Grammar Implementation with TAG

TAG with Feature Structures

Timm Lichte

HHU Düsseldorf

SS 2011

20.04.2011

Outline

1. Why feature structures?
2. Basics of feature structure logic
3. Feature Structure based TAG (FTAG)

Why feature structures?

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

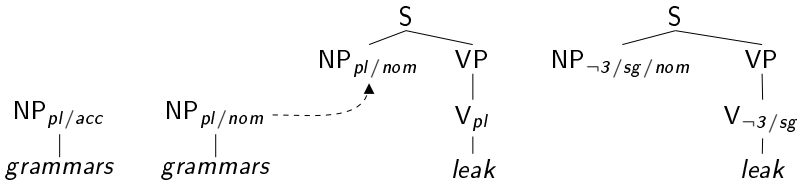
Two reasons with respect to TAG:

- generalizing agreement (via underspecification)
- modelling adjunction constraints

$\Rightarrow$ smaller grammars that are easier to maintain

Why feature structures? Agreement

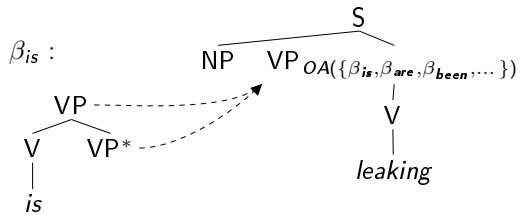
Example without feature structures:



$\Longrightarrow$ The generalization that the finite verb and its subject agree in number and person is not captured.
$\Longrightarrow$ Every morphological alternative gives rise to a new elementary tree!

Why feature structures? Adjunction constraints

Example without feature structures:

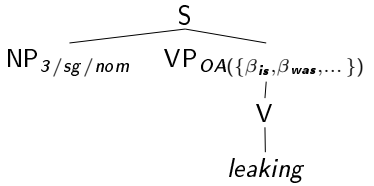


$\Longrightarrow$ The generalization that some form of the auxiliary *to be* needs to be adjoined to *leaking* is not captured.

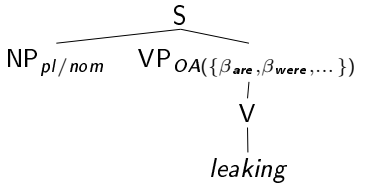
Why feature structures? Combining the two

Things get even worse when **combining agreement with adjunction constraints**:

- If *leaking* requires a singular auxiliary to adjoin at the VP node, then the subject must be $NP_{3/sg/nom}$.



- If *leaking* requires a plural auxiliary to adjoin at the VP node, then the subject must be $NP_{pl/nom}$.



Feature structures - Basics (1)

$$\begin{bmatrix} \text{attr}_1 & \text{val}_1 \\ \text{attr}_2 & \text{val}_2 \\ \dots & \dots \\ \text{attr}_n & \text{val}_n \end{bmatrix} \quad \{<\text{attr}_1,\text{val}_1>,<\text{attr}_2,\text{val}_2>, \dots,<\text{attr}_n,\text{val}_n>\}$$

subsumption $\sqsubseteq$:

$A \sqsubseteq B$, iff
if $t \in A$, then $t \in B$.

unification $\sqcup$:

$A \sqcup B = C$, iff
C is the smallest feature structure such that
$A \sqsubseteq C$ and $B \sqsubseteq C$.

Note: We are using only untyped feature structures!

Feature structures - Basics (2)

Feature structures as values:

- non-recursive: $\begin{bmatrix} \text{agr} & \begin{bmatrix} \text{num} & \text{sg} \\ \text{pers} & 1 \\ \text{3rdsing} & - \\ \text{gen} & \text{neuter} \end{bmatrix} \end{bmatrix}$
- recursive: $\begin{bmatrix} \text{subcat} & \left\langle \begin{bmatrix} \text{subcat} & [\dots] \end{bmatrix} \right\rangle \end{bmatrix}$

FTAG uses non-recursive feature structures!

