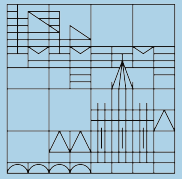
Universität
Konstanz



Grammar Development with LFG and XLE

Miriam Butt
University of Konstanz

Last Time

- Adjuncts:
  - Adjectives, Adverbs
  - PPs
- Punctuation/Tokenization

This Time: Lesson 5

1. Integration of Optimality Marks
   - Disambiguation
   - Grammar Parametrization
   - Generation
2. Pronouns

PP Ambiguity

- In the last exercise, you were asked to implement various types of PPs.
- PPs are notorious for causing ambiguities in grammars.
- General term: t**he PP attachment problem**
- Example:
  - » The zookeeper saw the monkey with the telescope.
- Constraining such ambiguity is a challenge.
- One way to help constrain the ambiguity in XLE is the use of OT-Marks.

PP Ambiguity

- Corpus studies have shown that PPs are preferentially attached locally.
- That is, the preference is to attach the PP *with the telescope* to *monkey*.
  - » The zookeeper saw [the monkey with the telescope].
- But this is only a **preference**.
- It is not a hard and fast rule of the type we have been writing so far.

Harnessing Optimality Theory

- Optimality Theory (OT) was invented within theoretical linguistics.
- Sees a grammar as a system of constraints.
- Classic OT only knows constraints, i.e. dispreferences.
- OT as implemented in XLE uses both dispreference marks (default) as well as preference marks (prefixed with +)

```
+Mark = preference
 Mark = dispreference
```

Harnessing Optimality Theory

- Classic OT assumes a simple hierarchy of constraints.
- OT as implemented in XLE uses a “structured hierarchy”.
- That means that the strength of (dis)preference can be set variably.
- The effect of individual OT-marks can differ markedly.
- OT-Marks can be added anywhere:
  - Rules
  - Lexicon
  - Templates

Rule Annotation (O-Projection)

- Common errors can be dispreferred rather than completely ruled out.
- Example: subject-verb agreement for CALL

```
Verb3Sg = { (^ SUBJ PERS) = 3
            (^ SUBJ  NUM) = sg
          | @(OTMARK BadVAgr) }
```

- Disprefer parses of ungrammatical structure
  - tools for grammar writer to rank rules
  - two+ pass system

OT Marks

- OT marks are projected to a separate projection, the o-structure (o::)
- The o-structure (unlike c- and f-structure) is not structured.
- It is treated as a “bag” of OT marks.
- That is, all OT marks are collected up in a set.

```
OTMarkName $ o::*
```

OPTIMALITYORDER

- Part of the grammar header
- Can be modified for grammar customization
- OPTIMALITYORDER is for parsing.
- GENOPTIMALITYORDER is for generation.
- OT marks can be organized into groups of equal rank via round brackets.

```
OPTIMALITYORDER  DisprefMark1
  +PrefMark1 DisprefMark2
  (DisprefMark3 DisprefMark4)
```

Example: Ranking Parses

- Start with the leftmost OT-Mark.
- Keep parses with fewest instances of DisprefMark1; consider all others suboptimal.
- Among remaining parses, keep those with most instances of PrefMark1; consider all others suboptimal.
- Among remaining parses, keep those with fewest instances of DisprefMark2; consider all others suboptimal.
- Etc.

```
OPTIMALITYORDER  DisprefMark1
    +PrefMark1 DisprefMark2
```

Examples: Potential Applications

- Prefer OBL interpretations of PPs over ADJUNCT interpretations
  - » The zookeeper waited for the gorilla.
- Prefer ditransitive subcategorization frames over transitive ones.
  - » The girl gave her brother money.
- Prefer grammatical constructions, but also allow ungrammatical ones (e.g., subj-verb agreement for CALL applications).

