**Introduction to Lexical Functional Grammar**

**Session 3**

**Introduction to LFG (c-structure, a-structure, f-structure & introduction to graph theory)**

---

**Intro to LFG**

- This session is a first introduction to LFG:
  - Introduce LFG on an informal basis
  - Go through each part of the name
  - Spell out the concepts behind it
  - Explain graph theory and terminology where needed

---

**Lexical**

- Obey the Lexical Integrity Principle
- Non-transformational/ non-derivational
- Psychologically/computationally plausible
- Treats syntactic phenomena locally
- Monotonic
- Constraint-based
- One level of constituent structure: c-structure

**Functional**

- Autonomous representation of grammatical functions
- Grammatical functions are feature-like
- f-structure
- Unification-based
- Typologically plausible
- Parallel architecture

**Grammar**

- “Grammar” = “generative grammar”
- More realistic approach to universals
- Higher psychological and computational plausibility makes it a better approach
- Grammars are formalized, and thus testable

---

**LFG**

- **Joan Bresnan** (former student of Chomsky’s; syntactician concerned about psycholinguistic issues)
- **Ronald M. Kaplan** (computational linguist)


---

**Lexical**

- A lexical (or lexicalist) theory is one in which words and the lexicon play a major role

**Lexical Integrity Principle**

Words are the „atoms“ out of which syntactic structure is built. Syntactic rules cannot create words or refer to the internal structure of words, and each terminal node (or „leaf“ of the tree) is a word. Words are built in a different module: the lexicon.
Lexical

- Only one level of constituent structure: c-structure

\[
\begin{align*}
S & \quad \text{NP} \quad \text{VP} \\
\text{NP} & \quad \text{D} \quad \text{N} \\
\text{VP} & \quad \text{V} \quad \text{NP}
\end{align*}
\]
- \text{the dinosaur} ate \text{the tree}

Functional

- Representation of grammatical functions, i.e. notions like "subject" and "object", parallel to c-structure
- Function also in the mathematical sense
- Grammatical functions are not represented in a tree-structure but as features in the f-structure (functional structure)
- C-structure and f-structure are completely different structures

Feature structures

- Excursus:
  - An introduction to graph theory

\[
\begin{align*}
\text{S} & \quad \text{NP} \quad \text{VP} \\
\text{NP} & \quad \text{D} \quad \text{N} \\
\text{VP} & \quad \text{V} \quad \text{NP}
\end{align*}
\]
- Linguistic information is modelled by means of feature structures
- Feature structures are used on all linguistic description levels
- A feature structure is a set of feature-value pairs (also called attribute-value pairs)
- Values can be atomic or again a feature structure
- Important is the idea of underspecification: not all features must be explicitly stated

Feature structures

- Binary features: values can only be "+" or "−"

\[
\begin{align*}
\text{Matrix notation} & \quad \text{VOICED} + \\
\text{(AVM: attribute value matrix)} & \quad \text{BILABIAL} + \\
\text{Graph notation} & \quad \text{VOICED} + \\
\text{(DAG: directed acyclic graph)} & \quad \text{BILABIAL} +
\end{align*}
\]
Feature structures

- **Simple features**: have a set of possible values

  **AVM notation**
  
  \[
  \begin{array}{c}
  \text{CAT} \\
  \text{N} \\
  \text{CASE} \\
  \text{NOM} \\
  \text{GEN} \\
  \text{FEM}
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{AVM notation}
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{CAT} \\
  \text{N} \\
  \text{CASE} \\
  \text{NOM} \\
  \text{GEN} \\
  \text{FEM}
  \end{array}
  \]

- **Complex features**: have feature structures as their values

  **AVM notation**
  
  \[
  \begin{array}{c}
  \text{CAT} \\
  \text{V} \\
  \text{FIN} \\
  \text{+} \\
  \text{NUM} \\
  \text{SG} \\
  \text{PER} \\
  \text{3}
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{Graph notation}
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{CAT} \\
  \text{V} \\
  \text{FIN} \\
  \text{+} \\
  \text{NUM} \\
  \text{SG} \\
  \text{PER} \\
  \text{3}
  \end{array}
  \]

Feature structures

- **Structure sharing (or coreference or path equivalence)**: to express identity of values of different attributes

  \[
  \begin{array}{c}
  \text{AVM notation}
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{CAT} \\
  \text{V} \\
  \text{FIN} \\
  \text{+} \\
  \text{NUM} \\
  \text{SG} \\
  \text{PER} \\
  \text{3}
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{Graph notation}
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{FIN} \\
  \text{CAT} \\
  \text{V} \\
  \text{NUM} \\
  \text{SG} \\
  \text{PER} \\
  \text{3}
  \end{array}
  \]

Feature structures

- **Structure sharing (or coreference or path equivalence)** contd.

