties can be represented by using underspecified feature structures. When attaching two trees, the feature structures of the participating nodes are unified.

FTAG (1)

Feature-structure based TAG (FTAG): Vijay-Shanker & Joshi (1988).

Modelling adjunction constraints requires to split the feature structure of nodes:

- **top features**: "what the node represents in the surrounding structure"
- **bottom features**: "what the tree below the node represents"

Substitution nodes have only a top feature structure. Nodes in the same elementary tree can share features (extended domain of locality).

In the final derived tree, top and bottom unify.

FTAG (2): Adjunction constraints

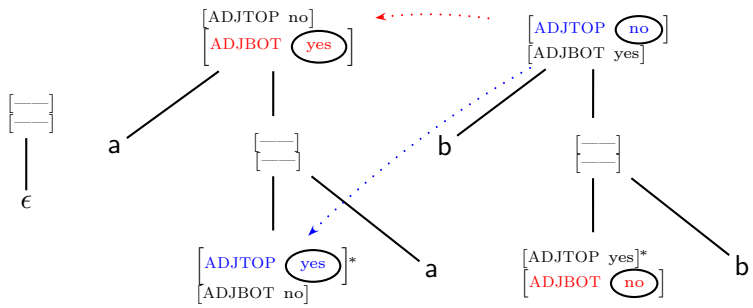
Adjunction constraints are encoded in the following way:

- - : $\begin{bmatrix} \text{CAT} & \text{VP} \end{bmatrix}$ $\begin{bmatrix} \text{CAT} & \text{VP} \end{bmatrix}$ unifiable
- OA + SA: $\begin{bmatrix} \text{CAT} & \text{VP} \\ \text{MOD} & \text{ind} \end{bmatrix}$ $\begin{bmatrix} \text{CAT} & \text{VP} \\ \text{MOD} & \text{ger} \end{bmatrix}$ feature mismatch
- NA: Can also be expressed in terms of features. No extra mechanisms is necessary

continued...

FTAG (2): Adjunction constraints (NA)

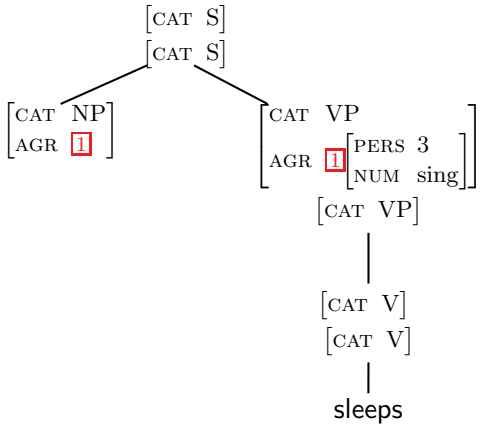
- NA: Features must be chosen in a way that at nodes with an NA constraint, no unification is possible (and therefore no adjunction). Example: FTAG for the copy language.



No adjunction into root nodes and foot nodes because of marked feature collisions.

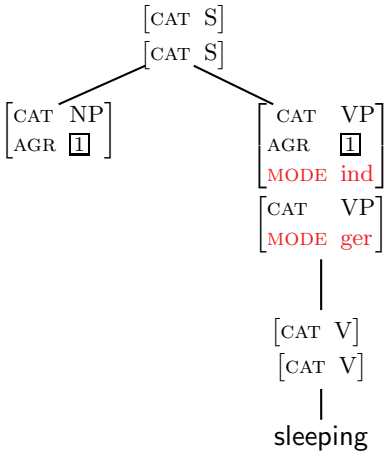
FTAG (3)

Example:



FTAG (4)

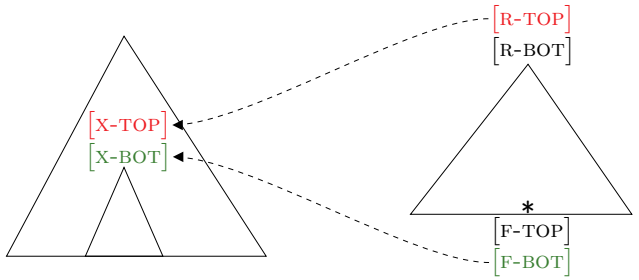
Example:



FTAG (4)

Unification during derivation:

