Finite-state morphology/phonology tutorial, UD, Jan 23, 2013

Mans Hulden (University of Helsinki)
mhulden@email.arizona.edu

Resources

- **The Finite-State Morphology Book [Draft version] (Beesley and Karttunen, 2003)**

https://victorio.uit.no/langtech/tags/DIVVUN-NO_1_0_RELEASE/gt/doc/book.pdf_1.pdf

- **Foma: program, documentation, etc.**

http://foma.googlecode.com

- **Morphological analysis tutorial (with foma)**

https://code.google.com/p/foma/wiki/MorphologicalAnalysisTutorial

- **Tutorial slides**

https://foma.googlecode.com/files/lrec2010_slides.tar.gz

Foma frequently used commands

Starting foma

foma	
foma -l <filename>	Start foma and execute script on startup

Interface commands

help X	Get help on X
apropos X	Short help on X
clear	Clear stack
define X Y ;	define X with regular expression Y **(Ex.: define V [a|e|i|o|u])**
down	Enter apply down mode (CTRL-D exits this mode)
net	Print top FSM with text

pop	Pop the top FSM off of stack
up	Enter apply up mode (CTRL-D exits this mode)
random-lower	Print random selection of output words from top FSM
random-upper	Print random selection of input words from top FSM
regex X ;	Compile regular expression X
test equivalent	Test top two FSMs for equality
tseq	Test top FSM for sequentiality
view	View top FSM visually
words	Print all words of top FSM (or subset if cyclic)

Regular expressions

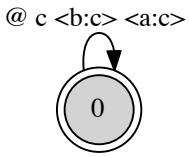
A | B	Union
A*	Kleene Star
A+	Kleene Plus
A B	Concatenation
?	Any symbol
0	Epsilon symbol
$A	"Contains" A (equiv. to ?* A ?*)
A - B	Subtraction
A & B	Intersection
~A	Complement of A
(A)	"Optionally"A: equivalent to A | 0
[ ]	Grouping brackets
A .o. B	Composition
A.r	Reversal
A -> B	Unconditional rewriting

A (->) B	Optional rewriting
A -> B || L _ R	Context-restricted rewriting (|| , \\\, //, \/) are all valid context specifiers
A -> B ... C	Insert B and C "around" instances of A (can take context specifiers)
[..] -> B	Epenthesis rules are specified like this (do not use 0 -> B)
.#.	Word-boundary symbol in context specifications
A.u	Extract input projection of transducer A
A.l	Extract output projection of transducer A
A < B	All instances of A precede B
A > B	All instances of A follow B
A => B _ C	Context restriction: the language where all instances of A occur only between C and D
A -> B, C -> D, ...	Multiple simultanous rewrites with same context
A -> B || C _ D ,, ...	Multiple simultanous rewrites with different contexts
"^"	Use quotes to escape special characters

Some examples covered

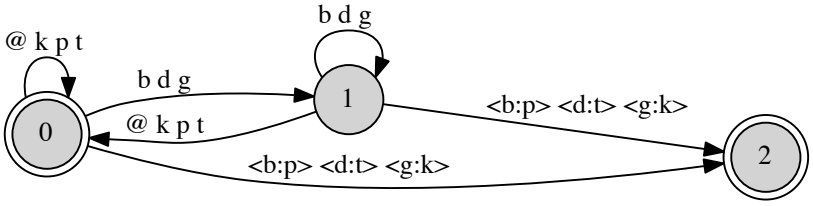
Basic composition

```
regex a -> b .o. b -> c ;
```



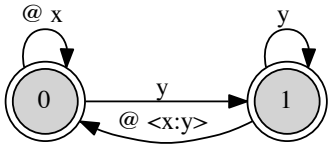
End of word stop devoicing:

```
regex b -> p , d -> t , g -> k || _ .#. ;
```



Spreading with // rules vs. regular ||-type rules:

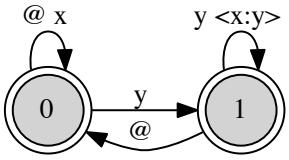
regex x -> y || y _ ;



apply down> **yxx**

yyx

regex x -> y // y _ ;



apply down> **yxx**

yyy

Illustration of differences between rule types (||, //, \\, \/)

a b -> x || a b _ a *[L holds on input side, R on input side]*

a b a b a b a (INPUT)

a b x x a (OUTPUT)

a b -> x \\ a b _ a *[L holds on input side, R on output side]*

a b a b a b a (INPUT)

a b a b x a (OUTPUT)

a b -> x // a b _ a *[L holds on output side, R on input side]*

a b a b a b a (INPUT)

a b x a b a (OUTPUT)

a b -> x \/ a b _ a *[L holds on output side, R on output side, gives two valid pairings]*

a b a b a b a a b a b a b a (INPUT)

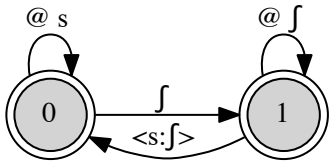
a b x a b a a b a b x a (OUTPUT)

Sibilant harmony example

```
#We condition the sibilant harmony on the sibilant immediately to
#the left, with possibly intervening nonsibilants, encoded by
#[?-SIB]*

define SIB [s|ʃ];

define Rule1 s -> ʃ || ʃ [?-SIB]* _ ;

define Rule2 s -> ʃ // ʃ [?-SIB]* _ ;
```

- With Rule1, we get no spreading:

```
foma[0]: regex Rule1;

423 bytes. 2 states, 6 arcs, Cyclic.

foma[1]: down

apply down> ʃisisi

ʃiʃisi

apply down>
```



- With Rule2, spreading occurs, since we condition on output side

```
foma[0]: regex Rule2;

423 bytes. 2 states, 6 arcs, Cyclic.

foma[1]: down

apply down> ʃisisi

ʃiʃiʃi
```