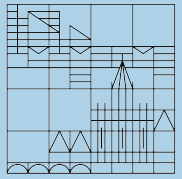
Universität
Konstanz



Grammar Development with LFG and XLE

Miriam Butt
University of Konstanz

Last Time

- Functional Uncertainty
  - Long Distance Dependencies (LDD)
  - Functional Uncertainty Paths
  - Inside-Out Functional Uncertainty
- Word Order Scrambling
  - Shuffle Operator

This Time: Lesson 10

1. Meta-categories and Complex Categories
   - Cross-Categorial Generalizations
   - Rule Parameterization
2. Variables (Local Names)
   - Agreement in Relative Clauses
3. Further Advanced Features of XLE
   - Statistical/Stochastic Information
   - Restriction Operator, Off-Path Constraints
   - ... (there is much, much more)

Meta-categories, Complex Categories

1. Modern syntactic theory has developed certain ideas about how to represent syntactic structures via trees.
   - Some syntactic generalizations are difficult to state within these ideas.
   - Example: Topological Fields (as found in German)
   - **Meta-categories** allow the required generalizations across categories

2. "Families" of rules differ by just a feature or two.
   - This could be encoded via f-structure features, however, it is computationally inefficient.
   - **Complex categories** allow for rule parameterization.

Topological Fields

- German clauses are generally divided into so-called *topological fields*.
  - The **Vorfeld** (pre field) is everything before the finite verb.
  - The **Mittelfeld** (middle field) is everything between the finite verb and the non-finite verbal complex.
  - The **Nachfeld** (post field) is everything after the non-finite verbal complex.

- Generalizations about German syntax can be stated very conveniently via these "areas" of a clause.

Der Tiger hat **im Garten die Katze** gejagt, **die frech war**.
the tiger has in.the garden the cat chased that cheeky was
'The tiger chased the cat that was cheeky in the garden.'

Topological Fields

- Different kinds of constituents can appear in these fields.
- Each of the fields is governed by particular rules.
- In particular, word order in the **Mittelfeld** is fairly free.

Der Tiger hat **im Garten die Katze** gejagt, **die frech war**.
the tiger has in.the garden the cat chased that cheeky was
'The tiger chased the cat that was cheeky in the garden.'

Im Garten hat **der Tiger die Katze** gejagt, **die frech war**.
in.the garden has the tiger the cat chased that cheeky was
'In the garden the tiger chased the cat that was cheeky.'

Meta-categories

- German topological fields can be modeled via **meta-categories**.
- A meta-category is like an ordinary syntactic category in LFG in that:
  - it rewrites/expands to a set of categories.

- It is unlike the other categories in that:
  - it does not appear in the syntactic tree (c-structure)
  - it uses a '=' rather than an '-->' for the rewriting expansion
  - it is called up via the @ prefix, just as templates and metarule macros are.

Meta-categories

- Simplified example for the Middle Field (Mittelfeld).
  - note the '=' and the @ prefix before MITTELFELD in the S rule.
  - This has the effect that no "MITTELFELD" node appears at c-structure.
  - Note also the use of the shuffle operator for scrambling in the German middle field.

```
S -->    @VORFELD
         V2  "finite verb"
         @MITTELFELD
          VC  "non-finite verbal complex"
         @NACHFELD
         (PERIOD).
```

```
MITTELFELD = " either a NP, ADV or PP in any order with shuffle operator"
             NP*: @GF,
             PP*: @P-GF, "adjuncts or obliques"
             ADV*: ! $ (^ ADJUNCT).
```

Meta-categories

- From Meta-category to ordinary category:
  - If one replaced the '=' with a '-->' and took out the @ prefix from the MITTELFELD in the S rule, then a "MITTELFELD" node would appear at c-structure.
  - The modified rules are shown below.

```
S -->    @VORFELD
         V2  "finite verb"
         MITTELFELD
          VC  "non-finite verbal complex"
         @NACHFELD
         (PERIOD).
```

```
MITTELFELD --> " either a NP, ADV or PP in any order with shuffle operator"
         NP*: @GF,
         PP*: @P-GF, "adjuncts or obls"
         ADV*: ! $ (^ ADJUNCT).
```

Meta-categories

- A small **german-toy.lfg** grammar demonstrates the use of meta-categories.
  - Experiment with this grammar.
  - Try the sentences:
    - Der Affe will dem Hund einen Knochen geben.
    - Der Affe will einen Knochen dem Hund geben.
    - Dem Hund will der Affe einen Knochen geben.
  - These are all variations of: "*The monkey wants to give the dog a bone.*"
- Now try doing the type of changes described on the previous slide: change a meta-category into an ordinary category and see the effect on the c-structure.