Feature structures - Basics (3)

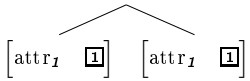
Re-entrancies (or "links"):

- boxed numbers (1, 2, ...)
- within feature structures:

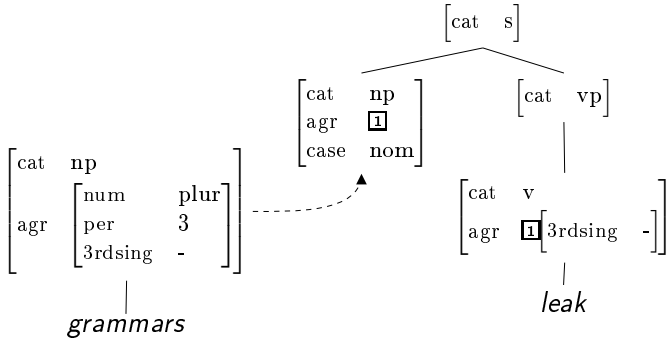
$$\begin{bmatrix} \text{attr}_1 & \boxed{1} \\ \text{attr}_2 & \boxed{1} \end{bmatrix} \qquad \begin{bmatrix} \text{attr}_1 & \boxed{1}\text{val}_1 \\ \text{attr}_2 & \boxed{1} \end{bmatrix} \qquad \begin{bmatrix} \text{attr}_1 & \boxed{1} \begin{bmatrix} \text{attr}_2 & \boxed{1} \end{bmatrix} \end{bmatrix}$$

FTAG uses acyclic re-entrancies!

- between feature structures (in a tree):



Feature structures - as tree nodes in a TSG



- Agreement properties can be undespecified.
- When combining two trees, the feature structures of the participating nodes are unified.
- TSG: substitution $\rightsquigarrow$ unification of leaf nodes and root nodes

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"

In the final derived tree, top and bottom unify.

FTAG (2): Adjunction constraints

Adjunction constraints are encoded in the following way:

- **SA:** top and bot are unifiable.

$$\begin{bmatrix} \text{cat} & \text{vp} \end{bmatrix} \\ \begin{bmatrix} \text{cat} & \text{vp} \end{bmatrix}$$

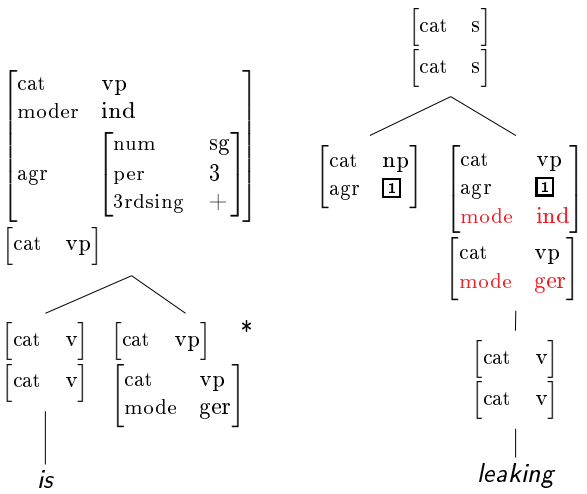
- **OA + SA:** feature mismatch between top and bot

$$\begin{bmatrix} \text{cat} & \text{vp} \\ \text{mode} & \text{ind} \end{bmatrix} \\ \begin{bmatrix} \text{cat} & \text{vp} \\ \text{mode} & \text{ger} \end{bmatrix}$$

- **NA:** top and bot are unifiable, but there is no auxiliary tree in the grammar that can be unified with top and bot.

FTAG (3): Agreement and adjunction constraints

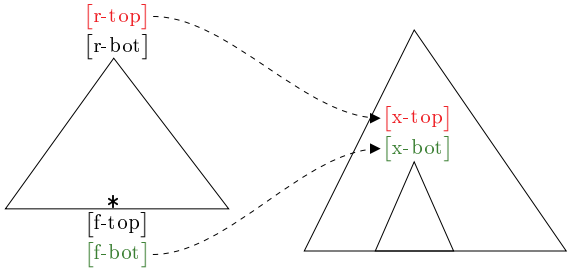
Example for top-bottom feature structures:



FTAG (4): Unification with top-bottom feature structures

Unification during derivation:

