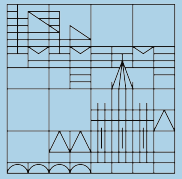
Universität
Konstanz



Grammar Development with LFG and XLE

Miriam Butt
University of Konstanz

Last Time

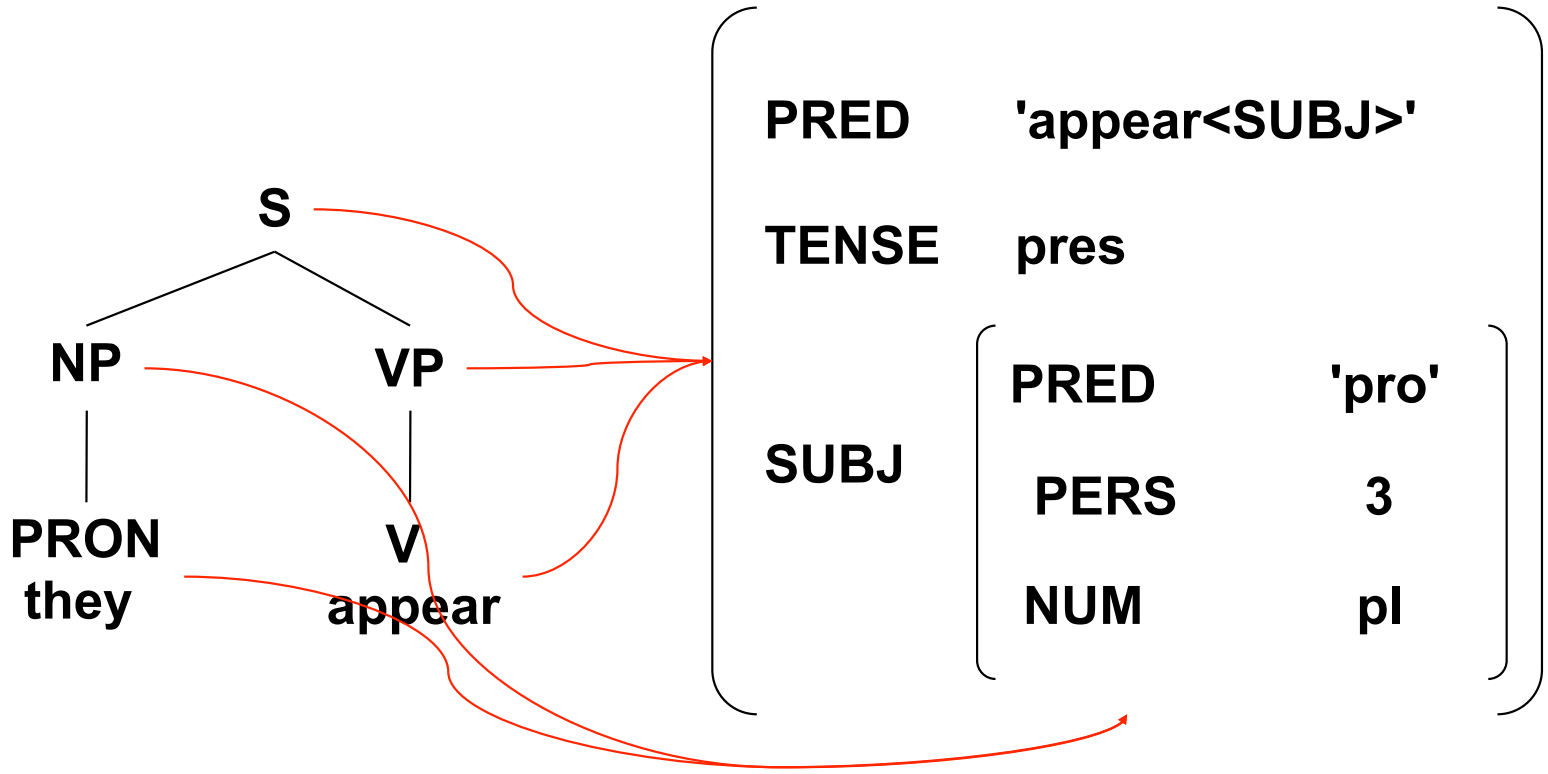
- Motivation for Deep Grammar Development
- Application Demos
- Basic Information about XLE and ParGram
- Practical Work:
  - Install XLE
  - Experiment with INESS Web-XLE

