Einführung in die Computerlinguistik

Feature Structures – Merkmalsstrukturen

Laura Kallmeyer

Heinrich-Heine-Universität Düsseldorf

Summer 2018

Introduction (1)

Non-terminals that are used in CFGs are usually not enough to express linguistic generalisations

Exmample: Agreement

Missed generalisation:

$S \rightarrow$ NP-Sg VP-Sg $\quad S \rightarrow$ NP-Pl VP-Pl

Better: $S \rightarrow$ NP VP $\quad$ Condition: NP and VP agree in their number

Introduction (2)

To express such generalisations, we can factorise the non-terminals:

- A non-terminal is no longer atomic, but it has a structure.
- The content of the non-terminals is described via attributes (i.e., features) that can have certain values.
- Such structures are called attribute-value structures or feature structures. They are often represented in an attribute-value matrix (AVM).

Feature structures

$$\begin{bmatrix} \text{cat} & \text{NP} \\ \text{num} & \text{Pl} \end{bmatrix} \qquad \begin{bmatrix} \text{lex} & \text{ihm} \\ \text{cat} & \text{Pro} \\ \text{case} & \text{dat} \\ \text{num} & \text{Sg} \\ \text{gen} & \text{m} \end{bmatrix} \qquad \begin{bmatrix} \text{pred} & \text{give} \\ \text{donor} & \text{Adam} \\ \text{theme} & \text{apple} \\ \text{recipient} & \text{Eve} \end{bmatrix}$$

Introduction (2)

- It is possible to refer to the same attribute value in different places (structure sharing)

Structure sharing

$$\begin{bmatrix} \text{cat} & \text{S} \end{bmatrix} \rightarrow \begin{bmatrix} \text{cat} & \text{NP} \\ \text{num} & \boxed{1} \end{bmatrix} \begin{bmatrix} \text{cat} & \text{VP} \\ \text{num} & \boxed{1} \end{bmatrix}$$

(The variable $\boxed{1}$ always denotes the same value.)

$$\begin{bmatrix} \text{pred} & \text{give} \\ \text{donor} & \boxed{1}\text{Adam} \\ \text{agent} & \boxed{1} \\ \text{theme} & \text{apple} \\ \text{recipient} & \text{Eve} \end{bmatrix}$$

Introduction (4)

- Underspecification: Not all the values are always known. Instead of listing all the possibilities it is possible to specify only those values that are known.

Underspecification of attributes

$$\begin{bmatrix} \text{cat} & \text{N} \\ \text{num} & \text{Sg} \\ \text{gen} & \text{m} \end{bmatrix} \rightarrow \text{man} \qquad \begin{bmatrix} \text{cat} & \text{N} \\ \text{gen} & \text{n} \end{bmatrix} \rightarrow \text{fish}$$

$$\begin{bmatrix} \text{cat} & \text{Det} \\ \text{num} & \text{Sg} \end{bmatrix} \rightarrow \text{a} \qquad \begin{bmatrix} \text{cat} & \text{Det} \end{bmatrix} \rightarrow \text{the}$$

$$\begin{bmatrix} \text{cat} & \text{NP} \\ \text{num} & \boxed{1} \\ \text{gen} & \boxed{2} \\ \text{pers} & 3 \end{bmatrix} \rightarrow \begin{bmatrix} \text{cat} & \text{Det} \\ \text{num} & \boxed{1} \end{bmatrix} \begin{bmatrix} \text{cat} & \text{N} \\ \text{num} & \boxed{1} \\ \text{gen} & \boxed{2} \end{bmatrix}$$

Introduction (5)

- Attributes do not necessarily have atomic values. The value of an attribute can be another attribute-value structure.

Recursive feature structures

$$\begin{bmatrix} \text{cat} & \text{N} \\ \text{agr} & \begin{bmatrix} \text{gen} & \text{n} \end{bmatrix} \end{bmatrix} \rightarrow \text{fish} \qquad \begin{bmatrix} \text{cat} & \text{Det} \\ \text{agr} & \begin{bmatrix} \text{num} & \text{Sg} \end{bmatrix} \end{bmatrix} \rightarrow \text{a}$$

$$\begin{bmatrix} \text{cat} & \text{NP} \\ \text{agr} & \boxed{1}\begin{bmatrix} \text{pers} & 3 \end{bmatrix} \end{bmatrix} \rightarrow \begin{bmatrix} \text{cat} & \text{Det} \\ \text{agr} & \boxed{1} \end{bmatrix} \begin{bmatrix} \text{cat} & \text{N} \\ \text{agr} & \boxed{1} \end{bmatrix}$$

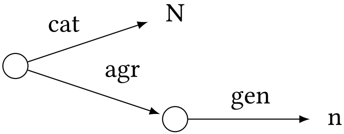
Attribute-value structures as graphs (1)

Attribute-value structures are usually formalised as directed graphs.

