Creating LRs and applications using finite-state morphological grammars HANDOUT

Bibliography and Tools

Bibliography

[1] Alegria I., Etxeberria I., Hulden H., Maritxalar M. 2009. Porting Basque Morphological Grammars to foma, an Open-Source Tool. FSMNLP2009. Pretoria. South Africa.
[2] Antworth, E.L. 1990. *PC-KIMMO: a two-level processor for morphological analysis*. Occasional Publications in Academic Computing, vol. 16.
[3] Beesley K. R. and Karttunen L. 2003. *Finite State Morphology*. CSLI Publications, Palo Alto, CA.
http://www.stanford.edu/~laurik/fsmbook/home.html
[4] Oflazer K. *Computational Morphology* (slides). Tutorial. FSMNLP2009.
http://fsmnlp2009.fastar.org/Program_files/Oflazer%20-%20slides.pdf

Slides and examples

http://foma.sf.net/lrec2010 (contains this handout, slides, complete example scripts)

Tools

foma: http://foma.sf.net/
hfst: http://www.ling.helsinki.fi/kieliteknologia/tutkimus/hfst/
Xerox tools: http://www.stanford.edu/~laurik/.book2software/

Additional links

Graphviz: http://www.graphviz.org (for automata visualization)
Graphviz for OSX: http://www.pixelglow.net (for automata visualization on Macs)
Beesley & Karttunen FSM book: http://www.fsmbook.com

Foma frequently used commands

apropos <keyword>	short help on keyword
help <keyword>	long help on keyword
regex regular-expression	compile regular expression
define name regular-expression	compile and name a regular expression
define name	name the top FSM on stack
clear/clear stack	clear stack
pop/pop stack	pop the top FSM off stack
down/apply down	enter apply down mode
down <word>	apply a single word
up/apply up	enter apply up mode
up <word>	apply a single word
med/apply med	enter minimum edit distance mode (automata only)
load stack <filename>	load binary FSM
load defined <filename>	load definitions from binary file
save defined <filename>	save definitions into binary file
save stack <filename>	save all the FSMs on the stack into binary file
source <filename>	compile a file of foma-commands
read lexc <filename>	compile a lexc file
print words/words	print all words in a FSM
print upper-words/upper-words	print all words on upper side
print lower-words/lower-words	print all words on lower side
print random-lower/random-lower	print a random selection of words (lower)
print random-upper/random-upper	print a random selection of words (upper)
print net/net	print information about top FSM on stack
view net/view	display top FSM visually

Foma regular expressions

Standard:

`A B`	Concatenation
`A | B`	Union
`A & B`	Intersection
`A*`	Kleene star
`A+`	Kleene plus
`$A`	"Contains" a string from A
`A-B`	Subtraction
`~A`	Complement of A
`A.r`	Reverse of A
`(A)`	Optionally A (same as `A | 0`)

Transducer-related:

`A:B`	Cross-product of A and B
`A .o. B`	Composition of A and B
`A.i`	Invert A
`A.u`	Extract upper side (domain) of A
`A.l`	Extract lower side (range) of A
`A .P. B`	Priority union of A and B

Rewrite operations:

`A -> B`	Rewrite strings in A as B
`A (->) B`	Optionally rewrite A as B
`A -> B || C _ D`	Conditional rewrite of A as B (between C and D)
`[..] -> B || C _ D`	Insert a single B between C and D
`A -> B , C -> D ,...`	Multiple simultaneous rewrites (w/ or w/o contexts)
`A -> B ... C`	Markup: insert B before and C after A (w/ or w/o contexts)

Special symbols:

`0 or []`	Epsilon (the empty string)
`?`	The "any" symbol
`.#.`	Word boundary in rewrite rules
`[ and ]`	Grouping symbols for forcing precedence
`" "`	Reserved symbols need to be escaped by quotes

