3. Finite State Morphology

CS674
Spring 2007

Verb Roots and Inflections

```
define Root [ {eat} | {file} | {swallow} ];
define Inflection   [ [%+ {ing} %+ VG] |
   [ %+ {+ed+} %+ VBD] |
   [ %+ s %+ VBZ] ];
```

Concatenate root and inflection

```
define VerbUpper Root Inflection;

read regex VerbUpper;
print words
file+ing+VG             file+ed+VBD
file+s+VBZ
eat+ing+VG              eat+ed+VBD
eat+s+VBZ
swallow+ing+VG          swallow+ed+VBD
swallow+s+VBZ
```

Deletion of e in file+ing

```
define eElision e -> 0 || _ %+ [e | i] ;

define Tag [ VG | VBD | VBZ ];
define symbolElision [ Tag | %+ ] -> 0 ;
```

Relation Composition

$||R .o. S|| =$

$\{\langle x,z\rangle \mid$ for some y,
$\langle x,y\rangle \in ||R||$ and
$\langle y,z\rangle \in ||S|| \}$

Composed verb lexicon

```
define verb
VerbUpper .o.   (set coerced to relation)
  eElision .o.     (delete e in file+ed+VBD)
  symbolElision ;
```

Result is relation between underlying and surface forms.

Verb Relation

read regex verb;
print words
swallow<+:0>s<+:0><VBZ:0>
swallow<+:0>ing<+:0><VG:0>
swallow<+:0>ed<+:0><VBD:0>
(six more)

Lower words

read regex verb.l;
print words
swallows swallowing swallowed
filing files filed
eats eating eated

read regex verb;
apply up
apply up> eated
eat+ed+VBD

List irregular verbs

define irregularVerb [e a t %+ e d %+ VBD]
.x. [a t e];

Cartesian product of two unit sets gives unit set of a pair.

Union is wrong

read regex verb | irregularVerb;
apply down
apply down> eat+ed+VBD
eated
ate
(Or do we want two outputs?)

define verb2 [[~irregularVerb.u] .o. verb] | irregularVerb ;
read regex verb2;
print lower-words;
swallows swallowing swallowed
filing filed files
eats eating ate

verb.fst

```
define Root [ {eat} | {file} | {swallow} ];
define Inflection   [ [%+ {ing} %+ VG] |
    [ %+ {ed} %+ VBD] |
    [ %+ s %+ VBZ] ];

define VerbUpper Root Inflection;
define Tag [ VG | VBD | VBZ ];

define eElision e -> 0 || _ %+ [e | i] ;

define symbolElision [ Tag | %+ ] -> 0 ;

define verb [VerbUpper .o. eElision .o. symbolElision ] ;

define irregularVerb [e a t %+ e d %+ VBD] .x. [a t e];

define verb2 [[~irregularVerb.u] .o. verb ] | irregularVerb ;
```

To use it in the interpreter…

[xfst 5] source verb
[xfst 6] read regex verb2;
[xfst 7] print lower-words