Two possibilities: an attribute-value matrix such as

$$\begin{bmatrix} \text{cat} & \text{N} \\ \text{agr} & \begin{bmatrix} \text{gen} & \text{n} \end{bmatrix} \end{bmatrix}$$

1. can be represented as a directed graph



2. or as a description of such a graph, that can be in principle satisfied by an infinite number of graphs.

Attribute-value structures as graphs (2)

In the following, we assume feature structures to be graphs (and not expressions in a feature logic).

Feature structure

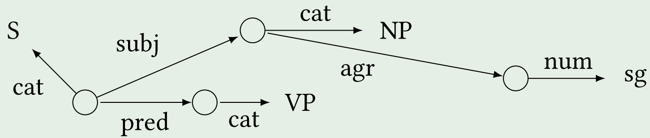
A (untyped) feature structure can be defined as a tuple $\langle V, A, Val, r\rangle$ such that

- V is a set of vertices (= nodes).
- A is a finite set of partial functions $a : V \rightarrow V$
- Val is a finite set of atomic values and there is a partial function $l_{Val} : \{v \in V \mid \text{there is no } a \in A \text{ such that } a(v) \text{ is defined, i.e., there is no outgoing edge for } v\} \rightarrow Val$
- $r \in V$ is the unique root of the feature structure, i.e., there is exactly one node in V (which is r) such that r does not have an incoming edge or, to put it differently, there is no $v \in V, a \in A$ with $a(v) = r$.

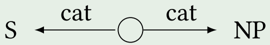
Attribute-value structures as graphs (3)

Feature structures as graphs

- possible feature structure:

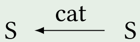


- ill-formed feature structures:



(attributes have to be functional)

(there must be a unique root node)



(only leaves are labeled with atomic values)

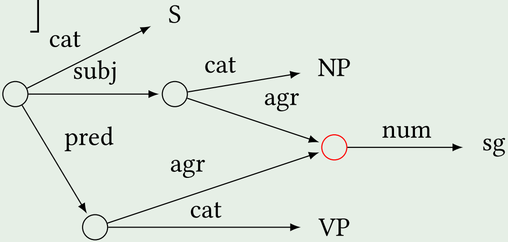
Attribute-value structures as graphs (4)

Attribute-value graphs are not always trees since we can have more than one incoming edge per node.

Structure Sharing

$$\begin{bmatrix} \text{cat} & \text{S} \\ \text{subj} & \begin{bmatrix} \text{cat} & \text{NP} \\ \text{agr} & \boxed{1}\begin{bmatrix} \text{num} & \text{sg} \end{bmatrix} \end{bmatrix} \\ \text{pred} & \begin{bmatrix} \text{cat} & \text{VP} \\ \text{agr} & \boxed{1} \end{bmatrix} \end{bmatrix}$$

structure sharing



Subsumption and unification (1)

Subsumption: Relation on feature structures: S_1 subsumes S_2 ($S_1 \sqsubseteq S_2$), if S_2 contains (at least) all the information from S_1.

Example

Subsumption

Ex. S_1: $\begin{bmatrix} \text{cat} & \text{V} \\ \text{agr} & \begin{bmatrix} \text{num} & \text{Sg} \end{bmatrix} \end{bmatrix}$ $\qquad$ S_2: $\begin{bmatrix} \text{orth} & \text{laughs} \\ \text{cat} & \text{V} \\ \text{agr} & \begin{bmatrix} \text{pers} & 3 \\ \text{num} & \text{Sg} \end{bmatrix} \end{bmatrix}$

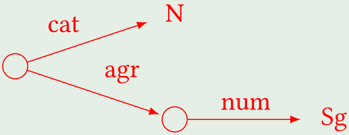
$S_1 \sqsubseteq S_2$

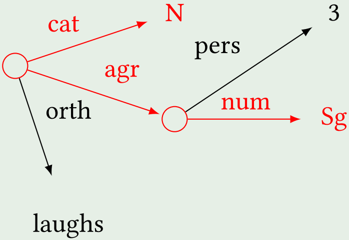
In other words: there is a homomorphism from the nodes of S_1 to the nodes of S_2 that preserves edges and labels and that maps the root of S_1 to the root of S_2.

Subsumption and unification (2)

Example

Subsumption S_1 as a graph and its image under the homomorphism in S_2:





Subsumption and unification (3)

Subsumption

Let $S_1 = \langle V_1, A, Val, r_1 \rangle$ and $S_2 = \langle V_2, A, Val, r_2 \rangle$ be feature structures.

