Carnegie Mellon University
Language
Technologies
Institute

11-324/11-624/11-724 Human Language for AI

The XFST Formalism and Practicing Morphophonology

David R. Mortensen

September 27, 2022

Language Technologies Institute
Carnegie Mellon University

Learning Objectives

Students will leave this lecture knowing the following things:

- The difference between phonemic analysis and morphophonological analysis (reinforced)
- What XFST is and how it relates to *Foma*
- The basic notation for regular expressions in XFST
- The basic notation for rewrite rules in XFST

Students will acquire the following skills:

- Writing phonological rules using the XFST formalism
- Combining such rules in ordered cascades using transducer composition
- Using Foma to complete a basic morphophonological analysis
- Recognizing phonological generalizations using an IPA chart
- Distinguishing a more economical analysis from a less economical analysis

FST/Foma Rule Schema

```
A -> B || L _ R
```

- “A” source
- “B” target
- “L” left context
- “R” right context
- “_” locus (where the substitution occurs)

An Example Rule

Consider the following rule:

```
z -> s || [p | t | k | f | θ | s | ʃ] _ .#.
```

“z is rewritten as s between a voiceless consonant (p, t, k, f, θ, s, or ʃ) and a word boundary”

- `[` and `]` (square brackets) indicate grouping
- `|` (pipe) indicates automaton union
- `.#.` indicates word boundary (matches at the beginning or the end of a string)

XFST Rule-Like Notations

- Simple replacement in all contexts (regular expression):

  ```
  [a -> b]
  ```

- Read rule onto the stack:

  ```
  read regex a -> b ;
  ```

- Define `Rule1` as a replacement rule `a -> b`:

  ```
  define Rule1 a -> b ;
  ```

XFST Regular Expressions

- `.o.` indicates concatenation
- `|` indicates union
- `[ A | B ]` indicates the union of `A` and `B`
- `0` indicates the empty string ϵ (but not on the left side of rules)
- `[..]` indicates the empty string ϵ (on the left side of rules only)
- `a:b` represents a ordered pair of strings (UPPER and LOWER)
- `?` matches any character
- `.#.` is the boundary symbol, indicating the beginning of a string and the end of a string.
- `(A)` represents `[A | 0]` (optional `A`)
- `A+` indicates the concatenation of A with itself one of more times
- `A*` denotes `[A+ | 0]` (A repeated zero or more times)
- ~`A` matches the set of all strings that are not in `A`
- `\A` matches the set of all characters that are not in `A`

English -z Allomorphy

Singular	Phonemic	Plural	Phonemic
dog	/dɑg/	dogs	/dɑg-z/
cat	/kæt/	cats	/kæt-s/
horse	/hɔɹs/	horses	/hɔɹs-əz/
take	/tejk/	takes	/tejks/
give	/gɪv/	gives	/gɪvz/
watch	/wɑt͡ʃ/	watches	/wɑt͡ʃəz/

Rules in Context

```
define Sibilant s | z | ʃ | ʒ | t ͡ʃ | d ͡ʒ;
define Voiceless p | t | k | f | θ | s | ʃ;
define Epenthesis [..] -> ə || Sibilant _ Sibilant;
define Devoicing z -> s || Voiceless _ .#.;
read regex Epenthesis .o. Devoicing;
```

- `define` declares a transducer/regular expression; these statements are terminated by a semicolon
- Symbols are delimited by whitespace; the string “t͡ʃ” is entered as “t ͡ʃ” so that it will be treated as three symbols rather than one
- `[..]` indicates ϵ, the empty string, in insertion rules
- `read regex` adds a transducer/regular expression to the stack; these statements are also terminated by a semicolon
- `.o.` composes two transducers