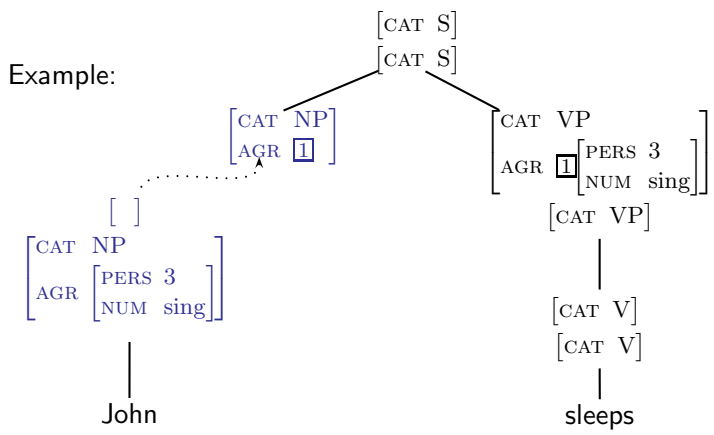
- **Substitution**: the top of the root of the new initial tree unifies with the top of the substitution node
- **Adjunction**: the top of the root of the new auxiliary tree unifies with the top of the adjunction site, and the bottom of the foot of the new tree unifies with the bottom of the adjunction site.



- In the final derived tree, top and bottom unify for all nodes.

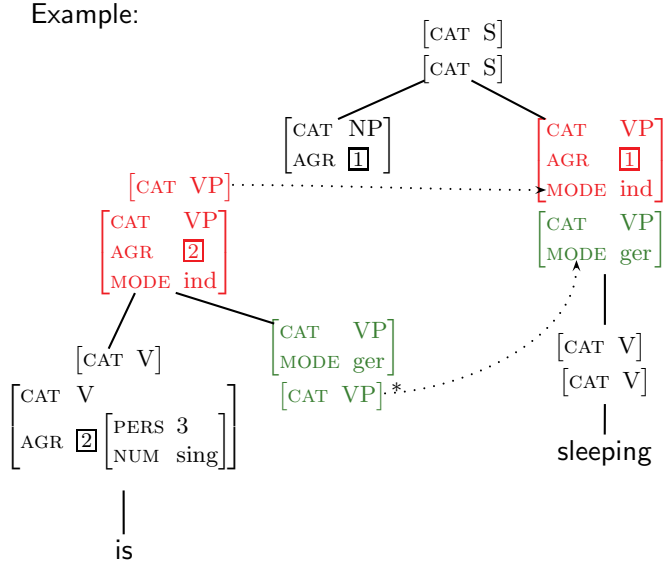
FTAG (5)

Example:



FTAG (6)

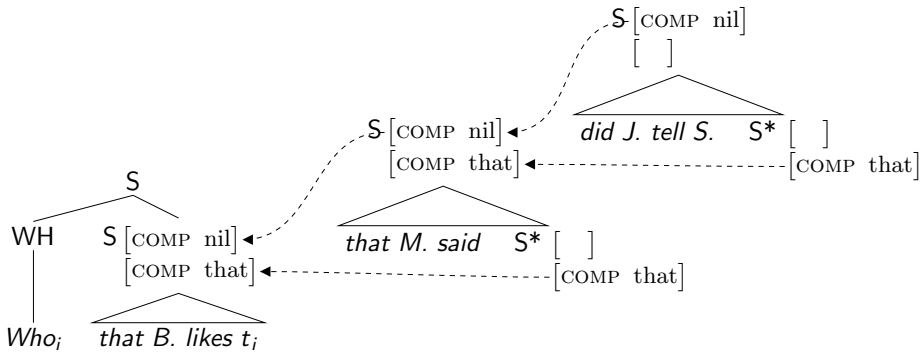
Example:



FTAG (7)

Example:

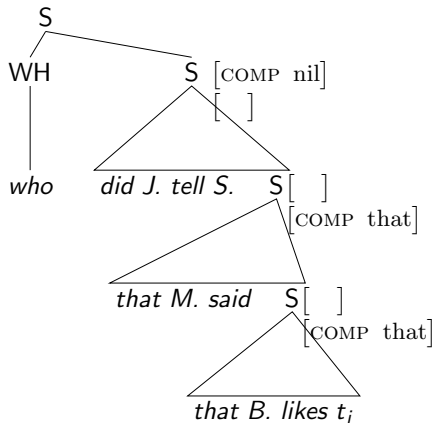
(1) who$_i$ did John tell Sam that Mary said that Bill likes t_i



FTAG (7)

Example:

(2) who$_i$ did John tell Sam that Mary said that Bill likes t_i



FTAG (8)

LTAG feature structures are restricted; there is **only a finite set of possible feature structures** (given finite sets of features and values, and non-recursivity).

Therefore, the following can be shown:

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.

Summary

- Feature structures as nodes allow to abstract away from agreement properties by underspecification. Linguistic generalizations can be expressed more conveniently.
- Adjunction constraints can be encoded into feature structures.
- The feature structures of FTAG do not add expressive power, hence FTAG and TAG are weakly equivalent.