Demo

grammar4.lfg
testsuite4.lfg

OT-Marks to constrain PP ambiguity

OT Ranking with Special Marks

- **Order of Marks**: Mark3 is preferred to Mark4

  `OPTIMALITYORDER Mark4 Mark3 +Mark2 +Mark1.`

- **NOGOOD Mark:** Marks to the left are always bad. Useful for parameterizing a grammar with respect to certain domains.

  `OPTIMALITYORDER Mark4 NOGOOD Mark3 +Mark2 +Mark1.`

- **STOPPOINT Mark:** slowly increases the search space of the grammar if no good solution can be found (multipass grammar).

  `OPTIMALITYORDER Mark4 NOGOOD Mark3 STOPPOINT Mark2 STOPPOINT Mark1.`

NOGOOD OT Marks

- If (part of) a lexicon entry or a rule projects an OT mark that is listed to the left of `NOGOOD` in `OPTIMALITYORDER`, that part of the grammar is **deactivated**.

- Can be used for expensive constructions or particular readings of ambiguous lexical items which are known to be of no/little importance in the application domain.

- Grammar Parameterization!

STOPPOINT OT Marks

- Intended for better performance.
- Only beneficial when used cautiously.
- (Parts of) lexical entries and rules marked with STOPPOINT OT marks are not used for first parsing attempt.
- If first attempt is unsuccessful, the parser activates those lexicon or rule parts and makes a second attempt.

- Example: `Mark1 Mark2 STOPPOINT`

Generation

- XLE can generate strings from well-formed f-strs.
- GENOPTIMALITYORDER can be different from OPTIMALITYORDER.
- In the ParGram grammars, the orders and OT-Marks uses generally differ.
- This is comparable to the situation with transducers:
  - typically, the generation tokenizer is more restrictive than the parsing tokenizer
  - Example: white space or commas (typos): ,,, instead of ,

Generation

Two ways of generating from an f-structure in XLE.

1. Go to the “Commands” menu of your f-structure window and select “Generate from this FS”.
2. At the XLE command line type in:
   regenerate {sentence to be parsed}

Demo

grammar4.lfg
testsuite4.lfg

Generation

Pronouns

- So far, we have been using full NPs in all the examples.
- It would be nice to be able to use pronouns as well.
- So, will now determine what that should look like.
- And use the problem to illustrate the basic, typical steps involved in grammar engineering.

Grammar Engineering – First Steps

- What should the f-structure be?
- What should the c-structure be?
- After having determined this: implement
  - the rules with functional annotations
  - the lexical entries with POS category and functional information
  - add templates where appropriate
- Remember that you need to think about both:
  - c-structure: context free rules to span the words of the sentence
  - f-structure: annotations to produce the correct functional information

