Grammar Development with LFG and XLE

Miriam Butt
University of Konstanz

Last Time

- imperatives (empty nodes)
  - Parsing
  - Generation
- coordination
  - regular expression macros
  - metarulemacros

This Time: Lesson 7

1. Integrating a Finite-State Morphological Analyzer
   - Morphology Section: Analyze
   - Sublexical rules
   - Sublexical entries
   - The -unknown entry
2. The XLE Lexicon Lookup Model

Interfacing finite-state transducers

- Maintaining a full-form lexicon is tedious.
- Many lexicon entries are the same (e.g., nouns).
- Is there a way to get information from somewhere about
  - the category of a word (Part-of-Speech; POS)
  - along with information about morphosyntax (tense, mood, case, number, person, etc)

Finite-State Morphologies (FSM)

- Yes – there is!
- Finite-state morphological analyzers
  - often commercially produced
  - often available from research institutions
  - very easy to implement on your own
  - finite amount of time necessary to build one
  - efficient
  - can be composed with tokenizers
  - easy to integrate into XLE

Software for Implementing FSMs

- XFST (PARC/Xerox)
  - www.fsmbook.com
  - Kenneth R. Beesley and Lauri Karttunen. 2003. *Finite State Morphology*. CSLI Publications.

- FOMA: Finite-State Compiler and C Library
  - Hulden, Mans. 2009. Foma: a finite-state compiler and library. *Proceedings of the 12th EACL Conference*, 29-32.

- SFST
  - Helmut Schmid, A Programming Language for Finite State Transducers, Proceedings of the 5th International Workshop on Finite State Methods in Natural Language Processing (FSMNLP 2005), Helsinki, Finland

Software for Implementing FSMs

- OpenFST (Google Research and NYU)
  - HFST (Helsinki)
  - Kleene Programming Language (Beesley)

Interfacing finite-state transducers

- Cascade of finite-state transducers is specified in MORPHOLOGY section.
- At least two subsections:
  - TOKENIZE
  - ANALYZE
- By default, the transducers listed are used both for parsing and for generation.
- This behavior can be altered by prefixing the names of transducer files with `P!` or `G!`

Tokenization

- Recall that in the first grammars only white spaces were considered as token boundaries.
- However, there are more kinds of token boundaries in real-word text:
  - Punctuation has to be split off the preceding token.
  - Some white spaces should not be treated as token boundaries, e.g. “Sri Lanka” (MWE).
  - Upper-case letters at sentence beginnings should optionally be lower-cased.
- A finite-state tokenizer takes care of these things.

Tokenization

Integrated from Starter Grammar when we did punctuation.

```
DEMO ENGLISH MORPHOLOGY (1.0)

TOKENIZE:
P!basic-parse-tok.fst G!default-gen-tokenizer.fst
```

Finite-state morphologies

Map surface forms to canonical form (lemma) and series of “morphological” tags.

Examples:

`rode`	`ride +Verb +PastTense +123P`
`rides`	`ride +Verb +Pres +3sg`
	`ride +Noun +Pl`
`children`	`child +Noun +Pl`

Both generation and parsing directions available.

Interfacing Finite-state Morphology

- From XLE’s perspective, the output of a FSM needs to be parsed, just like a string.

```
ride +Verb +Pres +3sg
```

- So we need a (sublexical) rule that can parse a given sequence of lemma+tags.
- This means that we need lexical entries for the lemma and all of the tags.
  - They are treated as (sublexical) terminal nodes.
  - We can also code functional information in the lexical entries.

Interfacing Finite-state Morphology

- Sublexical lexicon entries look just like regular lexicon entries.
- **Difference:** morphcode `XLE` instead of `*`.

```
+Pres   TNS    XLE @VPRES.
```

- This signifies that the lexical look up is being done with reference to a morphological analyzer.
- In contrast, `*` tells XLE to take the lexical item “as is” (i.e., as a fully inflected lexical item).

Lexical Look up in XLE

- XLE has a very powerful and complex mechanism for lexical entry look up.
- Can combine entries from different files and block readings.
- The * is useful for items that the morphological analyzer cannot deal well with in terms of grammar writing .
  - Functional elements with a specialized role in the grammar (e.g., auxiliaries).
  - Elements that do not inflect (much).
  - Punctuation (take “as is”, no morphological analysis)

Sublexical Rules

- The sublexical entries corresponding to the tags produced by the FSM are treated as sublexical c-structure categories (terminal nodes).
- They must be parsed by sublexical rules
  - These look like regular rules.
  - Can have f-annotations like regular rules.
  - **Difference:** Sublexical categories are marked with the suffix `_BASE`.

Sublexical Rules and Lexical Entries

```
V --> V-S_BASE
      V-POS_BASE
      { TNS_BASE
        PERS_BASE
      | ASP_BASE}.
```

```
ride +Verb +Pres +3sg
```

```
+Verb  V-POS XLE .
+Pres  TNS   XLE @VPRES.
+Prog   ASP XLE @VPROG.
+3sg   PERS  XLE @S-AGR.
ride   V-S   XLE @(TRANS %stem).
```

Demo

grammar6.lfg
testsuite6.lfg

verbs via FSM

The unknown Entry

- Lemmas with non-predictable subcategorization frames must be listed in the lexicon.

```
hate   V-S   XLE @(TRANS %stem).
donate V-S   XLE @(DITRANS %stem).
```

- Other lemmas with predictable information can be dealt with by the `-unknown` entry

```
-unknown A-S XLE @(PRED %stem);
         N-S XLE @(PRED %stem).
```

The unknown Entry

- The unknown entry is a very powerful device.
- Saves effort of individually specifying lexical items that belong to the same class.
- In our grammars it should now become unnecessary to specify nouns, adjectives and adverbs separately.
- Verbs, auxiliaries, determiners and pronouns contain specialized information.
- For these it is better to write explicit lexical entries.

XLE Lookup Model

- Recall: we can have only one entry per headword per lexicon section.
- But – there are situations in which the same headword may be covered by
  - – an explicit entry
  - – and by an`-unknown` entry
- In order to for allow this, XLE uses **edit entries**.
- The possibilities allowed by XLE are very complex – see the XLE documentation.
- Here, just two examples.

XLE Lookup Model – ETC

- `ETC` signals that other entries are allowed.
- So if another entry for the same headword is encountered, this entry is added to the entry that has already been processed.
- Example: noun version of *sleep* coming from the unknown entry is added to the explicitly specified verb version.

```
sleep    V-S @(INTRANS sleep); ETC.
-unknown N-S @(PRED %stem).
```

XLE Lookup Model – ONLY

- `ONLY` signals that this is the only entry.
- So if another entry for the same headword is encountered, this entry is ignored by XLE.
- Example:
  - the noun version of *sleep* coming from the unknown entry is ignored
  - only the verb version of *sleep* is used by the grammar.

```
sleep    V-S @(INTRANS sleep); ONLY.
-unknown N-S @(PRED %stem).
```

Practical Work

- This concludes Lesson 7.
- The practical work you should do now is detailed in Exercise 7.
- You will practice with
  - integrating a finite-state morphological analyzer
  - writing sublexical rules for nouns and adjectives
  - writing sublexical entries for the tags associated with nouns and adjectives