Demo

Complex Categories

- Complex categories address a computational rather than a theoretical problem.
- In grammar engineering, situations arise where rules share quite a bit of the code, but where parts of the rule are sensitive to certain features and differ in these parts.
- Examples:
  - English auxiliary selection.
  - German finite vs. non-finite verbal phrases.
  - German NPs (Stefanie Dipper. 2003. *Implementing and Documenting Large-Scale Grammars --- German LFG*. Doctoral Dissertation, IMS, University of Stuttgart. Arbeitspapiere des Instituts für Maschinelle Sprachverarbeitung (AIMS), Vol. 9(1).)
- You can experiment with the large-scale English and German grammars at with the INESS XLE-Web functionality.

Complex Categories

- Complex categories work just like ordinary categories.
- Differences:
  - They are “decorated” with features.
  - These features are enclosed in square brackets: e.g., VP[fin]
- German Verb Example:
  - In German matrix clauses need a finite verb after the first constituent.
  - Embedded clauses do not.
  - The overall clausal structure (order and type of arguments and adjuncts) is quite similar.
  - But there are small differences depending on whether there is finite verb or not.
  - So the overall VP rules can be “sensitized” to the feature [fin] – a certain subset will only fire if the verb is finite, another subset will only fire if the verb is non-finite.

Complex Categories

- Situations like the German VP[fin] vs. VP[nofin] could also be solved via regular f-structure annotations.
- However, these are computationally costly.
- Complex categories move feature annotations into the context-free part of the grammar.
- The context-free part is much more efficient computationally.

Demo
XLE-Web

Variables (Local Names)

- Recall that XLE/LFG can handle Long-Distance Dependencies (LDD) quite easily.
- Example: `(^ {COMP|XCOMP}* OBJ) = !`
- Situations arise in which one might want to specify certain information about that object at the end of an LDD path.
- Often this involves agreement of some type:
  - Number, Gender, Person, ...
  - `(^ {COMP|XCOMP}* OBJ NUMBER) = sg`
  - `(^ {COMP|XCOMP}* OBJ PERS) = 3`
- However, there is no guarantee that the functional uncertainty path will always pick out the same OBJ when applying the NUMBER and the PERS information.

Variables (Local Names)

- **Solution**: Introduce variables that "point" to a certain f-structure.
- In our example:
  - bundle all the agreement information under one feature: AGR
  - point to this f-structure via a local variable name
  - variables in XLE are generally prefixed with a %
    - `(^ {COMP|XCOMP}* OBJ AGR) = %Agr`
  - Now the f-structure instantiated by this AGR feature can be referred to via the variable name:
    - `(^ {COMP|XCOMP}* OBJ %Agr NUMBER) = sg`
    - `(^ {COMP|XCOMP}* OBJ %Agr PERS) = 3`

Variables (Local Names)

- Variables are also useful for more local relations.
- Typical example in ParGram: **relative clauses.**
- Situation: several different agreement dimensions need to be checked.
- Solution: same as on previous slide, just without the functional uncertainty path.

```
(^ AGR) = %Agr
(^ %Agr NUMBER) = sg
(^ %Agr PERS) = 3
(^ %Agr GEND) = fem
```

Further Advanced Features

- XLE is a very complex piece of software (kudos to John Maxwell).
- As a grammar development platform it offers features that go well beyond the usual capabilities of state-of-the-art parsers/generators.
- This course has introduced
  - basic features of XLE
  - also show-cased some of the more advanced features.
- However – there are MANY more features.
- The features are too many to cover in one course. For more consult:
  - the XLE documentation
  - LFG/XLE grammar engineering publications
  - the on-line Wiki documentation