  \[
  \begin{array}{c}
  \text{AVM notation}
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{CAT} \\
  \text{V} \\
  \text{FIN} \\
  \text{+} \\
  \text{NUM} \\
  \text{SG} \\
  \text{PER} \\
  \text{3}
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{Graph notation}
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{FIN} \\
  \text{CAT} \\
  \text{V} \\
  \text{NUM} \\
  \text{SG} \\
  \text{PER} \\
  \text{3}
  \end{array}
  \]

Feature structures

- **Structure sharing (or coreference or path equivalence)** contd.

  \[
  \begin{array}{c}
  \text{AVM notation}
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{CAT} \\
  \text{V} \\
  \text{FIN} \\
  \text{+} \\
  \text{NUM} \\
  \text{SG} \\
  \text{PER} \\
  \text{3}
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{Graph notation}
  \end{array}
  \]

  \[
  \begin{array}{c}
  \text{FIN} \\
  \text{CAT} \\
  \text{V} \\
  \text{NUM} \\
  \text{SG} \\
  \text{PER} \\
  \text{3}
  \end{array}
  \]
Feature structures

- 3 notations

[Diagram of graph with nodes A, B, C, D, E and arrows]

\[
\begin{align*}
\text{AVM} & \rightarrow \begin{bmatrix} A & B & 1 \\ C & 2 & \{ E \} \end{bmatrix} \\
\langle A, B \rangle & = a \\
\langle A, C, E \rangle & = \langle A, B \rangle \\
\langle D \rangle & = \langle A, C \rangle
\end{align*}
\]

- Unification is the basic operation on feature structures
- It combines the information of two feature structures
- Unification is represented as the binary operator \( \sqcup \)

Feature structures

- Unification is monotonic, i.e., the unified feature structure still satisfies the original feature structure (no values are overwritten)
- Unification corresponds to the union operation in set theory, but may fail in case of incompatible information, i.e., feature structures have to be consistent even when they are the result of a unification.

Feature structures

- Formally, the unification of two feature structures \( F \) and \( G \) is defined as the most general feature structure \( H \), such that \( F \sqcup H \) and \( G \sqcup H \)
- This is notated as \( H = F \sqcup G \)

Feature structures

Examples of unification

- Equality
  \[ \text{NUMBER \ SG} \sqcup \{ \text{NUMBER \ SG} \} = \text{NUMBER \ SG} \]
- Incompatible values
  \[ \text{NUMBER \ SG} \sqcup \{ \text{NUMBER \ PL} \} = \bot \quad \text{\( \text{Fail!} \)} \]

- \( \{ \} \) value compatible with any value

- Adding information
  \[ \text{NUMBER \ SG} \sqcup \{ \text{PERSON \ 3} \} = \text{NUMBER \ SG} \quad \text{PERSON \ 3} \]
Feature structures

- Unification of features with similar values

$$\begin{align*}
\text{AGREEMENT} & \rightarrow \text{NUMBER SG} \\
\text{SUBJECT} & \rightarrow \text{AGREEMENT [PERSON 3]} \\
\text{AGREEMENT} & \rightarrow \text{NUMBER SG} \\
\text{SUBJECT} & \rightarrow \text{AGREEMENT [NUMBER SG \ PERSON 3]}
\end{align*}$$

- Unification of features with identical values

$$\begin{align*}
\text{AGREEMENT} & \rightarrow \text{NUMBER SG} \\
\text{SUBJECT} & \rightarrow \text{AGREEMENT [PERSON 3]} \\
\text{AGREEMENT} & \rightarrow \text{NUMBER SG} \\
\text{PERSON} & \rightarrow \text{PERSON 3} \\
\text{SUBJECT} & \rightarrow \text{AGREEMENT [PERSON 3]}
\end{align*}$$

- Further examples (instantiation)

$$\begin{align*}
\text{AGREEMENT} & \rightarrow \text{NUMBER SG} \\
\text{SUBJECT} & \rightarrow \text{AGREEMENT [PERSON 3]} \\
\text{AGREEMENT} & \rightarrow \text{NUMBER SG} \\
\text{PERSON} & \rightarrow \text{PERSON 3} \\
\text{SUBJECT} & \rightarrow \text{AGREEMENT [PERSON 3]}
\end{align*}$$

- Example of failure to unify

$$\begin{align*}
\text{AGREEMENT} & \rightarrow \text{NUMBER SG} \\
\text{PERSON} & \rightarrow \text{PERSON 3} \\
\text{SUBJECT} & \rightarrow \text{AGREEMENT [PERSON 3]} \\
\text{AGREEMENT} & \rightarrow \text{NUMBER SG} \\
\text{PERSON} & \rightarrow \text{PERSON 3} \\
\text{SUBJECT} & \rightarrow \text{AGREEMENT [PERSON 3]}
\end{align*}$$

