Creating LRs with FSTs Part V

Irregular forms and advanced morphotactics

Mans Hulden
(University of Helsinki)
Iñaki Alegria
(University of The Basque Country)

Overview

- Irregular forms
- Parallel “irregular” and regular forms
- Long-distance dependencies in lexc
- Agreement between separated morphemes

Errors in running example

```
foma[0]: source english.foma
...
foma[1]: down
apply down> make+V+PastPart
maked
apply down>
```

- Such irregularities are of course rampant, and we need to handle them
- Almost every language has suppletive forms for high-frequency verbs (be, was, were), adjectives (good, better, best), etc. and these need to be handled systematically

Adding exceptions in lexc

```
LEXICON Verb

fox   Vinf;
beg   Vinf;
make+V+PastPart:made  #;  !Bypass Vinf
make+V                #;
...
watch Vinf;
try   Vinf;
panic Vinf;
```

A separate "exceptions" grammar

```
foma[0]: define Exceptions [m a k e "+V" "+PastPart"]:[m a d e];
defined Exceptions: 370 bytes. 7 states, 6 arcs, 1 path.
foma[0]: regex Exceptions;
370 bytes. 7 states, 6 arcs, 1 path.
foma[1]: down
apply down> make+V+PastPart
made
apply down>
foma[1]:
```

Combining with priority union

Priority union operation for overriding regular with irregular forms

A

$$\begin{tikzcd}
a \arrow[d] & b \arrow[d] \\
b & c
\end{tikzcd}$$

B

$$\begin{tikzcd}
a \arrow[d] & c \arrow[d] \\
c & d
\end{tikzcd}$$

A .P. B

$$\begin{tikzcd}
a \arrow[d] & b \arrow[d] & c \arrow[d] \\
b & c & d
\end{tikzcd}$$

In this case, we need to calculate the transducer [Exceptions .P. Grammar]

Combining with priority union

```
foma[1]: define Grammar;
defined Grammar: 1.8 kB. 47 states, 70 arcs, 42 paths.
foma[0]: define Exceptions [m a k e "+V" "+PastPart"]:[m a d e];
defined Exceptions: 370 bytes. 7 states, 6 arcs, 1 path.
foma[0]: regex [Exceptions .P. Grammar];
1.9 kB. 52 states, 77 arcs, 42 paths.
foma[1]: down
apply down> make+V+PastPart
made
apply down>
foma[1]:
```

Alternate forms

- Alternate forms are also possible
- English: cactus+N+Pl→cactuses, cacti
- In this case we can create an alternate forms grammar, and calculate the union with the regular grammar

```
foma[0]: define ParallelForms [c a c t u s "+N" "+Pl"]:[c a c t i];
redefined ParallelForms: 410 bytes. 9 states, 8 arcs, 1 path.
foma[1]: regex ParallelForms | Grammar ;
...
```

Long-distance dependencies

- Overgeneration in the lexicon
- Constraining the co-occurrence of morphemes
- Strategy: compose a filter before or after lexical level
  - lexical information (tag sequence)
  - morphemes (morpheme sequence)
- Usually of the format ~$[PATTERN];
- “The language that does not contain PATTERN”
- Example from Basque: no double causatives (many agreement patterns work in similar manner)

Long-distance dependencies

Example of rule (for Basque)

```
# avoiding overgeneration from the lexicon
# causative prefix and suffix, but not both
#     RIGHT: bait+du, du+Elako
#     WRONG: bait+du+Elako
# LEXICAL LEVEL: [Kaus]+edun[V][P][3P]+[Kaus]
# INTERM. LEVEL:  bait+  du           +Elako
# SURFACE LEVEL:  bait   du            elako

# morphological inf. level
define NOTWO  ~$[ "[Kaus]" ?+ "[Kaus]" ];
define MORPHOFIL NOTWO .o. LEX .o. RULES ;
# intermediate level
define NOTWO2  ~$[ b a i t "+" ?+ "+" E l a k o];
define MORPHOFIL2 LEX .o. NOTWO2 .o. RULES;
```

Long-distance dependencies

```
foma[1]: regex MORPHO;
10.5 kB. 390 states, 589 arcs, Cyclic.
foma[2]: up baituelako
[Kaus]+edun[ADL][A1][P3]+[Kaus]
foma[2]: regex MORPHOFIL2;
11.6 kB. 450 states, 661 arcs, Cyclic.
foma[3]: up baituelako
???
foma[3]: up baitu
[Kaus]+edun[ADL][A1][P3]
foma[3]: up duelako
edun[ADL][A1][P3]+[Kaus]
```