Further Advanced Features

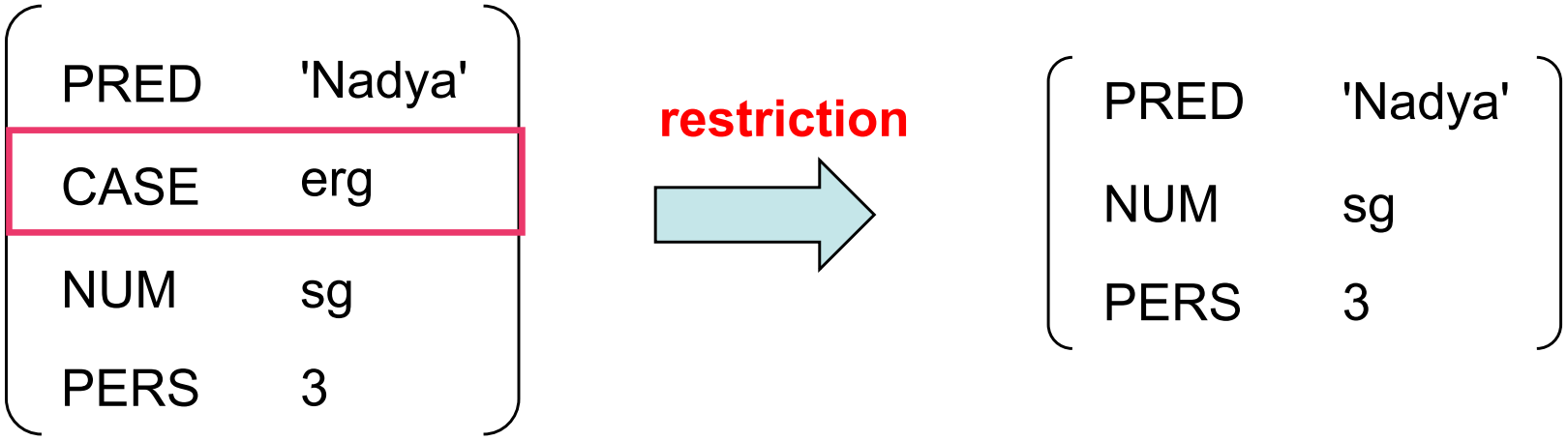
- Now: three more advanced features
  - Off-path constraints
  - Restriction Operator
  - Integration of information about statistical distribution/preferences
- These three features have been used in the ParGram grammars.
- The off-path constraints and the statistical preferences have been extremely useful for constraining ambiguity.
- The Restriction Operator has been used to compose predicates (PREDs) – necessary for handling complex predication correctly.

Off-path constraints

- Again, recall the long distance dependencies.
- And recall that one is able to formulate functional uncertainty paths.
  - `(^ {COMP|XCOMP}* OBJ) = !`
- One might want to state certain constraints as to what features/properties the COMP, XCOMP or OBJ can(not) have.
- This can be done via an off-path constraint that is added to the functional uncertainty path.
- For example, one could require that the XCOMPs that are encountered within the path cannot be in focus.
  - `(^ {COMP|XCOMP: ~(-> FOCUS); }* OBJ) = !`

Restriction Operator

- The **Restriction Operator** allows for the specification of "suppression" of information in an f-structure.
- LFG is non-derivational and monotonic so no information is actually thrown away.
- However, information can be left out of the main f-structure.
- Simple Example: Restrict out the CASE feature

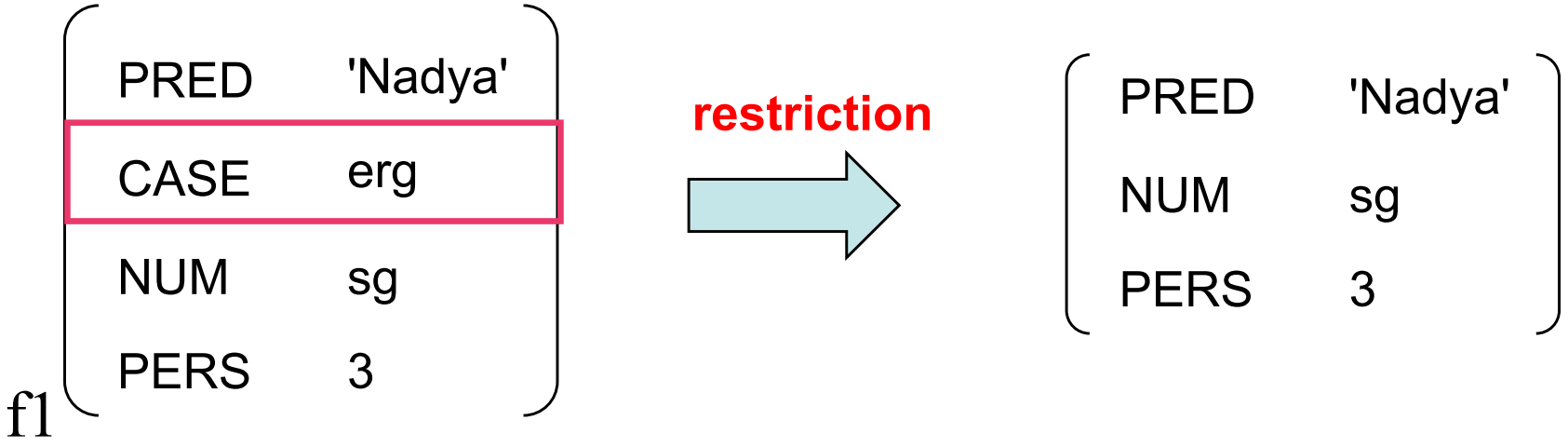


Restriction Operator

- The Restriction Operator is written as: /
- The rule for restricting out the case information below is:

  ^ /CASE

- The "^" points at a given f-structure (f1).
- The / restricts out the case feature from this f-structure.
- That is: f1 below is everything minus the case information.