Examples for English

english.lexc

```
Multichar_Symbols +N +V +PastPart +Past +PresPart +3P +Sg +Pl

LEXICON Root

Noun ;
Verb ;

LEXICON Noun

cat   Ninf;
city  Ninf;
watch Ninf;
try   Ninf;
panic Ninf;
fox   Ninf;

LEXICON Verb

fox   Vinf;
beg   Vinf;
make  Vinf;
watch Vinf;
try   Vinf;
panic Vinf;

LEXICON Ninf

+N+Sg:0   #;
+N+Pl:^s  #;

LEXICON Vinf
+V:0            #;
+V+3P+Sg:^s     #;
+V+Past:^ed     #;
+V+PastPart:^ed  #;
+V+PresPart:^ing #;
```

<u>english.foma</u>

```
# Vowels
define V [a | e | i | o | u ];

read lexc english.lexc
define Lexicon;

# Consonant doubling: 1-letter consonant doubled before
# -ing/-ed (beg/begging), we only handle g here
define ConsonantDoubling g -> g g || _ "^" [i n g | e d ];

# E deletion: silent e dropped before -ing and -ed (make/making)
define EDeletion e -> 0 || _ "^" [ i n g | e d ] ;

# E insertion e added after -s, -z, -x, -ch, -sh
# before s (watch/watches)
define EInsertion [..] -> e || s | z | x | c h | s h _ "^" s ;

# Y replacement: -y changes to -ie before -s, -i before -ed (try/tries)
define YReplacement y -> i e || _ "^" s    ,,
                    y -> i   || _ "^" e d  ;

# K insertion: verbs ending with vowel + -c add -k (panic/panicked)
define KInsertion [..] -> k || V c _ "^" [e d | i n g];

# Cleanup: remove morpheme boundaries
define Cleanup "^" -> 0;

regex Lexicon .o.
      ConsonantDoubling .o.
      EDeletion .o.
      EInsertion .o.
      YReplacement .o.
      KInsertion .o.
      Cleanup;
```

Examples for Basque

Simple rules:

```
#  phonology with r
#  epenthetical r (Q)
define Q0 Q -> 0 || Cons MM _ ;
       #ur+Qen:uren (R12)
define QR Q -> r ;
       # amA+Qen:amA+ren:amaren

define RandQ R2 .o. RR .o. Q0 .o. QR

define BAIT1 d  -> 0 ||  .#. b a i t MM _ ;
       # bait+da:baita
define BAIT2 g  -> k ||  .#. b a i t MM _ ;
       # bait+gara:bait+kara:baikara
define BAIT3 t -> 0 ||  .#. b a i _ MM [ Nasal | k ] ;
       # bait+naiz:bainaiz
       # bait+lezake:bailezake
       # bait+gara:bait+kara:baikara

define BAIT BAIT1 .o. BAIT2 .o. BAIT3 ;define MM "+" ;
## hard r (R)
define R2 R -> r r || _ MM (Q) Vowel ;
       # zakuR+a:zakurra
       # itziaR+Qen:itziarr+Qen:itziarren
define RR R -> r ;
       # ekaR+tzen:ekartzen
```

Long-distance dependencies

```
# avoiding overgeneration from the lexicon
# causal 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;
```

Competence errors

```
# Example of rule (for Basque)
# usual mistakes and dialectal phonological rules
# used in CALL (Computer Aided Lang. Learning)
# Sibilants
define Sibilant z | s | x ;
define H1 h (->) 0 ;
       # hoztu:oztu
define H2 [..] (->) h ||  [Vowel0 | .#.] _ Vowel0 ;
       # leihoa:lehioa
define Sib Sibilant (->) Sibilant ;
       # etxe:etze

define CompRules H1 .o. H2 .o. Sib ;
define ComPAnal MORPHO .o. CompRules ;
define ComPCorr MORPHO.l .o. CompRules ;
```

Competence errors in the lexicon

```
# For dialectal uses or idiosyncratic changes
# New entries in the lexicon (LEXPLUS)
#   LEXICAL LEVEL: +Etik
#   INTERM. LEVEL: +Etikan
+Etik:Etikan    # ablative case
...
ihardun:jardun  # old standard
...

define ENHANCED LEXPLUS .o. RULES .o. COMPET ;

define CORRECTOR MORPHO.i .o. ENHANCED ;
```