

English morphology

Mans Hulden

(University of Helsinki)

Iñaki Alegria

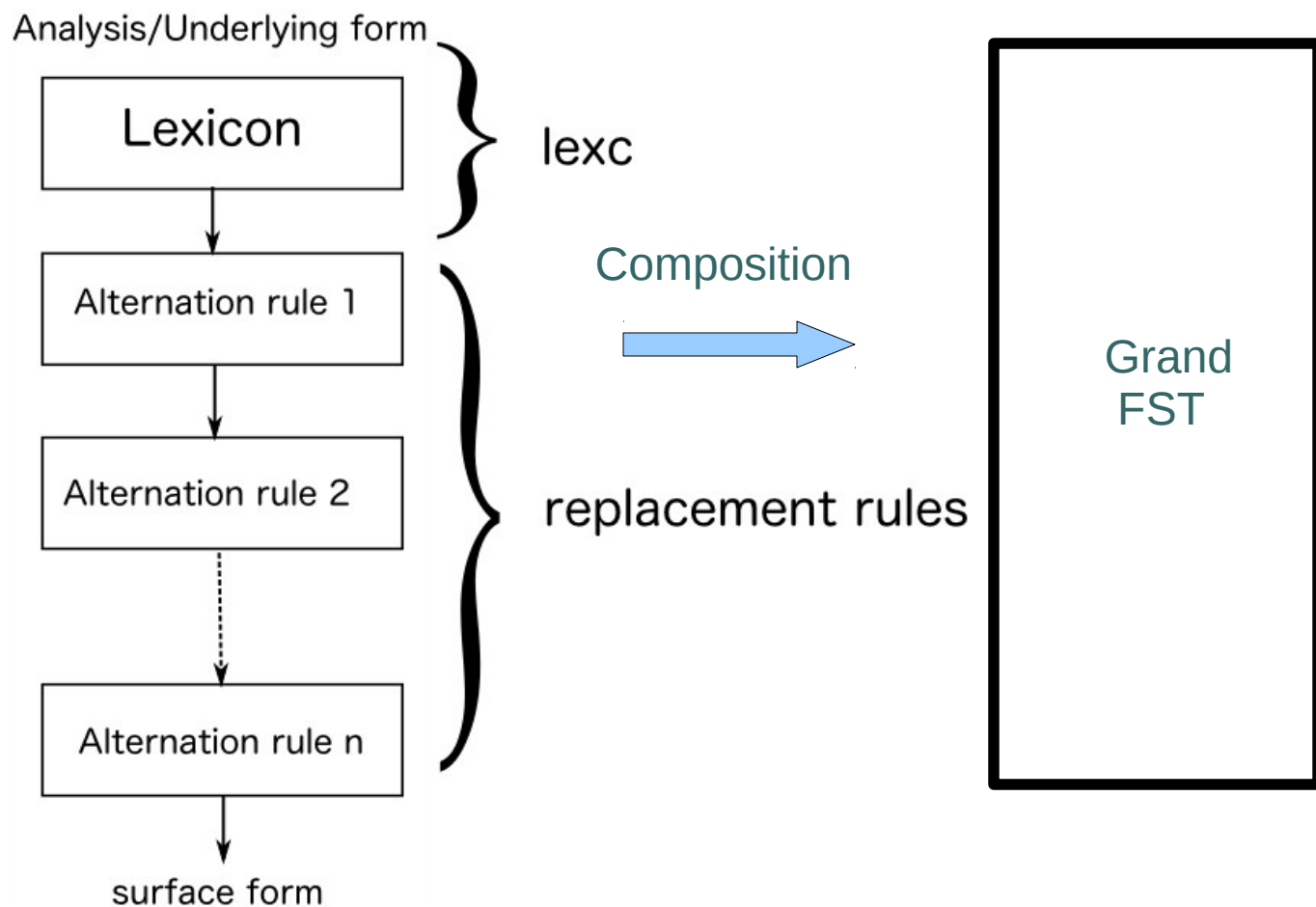
(University of The Basque Country)



Overview

- Designing a rewrite-grammar
- Composing the lexicon with the rules
- Compiling & testing a complete grammar
- A few examples

The Big Picture (again)



Running English example

- We created a lexc-grammar that takes us from analyses to intermediate forms:

```
c i t y +N +Pl  
c i t y ^ s
```

- The task now is to create the replacement rule transducers to be composed with the lexc-transducer, yielding correct surface forms:

```
c i t y +N +Pl (lexc upper)  
c i t y ^ s (lexc lower)  
c i t i e ^ s (after y -> i e rule)  
...  
c i t i e s (after nth rule)
```