This Time: Lesson 2

1. LFG/XLE Basics
   - Context Free Phrase Structure Rules
   - Grammatical Relations (Grammatical Functions)
   - Lexical Entries
   - Functional Annotations:
     - Unification (Consistency)
     - Completeness and Coherence
   - Templates
2. Practical Work: The XLE Walkthrough

Basic LFG

- Constituent-Structure (c-str): tree
- Functional-Structure (f-str): Attribute Value Matrix
- Information projected from c-str to f-str via Functional Annotations



Grammar Components

An LFG Grammar typically contains:

- Annotated phrase structure rules (S --> NP VP)
- Lexicon (verb stems and functional elements)
- Templates
- Finite-State Morphological Analyzer
- A version of Optimality Theory (OT):
  - used as a filter to restrict ambiguities
  - and/or parametrize the grammar
  - debugging
- Also possible:
  - disambiguation feature file based on statistical

Grammar sections

- Configuration section
- Rules, templates, lexicons
- Each has:
  - version ID
  - component ID
  - XLE version number (1.0)
  - terminated by four dashes ----
- Example:

```
STANDARD ENGLISH   RULES   (1.0)


----
```

Basic configuration section

```
TOY ENGLISH CONFIG (1.0)
ROOTCAT S.
FILES .
LEXENTRIES (TOY ENGLISH).
RULES (TOY ENGLISH).
TEMPLATES (TOY ENGLISH).
GOVERNABLERELATIONS SUBJ OBJ OBJ2 OBL COMP
    XCOMP.
SEMANTICFUNCTIONS ADJUNCT TOPIC.
NONDISTRIBUTIVES NUM PERS.
EPSILON e.
OPTIMALITYORDER
      NOGOOD.
---
```

Syntactic rules

- Annotated phrase structure rules

Category --> Cat1: Schemata1;
Cat2: Schemata2;
Cat3: Schemata3.

  - Example

S --> NP: (^ SUBJ)=!
(! CASE)=NOM;
VP: ^=!.

XLE vs. LFG Notation

- XLE uses a notation that is slightly different from standard LFG.
- The reason is that XLE is more “ASCII” conscious.
- Also see the file basic-notation.pdf

Meaning	LFG	XLE
Functional annotation pointing to mother node	↑	^
Functional annotation pointing to current node	↓	!
Element of a set	∈	$