Catalan Example I

MASC SG	FEM SG		MASC SG	FEM SG	
əkel^{j}	əkel^{j}ə	‘that’	mal	malə	‘bad’
siβil	siβilə	‘civil’	əskerp	əskerpə	‘shy’
ʃop	ʃopə	‘drenched’	sɛk	sɛkə	‘dry’
əspɛs	əspɛsə	‘thick’	gros	grosə	‘large’
baʃ	baʃə	‘short’	koʃ	koʃə	‘lame’
tot	totə	‘all’	brut	brutə	‘dirty’
pɔk	pɔkə	‘little’	prəsis	prəsizə	‘precise’
frənses	frənsezə	‘French’	gris	grizə	‘grey’
kəzat	kəzaðə	‘married’	bwit	bwiðə	‘empty’
rɔt͡ʃ	rɔʒə	‘red’	bot͡ʃ	boʒə	‘crazy’
orp	orβə	‘blind’	l^{j}ark	l^{j}arɣə	‘long’
sek	seɣə	‘blind’	fəʃuk	fəʃuɣə	‘heavy’
grok	groɣə	‘yellow’	puruk	puruɣə	‘fearful’
kandit	kandiðə	‘candid’	frɛt	frɛðə	‘cold’

Catalan Example II

MASC SG	FEM SG		MASC SG	FEM SG	
səɣu	səɣurə	‘sure’	du	durə	‘hard’
səɣəðo	səɣəðorə	‘reaper’	kla	klarə	‘clear’
nu	nuə	‘nude’	kru	kruə	‘raw’
flɔɲd͡ʒu	flɔɲd͡ʒə	‘soft’	dropu	dropə	‘lazy’
əgzaktə	əgzaktə	‘exact’	əlβi	əlβinə	‘albino’
sa	sanə	‘healthy’	pla	planə	‘level’
bo	bonə	‘good’	sərɛ	sərɛnə	‘calm’
suβlim	suβlimə	‘sublime’	al	altə	‘tall’
fɔr	fɔrtə	‘strong’	kur	kurtə	‘short’
sor	sorðə	‘deaf’	bɛr	bɛrðə	‘green’
san	santə	‘saint’	kəlɛn	kəlɛntə	‘hot’
prufun	prufundə	‘deep’	fəkun	fəkundə	‘fertile’
dəsen	dəsentə	‘decent’	dulen	dulentə	‘bad’
əstuðian	əstuðiantə	‘student’	blaŋ	blaŋkə	‘white’

Catalan Classes

```
define Vowel a | e | i | o | u ;
define SonCons m | n | ŋ | l | r ;
define Sonorant Vowel | SonCons ;
```

Devoicing and Spirantization

```
define Devoicing b -> p,
                 d -> t,
                 g -> k,
                 z -> s,
                 d ͡ʒ -> t ͡ʃ || _ .#. ;
define Spirantization b -> β,
                      d -> ð,
                      g -> ɣ,
                      d ͡ʒ -> ʒ || Sonorant _ ;
read regex Devoicing .o. Spirantization ;
```

Deletion Rules

```
define Apocope Vowel -> 0 || _ Vowel ;
define DDeletion D -> 0 ;
define SonorantDeletion [ n | r ] -> 0 || _ .#. ;
define ClusterSimplification [ t | d | k ] -> 0 || SonCons _ .#. ;
```

Full Catalan

```
define Vowel a | e | i | o | u ;
define SonCons m | n | ŋ | l | r ;
define Sonorant Vowel | SonCons ;
define Apocope Vowel -> 0 || _ Vowel ;
define DDeletion D -> 0 ;
define SonorantDeletion [ n | r ] -> 0 || _ .#. ;
define Devoicing b -> p,
                 d -> t,
                 g -> k,
                 z -> s,
                 d ͡ʒ -> t ͡ʃ || _ .#. ;
define ClusterSimplification [ t | k ] -> 0 || SonCons _ .#. ;
define Spirantization b -> β,
                      d -> ð,
                      g -> ɣ,
                      d ͡ʒ -> ʒ || Sonorant _ ;
read regex Apocope .o. DDeletion .o. SonorantDeletion .o.
           Devoicing .o. ClusterSimplification  .o. Spirantization ;
```

Advertising: LLab

- Fridays, 4:00–5:00pm
- GHC 6721

LINGUISTICS
STRUGGLE FOR STRUCTURE
FRI 11AM — GHC 6708 — DMORTENS@CS.CMU.EDU
HTTP://WWW.CS.CMU.EDU/AFS/CS/PROJECT/LLAB/WWW/
LAB