Urdu Complex Predicate Example

- The Restriction Operator has been used to model complex predication in which two or more predicates combine to form one predication.

amraa=ne ravii=ko **kitaab dekHne dii**
Amra=Erg Ravi=Dat book look.at gave
'Amra let Ravi look at the/a book.'

- The verb *de* ('give') does not function as a full verb in this case, but means 'let'.
- It is analyzed as a light verb.
- The SUBJ (Amra) is contributed by the light verb.
- The OBJ (book) is contributed by the main verb (look.at).
- Both subcategorize for the permissee (Ravi).

Urdu Complex Predicate Example

- The lexical entry for *de* 'let' subcategorizes for a subject and a second argument that remains to be determined.
- This is encoded via a **variable** (recall the % notation).
- In this case: %PRED2

dE Vlight XLE (^ PRED) = `dE<(^ SUBJ), %PRED2>'

- This lexical entry combines with that of the main verb in the syntax – in this case a normal transitive verb.

dEkH V XLE (^ PRED) = `dEkH<(^ SUBJ), (^OBJ)>'

amraa=ne ravii=ko **kitaab dekHne dii**
Amra=Erg Ravi=Dat book look.at gave
'Amra let Ravi look at the/a book.'

Urdu Complex Predicate Example

- The combination (in this case) is via a syntactic rule.
- NB: The restriction analysis can also be applied within the lexicon, for example with respect to morphological causatives.

```
V -->     V                                                    Vlight
       !/PRED/SUBJ/VTYPE= ^/PRED/SUBJ/VTYPE
    (^ PRED ARG2) =  (! PRED)
       (! SUBJ) =  (^ OBJ-GO)
```

- The rule says that the second argument of the light verb (the variable %PRED2) is the PRED of the main verb.
- The subject of the main verb is reinterpreted as the indirect object (OBJ-GO = *Ravi*).
- The SUBJ is restricted out of the main verb's f-structure, as are various other bits of information (VTYPE).

Restriction Operator

- Understanding (and debugging) the Restriction Operator is not trivial.
- Generation may also be problematic.

```
V -->     V                                                    Vlight
      !/PRED/SUBJ/VTYPE= ^/PRED/SUBJ/VTYPE
   (^ PRED ARG2) =  (! PRED)
      (! SUBJ) =  (^ OBJ-GO)
```

```
dE   Vlight  XLE        (^ PRED) = `dE<(^ SUBJ), %PRED2>'
```

```
dEkH   V   XLE   (^ PRED) = `dEkH<(^ SUBJ), (^OBJ)>'
```

amraa=ne ravii=ko **kitaab dekHne dii**
Amra=Erg Ravi=Dat book look.at gave
'Amra let Ravi look at the/a book.'

Demo

Statistical/Stochastic Information

- So far: XLE as an exclusively rule-based system.
- We have learned to do a combination of:
  - implementing linguistic analysis
  - understanding grammar engineering requirements (regression testing, efficiency, disambiguation)
- But XLE also allows for the integration of statistical and stochastic information.
- Given a grammar and a corpus, one can calculate *property weights* over parses and use that to identify the most probable parse.

Demo
English Grammar

Chart Pruning

- It is also possible to eliminate possible (but eventually bad) analyses early one via **Chart Pruning**.
- This eliminates c-structures that are eventually bad before processing the f-structure contraints associated with them.
- Can increase grammar speed by 30%-40%.
- One can also easily parametrize the grammar to have a pruned and a non-pruned version.
  - The pruned version is generally faster but may lose some analyses.
  - The non-pruned version is slower but will have more analyses.
  - Norwegian grammar:
    - uses pruned version generally for tree banking
    - if a sentence cannot be parsed, switches to non-pruned version

Discriminants in INESS

- The INESS infrastructure for XLE is recent and is growing.
- Main purpose: provide support for tree banking.
- XLE-Web interface allows for:
  - Upload and on-line use of XLE grammars (parsing/generation).
  - Comfortable on-line disambiguation of parses via **discriminants**
    - c-structure
    - f-structure
- Banking (storage) of the desired structure.
- Updates of banked structures when grammars have changed.

Demo

Here Ends the Basics

- This concludes Lesson 10.
- There is no practical work for this lesson.
- You should be able to begin writing your own grammars now.
- Remember to consult the XLE documentation!
- Remember to do regression testing!
- For more information, see the XLE and ParGram forum and Wiki entry documentation.
- Explore the INESS XLE infrastructure.

Acknowledgements

- This material was developed over several years and owes much to several people who have contributed or co-taught over the years.

- Particularly:
  - Tracy Holloway King (A9)
  - Martin Forst (NetBase)
  - Sebastian Lory (née Sulger, University of Konstanz)
  - Many teaching assistants at the University of Konstanz.