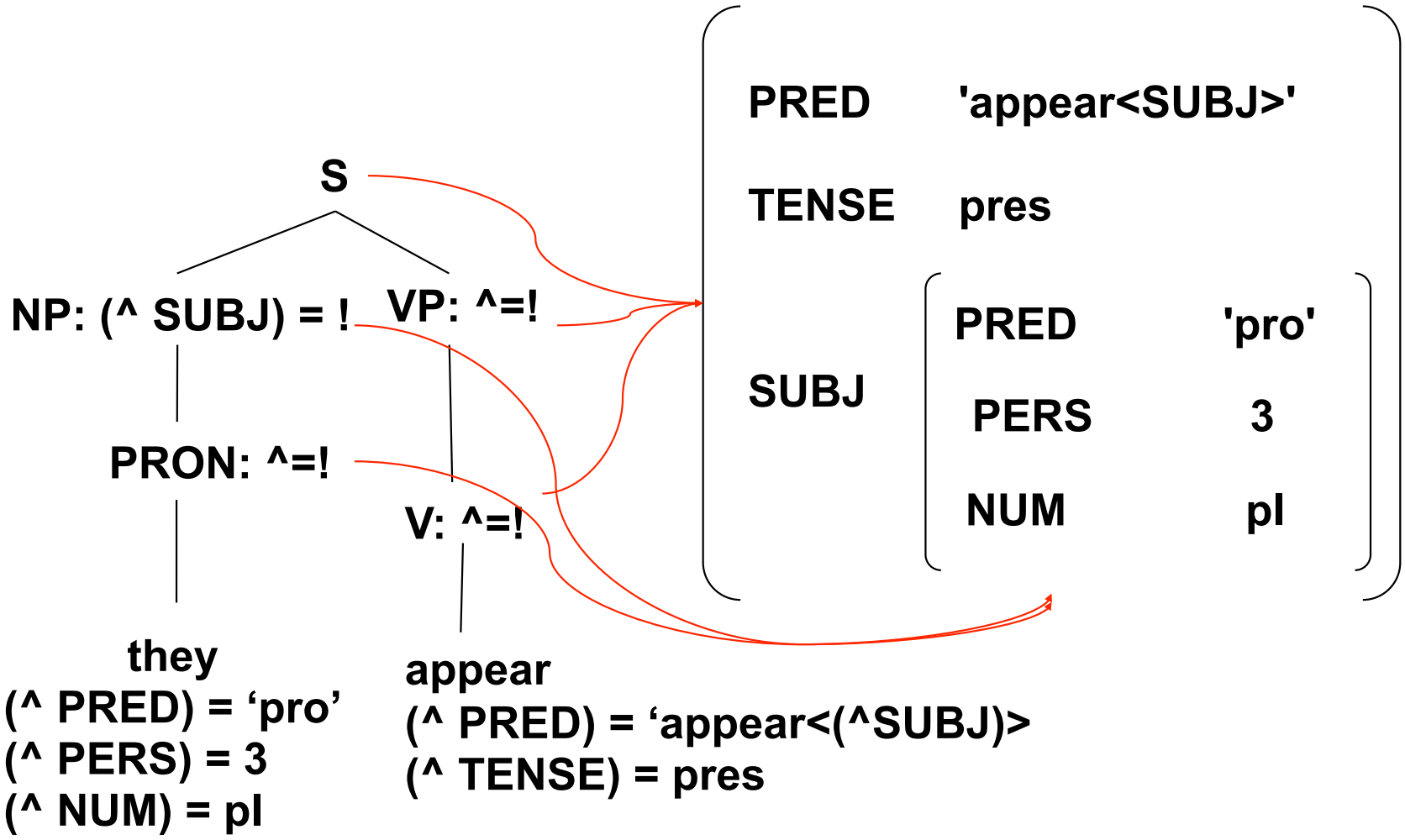
Another sample rule

```
                                        "indicate comments"
VP --> V: ^=!;                          "head"
        (NP: (^ OBJ)=!                  "() = optionality"
             (! CASE)=ACC)
        PP*: ! $ (^ ADJUNCT).           "$ = set, * = Kleene star"
```

VP consists of:

- a head verb
- an optional object
- zero or more PP adjuncts

Annotations and Projection



Lexicon

- Basic form for lexical entries:

  word Category1 Morphcode1 Schemata1;
  Category2 Morphcode2 Schemata2.

  - Examples

  walk V * (^ PRED)='walk<(^ SUBJ)>';
  N * (^ PRED) = 'walk' .

  girl N * (^ PRED) = 'girl'.

  kick V * { (^ PRED)='kick<(^ SUBJ)(^ OBJ)>'
  |(^ PRED)='kick<(^ SUBJ)>'}.

  the D * (^ DEF)=+.

Grammatical Relations in LFG

- Grammatical Relations are a central component of any LFG analysis.
- They are generally referred to as Grammatical Functions (GF) in LFG.
- Predicates are taken to have subcategorization frames, e.g.

  (^ PRED)='kick<(^ SUBJ)(^ OBJ)>'

- Deciding on which GFs are subcategorized for by a given predicate is a matter of linguistic analysis.