- Semantics of unification

$$\begin{align*}
\text{CAT} & \rightarrow \text{V} \\
\text{AGR} & \rightarrow \text{PER 3} \\
\text{NUM} & \rightarrow \text{SG}
\end{align*}$$

Examples: $[\text{CAT N \ CASE NOM}] = [\text{CAT N \ CASE NOM}]$

- Example of failure to unify

$$\begin{align*}
\text{CAT} & \rightarrow \text{V} \\
\text{AGR} & \rightarrow \text{PER 1} \\
\text{NUM} & \rightarrow \text{SG}
\end{align*}$$
Feature structures

- **Subsumption** is an *ordering relation* between feature structures: a less specific feature structure *subsumes* an equally or more specific one, e.g.

\[
\begin{bmatrix}
\text{CAT NP} \\
\text{AGR} \\
\text{NUM 3}
\end{bmatrix}
\text{subsumes}
\begin{bmatrix}
\text{CAT} \\
\text{NP} \\
\text{AGR} \\
\text{PER} \\
\text{SG}
\end{bmatrix}
\]

- Subsumption corresponds to the subset relation in set theory
- The subsumption relation is represented by the binary operator \(\sqsubseteq\).

Feature structures

- **Subsumption** is a *partial ordering relation* between feature structures (i.e. there are pairs of feature structures that neither subsume nor are subsumed by each other)
- There are two cases in which the ordering relation does not hold:
  - if feature structures contain different but compatible information
  - if they contain conflicting information

Feature structures

- The ordering relation between feature structures can be defined as a *semilattice* (with the most general feature structure \([\ ]\) at the top)

Feature structures

- Semantics of subsumption

\[
\begin{array}{c}
\begin{bmatrix}
\text{CAT} \\
\text{N}
\end{bmatrix}
\end{array} \sqsubseteq
\begin{array}{c}
\begin{bmatrix}
\text{CAT} \\
\text{N} \\
\text{AGR}
\end{bmatrix}
\end{array}
\]

examples: \([\text{CAT } n]\) \(\sqsubseteq\) \([\text{CAT } n \text{ CASE nom}]\)

NPs in nominative case

NP

Feature structures

- **Subsumption** is an *ordering relation* between feature structures: a less specific feature structure *subsumes* an equally or more specific one, e.g.

\[
\begin{bmatrix}
\text{CAT NP} \\
\text{AGR} \\
\text{NUM 3}
\end{bmatrix}
\text{subsumes}
\begin{bmatrix}
\text{CAT} \\
\text{NP} \\
\text{AGR} \\
\text{PER} \\
\text{SG}
\end{bmatrix}
\]

- Subsumption corresponds to the subset relation in set theory
- The subsumption relation is represented by the binary operator \(\sqsubseteq\).

Feature structures

- **Subsumption** is a *partial ordering relation* between feature structures (i.e. there are pairs of feature structures that neither subsume nor are subsumed by each other)
- There are two cases in which the ordering relation does not hold:
  - if feature structures contain different but compatible information
  - if they contain conflicting information

Feature structures

- The ordering relation between feature structures can be defined as a *semilattice* (with the most general feature structure \([\ ]\) at the top)

Feature structures

- Semantics of subsumption

\[
\begin{array}{c}
\begin{bmatrix}
\text{CAT} \\
\text{N}
\end{bmatrix}
\end{array} \sqsubseteq
\begin{array}{c}
\begin{bmatrix}
\text{CAT} \\
\text{N} \\
\text{AGR}
\end{bmatrix}
\end{array}
\]

examples: \([\text{CAT } n]\) \(\sqsubseteq\) \([\text{CAT } n \text{ CASE nom}]\)

NPs in nominative case

NP

Feature structures

- **Subsumption** is an *ordering relation* between feature structures: a less specific feature structure *subsumes* an equally or more specific one, e.g.

\[
\begin{bmatrix}
\text{CAT NP} \\
\text{AGR} \\
\text{NUM 3}
\end{bmatrix}
\text{subsumes}
\begin{bmatrix}
\text{CAT} \\
\text{NP} \\
\text{AGR} \\
\text{PER} \\
\text{SG}
\end{bmatrix}
\]

- Subsumption corresponds to the subset relation in set theory
- The subsumption relation is represented by the binary operator \(\sqsubseteq\).

Feature structures

- **Subsumption** is a *partial ordering relation* between feature structures (i.e. there are pairs of feature structures that neither subsume nor are subsumed by each other)
- There are two cases in which the ordering relation does not hold:
  - if feature structures contain different but compatible information
  - if they contain conflicting information

Feature structures

- The ordering relation between feature structures can be defined as a *semilattice* (with the most general feature structure \([\ ]\) at the top)
Homework

- Task on graphs and AVMs, see assignment file on our class web site

➤ Please hand it in until Saturday night (2nd November, 12 pm); mail to sabine.reinhard@uos.de and vreuer@uos.de