The facts to be modeled

(1) E-deletion: silent e dropped before -ing and -ed
(make/making)

```
m a k e +V +PresPart      (lexc upper)
m a k e ^ i n g           (lexc lower)
...
m a k 0 ^ i n g           (after E-deletion)
...
```

The rule can be defined as:

```
define EDeletion e -> 0 || _ "^" [ i n g | e d ] ;
```

Let's test the rule *separately [in foma]*:

The facts to be modeled

(2) K-insertion: verbs ending with vowel-c add -k at end of stem when succeeded by -ed/-ing

p a n i c +V +PresPart (lexc upper)

p a n i c ^ i n g (lexc lower)

...

p a n i c **k** ^ i n g (after K-insertion)

...

The rule can be defined as:

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

```
define KInsertion [..] -> k || V c _ "^" [e d|i n g];
```

The facts to be modeled

(3) E-insertion:

f o x +N +Pl (lexc upper)

f o x ^ s (lexc lower)

...

f o x e ^ s (after E-insertion)

...

The rule can be defined as:

```
define EInsertion [..] -> e || [s|z|x|c h|s h] _ "^" s;
```

*This is not foolproof: consider arch → arches vs. monarch → monarchs



The facts to be modeled

(4) Consonant doubling: 1-letter consonant doubled before -ing/-ed

b e g +V +PresPart (lexc upper)

b e g ^ i n g (lexc lower)

...

b e g **g** ^ i n g (after C-doubling)

...

The rule can be defined (for g) as:

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

```
define ConsonantDoubling g -> g g || V _ "^" i n g ;
```


The facts to be modeled

(5) Y-replacement: y changes to ie before -s, and i before -ed

t r y	+N	+Pl	(lexc upper)
t r y	^	s	(lexc lower)
...			
t r i e	^	s	(after Y-replacement)
...			

The rule can be defined as:

```
define YReplacement y -> i e || _ "^" s , ,  
                    y -> i   || _ "^" e d ;
```



The facts to be modeled

(6) After we're done with the alternations, we remove the boundary markers:

t r y	+N	+Pl	(lexc upper)
t r y	^	s	(lexc lower)
...			
t r i e		s	(after Cleanup)
...			

The rule can be defined as:

```
define Cleanup "^" -> 0;
```



Putting the grammar together

...

```
read lexc english.lexc  
define Lexicon;
```

```
Define Rules ConsonantDoubling .o. EDeletion .o.  
EInsertion .o. YReplacement .o. KInsertion .o.  
Cleanup;
```

```
regex Lexicon .o. Rules;
```



Compiling

```
foma[0]: source english.foma
Opening file 'english.foma'.
defined V: 317 bytes. 2 states, 5 arcs, 5 paths.
Root...2, Noun...6, Verb...6, Ninf...2, Vinf...5
Building lexicon...Determinizing...Minimizing...Done!
1.3 kB. 32 states, 46 arcs, 42 paths.
defined Lexicon: 1.3 kB. 32 states, 46 arcs, 42 paths.
defined ConsonantDoubling: 1.0 kB. 11 states, 47 arcs,
Cyclic.
defined EDeletion: 1.1 kB. 11 states, 52 arcs, Cyclic.
defined EInsertion: 1000 bytes. 7 states, 43 arcs, Cyclic.
defined YReplacement: 874 bytes. 9 states, 36 arcs, Cyclic.
defined KInsertion: 1.2 kB. 11 states, 59 arcs, Cyclic.
defined Cleanup: 260 bytes. 1 states, 2 arcs, Cyclic.
1.8 kB. 47 states, 70 arcs, 42 paths.
foma[1]:
```

Let's test the grammar!



Testing...debugging...

foma[1]: lower-words

cat

cats

city

cities

panic

panics

panic

panics

panicking

panicked

panicked

...

make

makes

making

maked

beg

Is an exception and we will postpone its treatment for a minute...





Review of lexc+rules

General strategy:

- Create lexc-grammar, load in foma, define:

```
read lexc english.lexc  
define Lexicon;
```

- Replacement rules in foma:

```
define Rule1 x -> y ...
```

- Combine with composition:

```
define Grammar Lexicon .o. Rule1 .o. ... .o. RuleN;  
regex Grammar;
```



Points...

Rule order

Simple Spanish (pluralization) rules

examples: papel+s:papeles; pez+s:peces

sequential (ordered)

(1) z -> c || _ "+" s .#. ;

(2) [..] -> e || Cons "+" _ s .#. ;

Compare:

pez+s (1) → pec+s (2) → pec+es

pez+s (2) → pez+es (1) → *pez+es



Points...

Simple Basque Rules

```
# phonology with r
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

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

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