S_1 subsumes S_2, $S_1 \sqsubseteq S_2$ if there is a function $h : V_1 \rightarrow V_2$ such that

- $h(r_1) = r_2$,
- for all $v_1, v_2 \in V_1$ and all $a \in A$: if $a(v_1) = v_2$, then $a(h(v_1)) = h(v_2)$, and
- for all $v \in V_1$ and all $l \in Val$: if $l_{Val}(v) = l$, then $l_{Val}(h(v)) = l$.

Subsumption and unification (3)

Subsumption: more examples

- S_1: $\begin{bmatrix} \text{cat} & \text{N} \\ \text{agr} & \begin{bmatrix} \text{num} & \text{Sg} \\ \text{case} & \text{acc} \end{bmatrix} \end{bmatrix}$ S_2: $\begin{bmatrix} \text{orth} & \text{laughs} \\ \text{agr} & \begin{bmatrix} \text{pers} & 3 \\ \text{num} & \text{Sg} \end{bmatrix} \end{bmatrix}$

  $S_1 \not\sqsubseteq S_2$, $S_2 \not\sqsubseteq S_1$

- S_1: $\begin{bmatrix} \text{cat} & \text{N} \\ \text{agr} & \boxed{1} \end{bmatrix}$ S_2: $\begin{bmatrix} \text{cat} & \text{N} \\ \text{agr} & \begin{bmatrix} \text{pers} & 3 \\ \text{num} & \text{Sg} \end{bmatrix} \end{bmatrix}$

  $S_1 \sqsubseteq S_2$

Subsumption and unification (4)

Subsumption is a partial order, so it is

1. reflexive: each structure subsumes itself $S \sqsubseteq S$ for all S;
2. transitive: if $S_1 \sqsubseteq S_2$ and $S_2 \sqsubseteq S_3$ then $S_1 \sqsubseteq S_3$ for all S_1, S_2, S_3;
3. asymmetric: if $S_1 \sqsubseteq S_2$ and $S_2 \sqsubseteq S_1$ then $S_1 = S_2$.

An empty feature structure [ ] subsumes all other feature structures.

Subsumption and unification (5)

A feature structure S is a unification of S_1 and S_2 ($S_1 \sqcup S_2$), if S is subsumed by both S_1 and S_2 and S subsumes all other feature structures, that are subsumed by both S_1 and S_2.

$$\begin{bmatrix} \text{cat} & \text{V} \\ \text{agr} & \begin{bmatrix} \text{num} & \text{Sg} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{cat} & \text{V} \\ \text{agr} & \begin{bmatrix} \text{pers} & 3 \end{bmatrix} \end{bmatrix} = \begin{bmatrix} \text{cat} & \text{V} \\ \text{agr} & \begin{bmatrix} \text{num} & \text{Sg} \\ \text{pers} & 3 \end{bmatrix} \end{bmatrix}$$

To make $\sqcup$ always defined, we introduce a symbol $\perp$ that refers to an inconsistent feature structure that is subsumed by all feature structures.

$$\begin{bmatrix} \text{cat} & \text{NP} \\ \text{agr} & \begin{bmatrix} \text{num} & \text{Sg} \end{bmatrix} \end{bmatrix} \sqcup \begin{bmatrix} \text{cat} & \text{V} \\ \text{agr} & \begin{bmatrix} \text{num} & \text{Sg} \\ \text{pers} & 3 \end{bmatrix} \end{bmatrix} = \perp$$

Subsumption and unification (6)

Feature structures that are related by the $\sqsubseteq$ relation, form a lattice: $\sqsubseteq$ is a partial order and for any S_1, S_2 the following holds:

(sup) There is a feature structure S, such that $S_1 \sqsubseteq S$ and $S_2 \sqsubseteq S$ and S also subsumes all other feature structures that are subsumed by both S_1 and S_2. S is called Supremum of $\{S_1, S_2\}$.

(inf) There is a feature structure S, such that $S \sqsubseteq S_1$ and $S \sqsubseteq S_2$ and S is subsumed by all other structures that subsume both S_1 and S_2. S is called Infimum of $\{S_1, S_2\}$.

From this it follows that with respect to the $\sqsubseteq$ the smallest element is $[\,]$, and the biggest element is $\perp$.