Grammatical Relations in LFG

- The set of GFs in LFG is:
  - SUBJ (subject)
  - OBJ (object)
  - OBJ2 or OBJ-TH (or OBJ_{θ}, secondary objects)
  - OBL (tends to be subcategorized for PPs in English)
  - XCOMP (non-finite complement clause)
  - COMP (finite complement clause)

- See Dalrymple (2001, Ch. 2) for background, discussion and possible tests to determine GFs across languages.

Surface (c-str) vs. Deep Structure (f-str)

- Languages vary substantially in terms of word order and constituency requirements.
- C-structure representations for languages will therefore vary substantially.
- However, all standardly transitive clauses should contain a SUBJ and an OBJ at f-structure.
- Example: Warlpiri vs. English

Surface (c-str) vs. Deep Structure (f-str)

- The Australian language Warlpiri allows for very free word order.
- The auxiliary/finite element just needs to be in 2nd position.

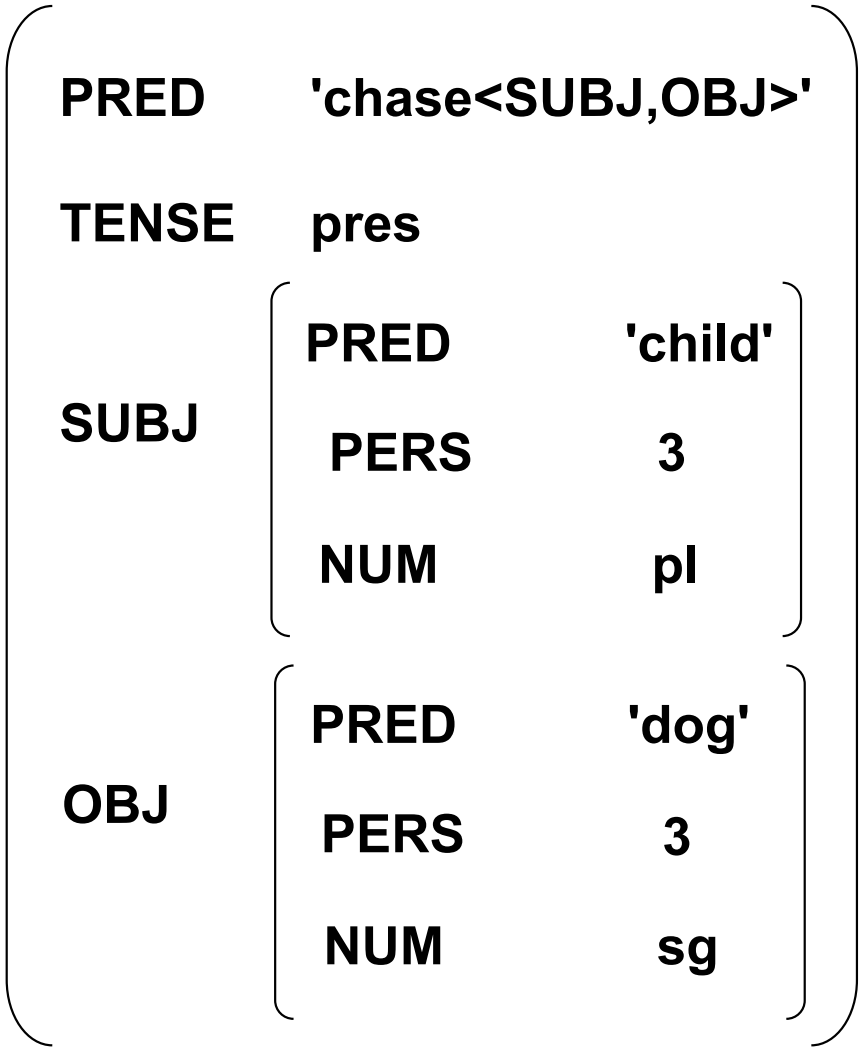
kurdu-jarra-rlu	kapala	maliki
child-Dual-Erg	Aux.Pres	dog.Abs
wajipili-nyi	wita-jarra-rlu	
chase-NonPast	small-Dual-Erg	

‘The two small children are chasing the dog.’