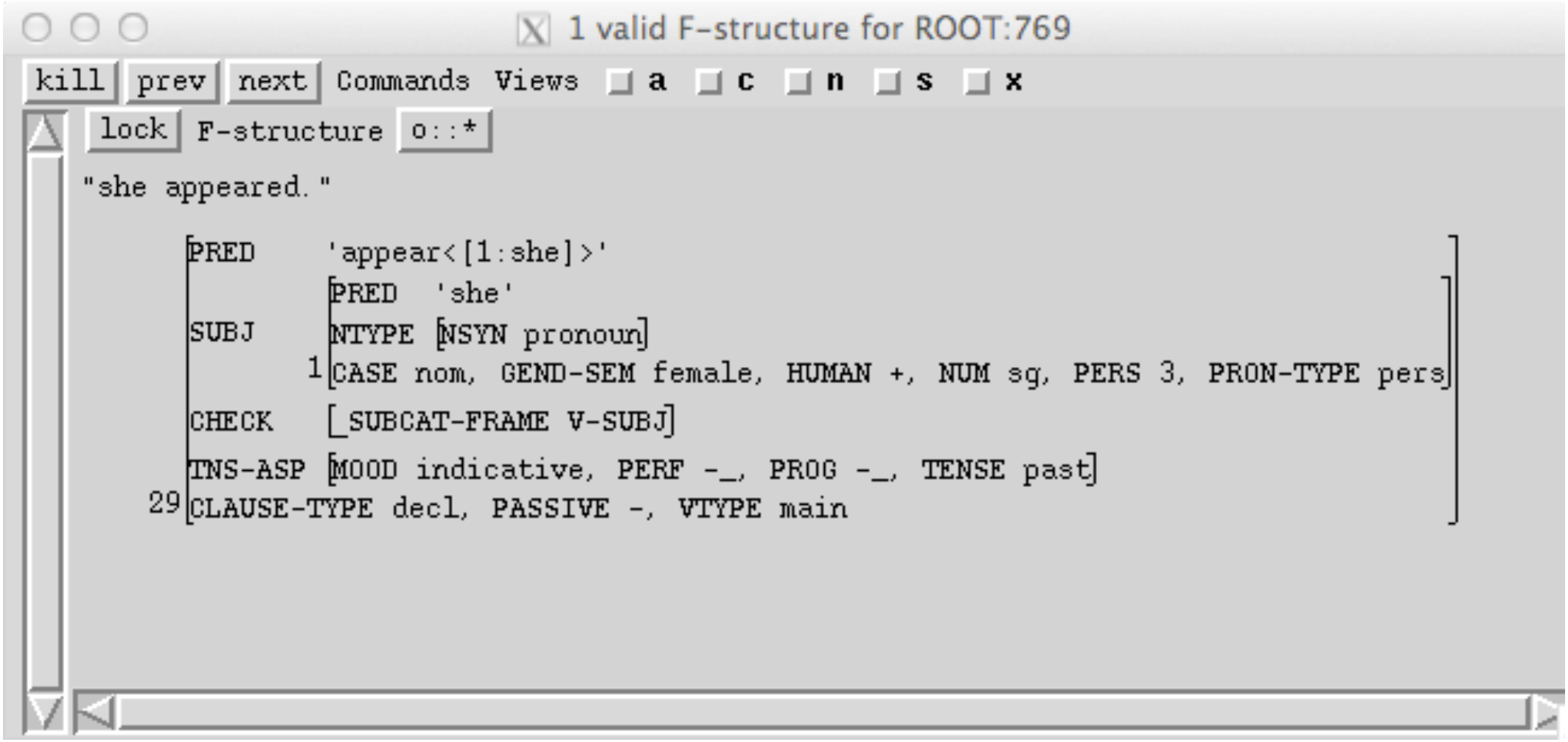
Determining F-Structure

- What pronouns are used in the language?
- English is fairly easy, only a handful and no morphology.
- Basic personal pronouns
  - *I, me, we, us, you, she, he, it, her, him, they, them*
- What do they encode?
  - Number (singular *I* vs. plural *we*)
  - Gender in some cases
  - Case (nominative *she* vs. accusative *her*)
  - Person (1st person *I* vs. 2nd person *you*)
- What else might one need from a grammar engineering perspective?

Determining F-Structure

- In general, if you are working on a new construction, it is a good idea to look at existing work for guidance.
- Good place to look: the English ParGram grammar (or grammars closely related to the language you are working on).
- Currently this is most easily available on the INESS XLE web interface.

English ParGram Grammar Example



Determining C-Structure

- Pronouns substitute for NPs.
- So, what needs to be done is to implement a disjunction in the NP rule (simplified below).

```
NP --> { (D)
         AP*: ! $ (^ ADJUNCT);
         N
         PP*: ! $ (^ ADJUNCT)
       | PRON}.
```

- Then you need to add pronouns to your lexicon with the right POS and one or more elegant templates.

```
we   PRON *   @(PRON we 1 pl pers).
```

Practical Work

- This concludes Lesson 5.
- The practical work you should do now is detailed in Exercise 5.
- You will practice with
  - pronouns
  - constraining PP ambiguity by using OT-marks
  - generation