Typed feature structures (1)

The examples mentioned above implicitly imply that CAT is a syntactic category and AGR is responsible for the agreement. I.e., the following feature structures should not be possible:

$$\begin{bmatrix} \text{cat} & \text{Sg} \\ \text{agr} & \begin{bmatrix} \text{num} & 3 \\ \text{pers} & \text{V} \end{bmatrix} \end{bmatrix} \qquad \begin{bmatrix} \text{cat} & \begin{bmatrix} \text{agr} & \begin{bmatrix} \text{pers} & 3 \end{bmatrix} \end{bmatrix} \end{bmatrix}$$

However, nothing prevents the existence of such structures so far, as there is no generalisation defined for this case.

Goal: formulate restrictions of the kind “an agreement feature structure can hve only attributes NUM, PERS and GEN”.

Typed feature structures (2)

So we introduce types for feature structures:

- Each feature structure has a type τ.
- For each type τ it is defined which attributes it has and what are the types of the values of these attributes.
- Types are organised in a type hierarchy, where specific types are ordered under the general types.
- Unification operation is extended in order to take care of the types.

Typed feature structures (3)

Types and their possible arguments are identified using the type hierarchy and attributes for single types.

$$\begin{bmatrix} \textit{agr-structure} & \\ \text{agr} & \textit{agr} \end{bmatrix} \begin{bmatrix} \textit{agr} & \\ \text{num} & \textit{num} \\ \text{gen} & \textit{gen} \\ \text{pers} & \textit{pers} \end{bmatrix}$$

$$\begin{bmatrix} \textit{determiner} & \\ \text{quant} & \textit{quant} \end{bmatrix} \begin{bmatrix} \textit{noun} & \\ \text{case} & \textit{case} \end{bmatrix} \begin{bmatrix} \textit{syncat} & \\ \text{cat} & \textit{cat} \end{bmatrix}$$

Type *quant*: {every, most, some, none}, Type *num*: {Sg, Pl}, Type *gen*: {m,f,n}, Type *pers*: {1, 2, 3}, Type *case*: {nom, acc, dat}, Type *cat*: {N, V, NP, VP, S, …}

Typed feature structures (4)

Types

$$\begin{bmatrix} \textit{agr-structure} & \\ \text{agr} & \textit{agr} \end{bmatrix} \quad \begin{bmatrix} \textit{syncat} & \\ \text{cat} & \textit{cat} \end{bmatrix} \quad \begin{bmatrix} \textit{agr} & \\ \text{num} & \textit{num} \\ \text{gen} & \textit{gen} \\ \text{pers} & \textit{pers} \end{bmatrix}$$

$$\begin{bmatrix} \textit{determiner} & \\ \text{quant} & \textit{quant} \end{bmatrix} \quad \begin{bmatrix} \textit{noun} & \\ \text{case} & \textit{case} \end{bmatrix}$$

Type hierarchy:

$$\begin{tikzcd} \textit{agr-structure} \arrow[d, no head] \arrow[dr, no head] & \textit{syncat} \arrow[dl, no head] \arrow[d, no head] \\ \textit{det} & \textit{noun} \end{tikzcd}$$

The attributes of *noun* are determined by the types *agr-structure*, *syncat* and *noun*.

$$\begin{bmatrix} \textit{noun} & \\ \text{cat} & \text{N} \\ \text{case} & \text{acc} \\ \text{agr} & \begin{bmatrix} \textit{agr} & \\ \text{num} & \text{Sg} \\ \text{gen} & \text{m} \\ \text{pers} & 3 \end{bmatrix} \end{bmatrix}$$

Extensions (1)

Some linguistic theories use also sets or lists as attribute values. Example.: Head-Driven Phrase Structure Grammar (HPSG) codes syntactic trees as feature structures, where all the daughters of the node are provided as a value of the respective attribute in form of a list.

set attributes

$$\begin{bmatrix} \textit{phrase} & \\ \text{dtrs} & \left\langle \begin{bmatrix} \text{cat} & \text{PRO} \\ \text{orth} & \text{I} \end{bmatrix}, \begin{bmatrix} \text{cat} & \text{VP} \\ \text{dtrs} & \left\langle \begin{bmatrix} \text{cat} & \text{V} \\ \text{orth} & \text{love} \end{bmatrix}, \begin{bmatrix} \text{cat} & \text{NP} \\ \text{orth} & \text{New York} \end{bmatrix} \right\rangle \end{bmatrix} \right\rangle \end{bmatrix}$$

Extensions (2)

- Some systems work directly with feature structures as graphs.
- Some use descriptions of features structures.

Advantage of descriptions: variable expressive power depending on the used Logic (of course in connection with the complexity). Some useful operations:

1. Disjunction: $\textsc{case} = \text{acc} \vee \textsc{case} = \text{dat}$
2. Negation: $\neg(\textsc{case} = \text{nom})$
3. Non-equality of paths: $\textsc{subj}\ [\textsc{case}] \neq \textsc{obj}\ [\textsc{case}]$