- In contrast, English is
  - restrictive about allowing NPs to be discontinuous
  - restrictive about word order possibilities

Surface (c-str) vs. Deep Structure (f-str)

- However, the f-structure representation for both languages is quite similar.



Basic LFG – Wellformedness

LFG operates with three basic wellformedness principles:

- Consistency

  Every attribute can only have one value.

- Completeness

  All grammatical functions listed in a subcategorization frame must be present and they must have a PRED (themselves be predicational). Exception: Non-Thematic Arguments (“It is raining.”)

- Coherence

  All grammatical functions in an f-structure must be licensed by a subcategorization frame.

Demo – Parsing Sentences

- The following demo shows how to load a grammar into XLE and to work with it.
- You can download the grammar file and work along with the examples – it is called grammar1.lfg.
- The basic LFG principles of Consistency, Coherence and Completeness are demonstrated.
- Basic errors occurring with parsing are also demonstrated, plus tips for debugging the grammar.

Testsuites

- The demo also features the use of a *testsuite*.
- This is a file that contains the sentences that your grammar can parse
- (and also ones that your grammar should not be able to parse).
- You should start working with a testsuite right away.
  - It saves typing.
  - It keeps a record of what your grammar could and could not do.

Testsuites

- When you make changes to your grammar, you should always check to make sure that the grammar can still parse all the sentences you had implemented before.
- This is called *regression testing*.
- Regression testing is a very important part of any software development, including grammar development.
- The testsuite used for the demo is testsuite1.lfg.
- ALWAYS work with a testsuite.

Demo

grammar1.lfg
testsuite1.lfg

Templates

- grammar1.lfg contains
  - a c-structure rules with annotations
  - a lexicon
- It does not contain any templates.
- We introduce templates very early on because they
  - save time in grammar writing
  - encode linguistic generalizations well
  - assist in keeping your grammar clean and organized (less debugging work likely)

Templates

- Abbreviatory device for f-annotations

- Have to be defined in a separate section of the grammar, the TEMPLATES section.

- Example for a template:

```
PRES3SG =    (^ TENSE) = pres
          (^ SUBJ PERS) = 3
          (^ SUBJ NUM) = sg
```

- Are invoked by means of the prefix `@`, e.g.

```
@PRES3SG
```

Templates

- Express generalizations
  - in the lexicon
  - in the grammar
  - within the template space

No Template

```
girl N * (^ PRED)='girl'
         { (^ NUM)=SG
           (^ DEF)
          |(^ NUM)=PL}.
```

With Template

```
TEMPLATE: CN = { (^ NUM)=SG
                 (^ DEF)
                |(^ NUM)=PL}.

girl N * (^ PRED)='girl' @CN.
boy N * (^ PRED)='boy' @CN.
```

"{ | } expresses a disjunction"

Templates

- Parameterize template to pass in values

```
CN(P) = (^ PRED)='P'
          { (^ NUM)=SG
            (^ DEF)
          | (^ NUM)=PL}.
```

```
girl N * @(CN girl).
boy N * @(CN boy).
```

- A template can call other templates

```
INTRANS(P) = (^ PRED)='P<(^ SUBJ)>'.
TRANS(P) = (^ PRED)='P<(^ SUBJ)(^ OBJ)>'.
OPT-TRANS(P) = { @(INTRANS P) | @(TRANS P) }.
```

Templates

- Make it possible to create an “inheritance” hierarchy of (f-annotations of) lexical entries.
- But nothing in LFG or XLE forces you to organize (f-annotations of) lexical entries in an inheritance hierarchy.
- This is fundamentally different between LFG/ XLE and HPSG and its implementations (LKB, Trale, etc.)
- In practice we have found that template calls have a maximum embedding of up to three levels.

Practical Work

- This concludes Lesson 2.
- The practical work you should do now is to go through the XLE Walkthrough.
- This is part of the XLE documentation.
  - It is very informative and very thorough.
- Be prepared to spend some time on this exercise.
- Details can be found in Exercise 2.