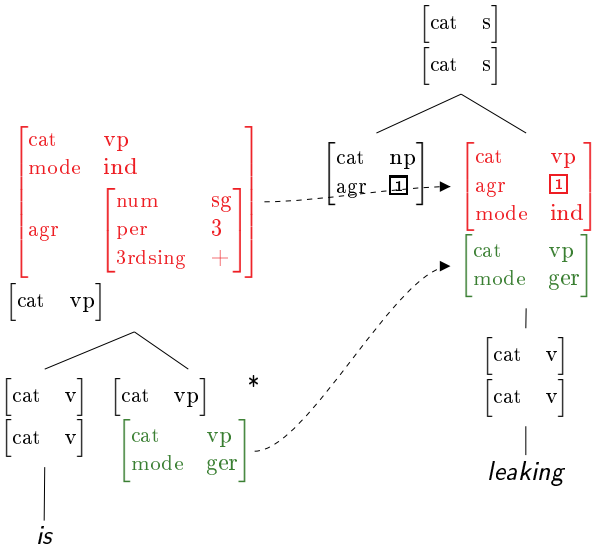
- **Substitution**: the top of the root of the rewriting tree unifies with the top of the substitution node
- **Adjunction**: the top of the root of the rewriting tree unifies with the top of the adjunction site, and the bottom of the foot of the rewriting tree unifies with the bottom of the adjunction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



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

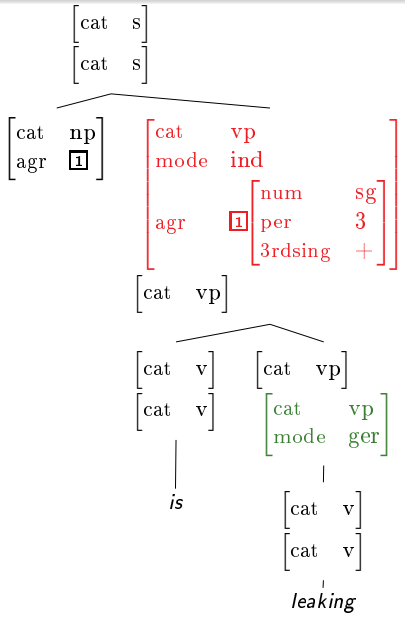
FTAG (3)

Example:



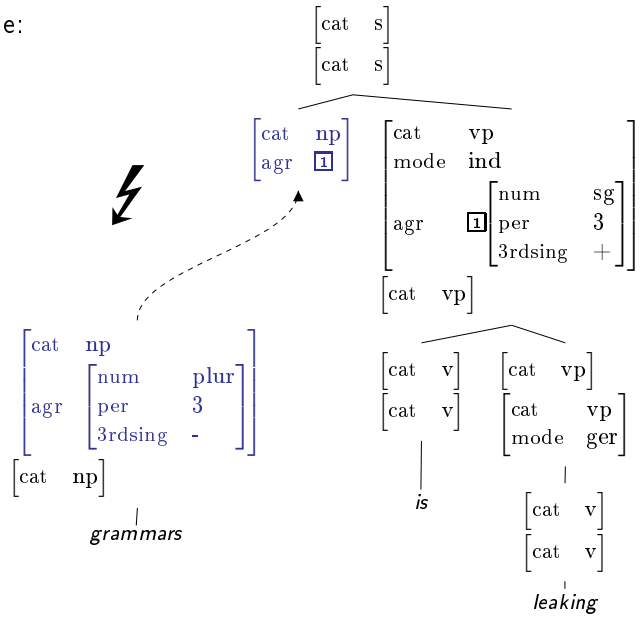
FTAG (4)

Example:



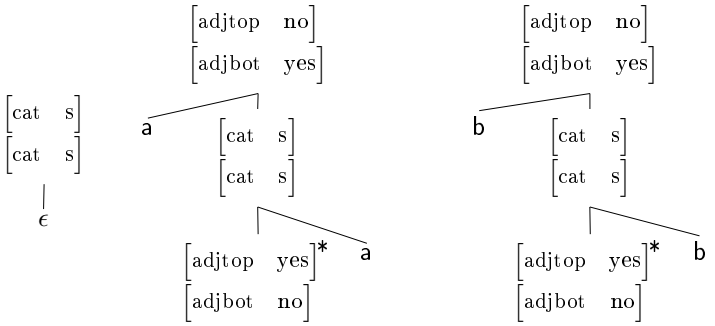
FTAG (5)

Example:



FTAG (6): Adjunction constraints (NA)

- Features must be chosen in a way that no unification with feature structures of auxiliary trees is possible (and therefore no adjunction).
- Example: FTAG for the copy language.



FTAG (7)

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.