Grammatikformalismen 2010
Lecture Overview

1 Introduction, formal grammars

- What is the role of grammars in a language model?
- Formal grammars:
  - What is a formal grammar?
  - How can it be defined?
  - You need to be able to identify the language that is described by a specific formal grammar
  - What are the different grammar types (type 0 - type 3), how are they defined?
  - What can they describe? What are the limits of different grammar types?
  - Write grammars for type 2 and 3
- Syntactic structure tree:
  - Dominance and Precedence
  - Constraints: root constraint, exclusiveness constraint, crossing constraint

2 Unification, Subsumption and Generalization

- You need to be able to perform unification and generalization operations on attribute-value matrices.

- You need to be able to identify subsumption relations between attribute-value matrices
- How do subsumption and unification relate?

3 PATR-2

- A general understanding of how PATR-2 works is required
- How does PATR-2 relate to formal grammars and the type-hierarchy?
- How does PATR-2 relate to LFG/HPSG (can you name some differences and common points?)

4 Background LFG

4.1 Motivation

What were the motivation to develop Lexical Functional Grammar as a syntactic framework? In particular, what is the motivation for introducing the f-structure

4.2 Overall Architecture

What is the basic architecture used in LFG? Which representations are used? What do individual structures (c- and f-structures) represent? What are basic properties of c- and f-structure?

- It is important to see that the c-structure is, in the first place, a phonological representation, and the f-structure represents grammatical relations

4.3 F-structures

Formal properties should be known, so that you can draw well-formed f-structures and see when an f-structure is not well-formed.

The principles of completeness, coherence and consistence should be known. You should be able to:

- Define/explain them (in your own words is fine)
- recognize when either of these principles is violated in an f-structure

- be able to use them to exclude ungrammatical sentences
- make sure they are respected when analyzing grammatical sentences

For definitions/examples see:

- Lecture of 25.05.2010, slides 21 - 25

4.4 C-structure

What is a constituent? What does the c-structure represent in LFG? You need to be able to draw reasonable c-structures (i.e. no [the [man ran]]): rules used on slides are meant to help you practice, but do not need to be learned by heart.

You need to be able to:

- draw well-formed c-structure trees (that means non-tangled in LFG).
- recognize when a tree is not well-formed.
- provide PS-rules that create the desired tree, or derive a correct tree based on given PS-rules.
- understand the definitions that allow to refer to specific nodes (since this is needed to understand how the correspondence rules work).

4.5 Function ϕ and syntactic correspondences

Function ϕ relates nodes to their associated f-structure. It is used to derive the f-structure from the c-structure. It is **very important** to understand:

- that every node has an associated f-structure
- that $\downarrow$ refers to the f-structure associated with the current node
- that $\uparrow$ refers to the f-structure associated with the mother node
- that syntactic correspondence equations define the f-structure, which means
  - that they must **always** be present
  - that you need to make sure that they provide all information that ends up in the f-structure (recall that words also come with syntactic correspondences) and that every thing that ends up in the f-structure must be defined somewhere

4.6 LFG equations

The basic equations used in LFG (assigning, constraining, existential and negative) should be known and understood. Assigning equations should be known actively, i.e. how to do agreement in LFG, how to assign cases.

4.7 LFG analyses

You should be able to:

- provide lexical entries with appropriate equations (the semantics of lexical entries will be similar to what we have seen in several examples in class, or it will be given)
- provide PS-rules with appropriate annotations
- provide c-structures (appropriately annotated of course) and f-structures
- use assigning equations to capture grammatical properties of the language
- understand (i.e. explain) a given analysis

When asked to provide an analysis:

- make sure that all relevant equations are introduced via lexicon and ps-rules
- check whether your ps-rules can generate your c-structure, and that your f-structure is derived from your c-structure (important!)
- you only need to capture constraints that are pointed out (e.g. if no examples with overt case are provided for English, you need not include case marking in your analysis). We may point out constraints in the following way:
  - contrasting examples: make sure your grammar captures the grammatical sentence(s) and excludes the provided ungrammatical expression(s)
  - glosses: when given a foreign sentence, the information present in glosses should return in the f-structure
  - explicit explanation (e.g. 'make sure your analysis captures subject agreement')
  - if you add constraints that are not required, make sure that they capture facts of the language (e.g. providing subject agreement information for an English regular past tense is wrong!)

5 Typed feature structures

- Why typed feature structures?
- Appropriateness conditions
- What is a type hierarchy, how does it work?
- Subsumption and unification in typed feature structures
- Relevant properties of a typed feature structure:
  - well-typedness
  - completeness
  - type-resolvedness
- Be able to carry out subsumtion and unification on typed AVMs (understand differences between typed and non-typed)

6 HPSG

6.1 Introduction

- What are the main properties of HPSG?
- Make sure you understand slides 2-7 from 15.06.2010
- Understand the use of typed feature structures in HPSG

6.2 Overall architecture of HPSG

What is the overall architecture of HPSG? What are the main parts of an HPSG grammar? (overview: slides 19-21, 15.06.2010)

- *zeichen* ('sign') is the basic type for lexical items (words) and phrases
  - The general architecture of *zeichen* should be known
  - Try to understand what particular attributes stand for, and why they are located at a specific place (this will help remembering the overall architecture)
- Principles:

  - The ID-principle: states that only phrases that correspond to one of the ID-schemata may be formed
  - The head-feature principle: forces the HEAD properties of the HEAD-DTR to be passed up to the mother
  - The Valency principle: "counts" which arguments specified on the HEAD-DTR's valency still need to be found by the mother, and which have been found
  - Semantic principles: make sure the mother phrase has all semantics that are contributed by her daughters, and that her INDEX is identical to the HEAD-DTR's

6.3 HPSG: understanding

- You need to understand how HPSG principles work, and what their function is
- How do HPSG-principles interact with information coming from the lexicon?
  - How does subcategorization work in HPSG?
  - How does agreement work in HPSG?
  - How are adjuncts treated in HPSG?

For analyses:

- Be able to provide analyses for sentences as seen in the exercises
- Be able to identify errors in a presented analysis
- Beware of the details (when is a value SYNSEM, when *sign*)!
- Make sure you know what the SYNSEM abbreviations stand for!

7 LFG and HPSG comparison

- How is information from lexical entries incorporated in more complex expressions in HPSG and LFG?
- How does subcategorization work in LFG and HPSG?
- How are adjuncts treated in HPSG and LFG?

- How do HPSG and LFG make sure that the main information from the syntactic head is passed up to a mother node?
- What is the difference between unification in HPSG and LFG?
- How can you state in HPSG that two values are by definition identical, and how is this done in LFG?

8 Categorial grammar

You need to know

- the basic idea behind categorial grammar
- basic principles of how it works (e.g. what does "/" mean)