Finite State Morphological Parsing

April 6, 2011

Overview

- Review: Finite state methods in morphology
- Ambiguity
- XFST demo
- FSTs for spelling change rule
- Lexicon-free morphology
- Detection and correction of spelling errors

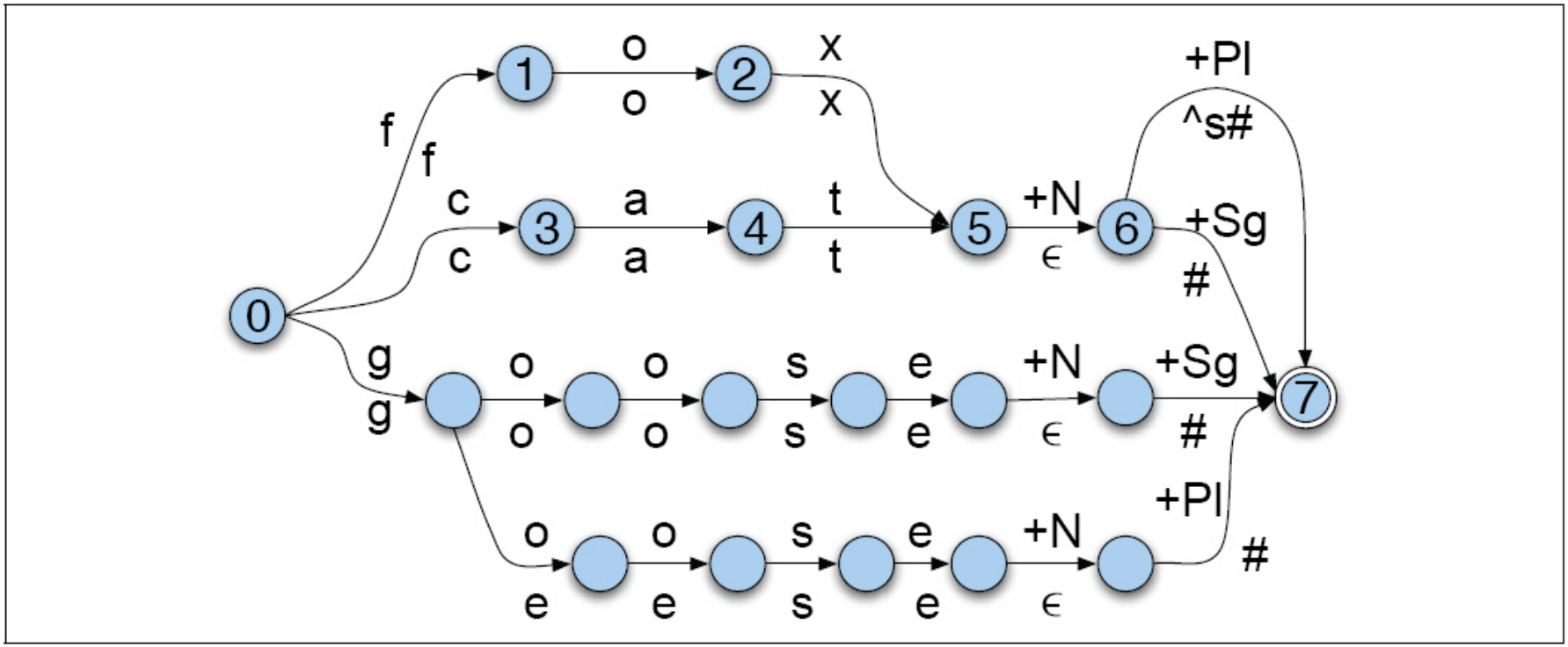
Review: FSAs and FSTs

- FSAs define sets of strings (regular languages).
- FSTs define sets of ordered pairs of strings (regular relations).
- Formally interesting because not all languages/ relations can be defined by FSAs/FSTs.
- Are all finite languages and relations regular?
- Linguistically interesting because:
  - FSAs have enough power for morphotactics.
  - FSTs have (almost) enough power for morphophonology.
  - Both are very efficient.

FSTs: Quiz

- Why do FSTs have complex symbols labeling the arcs?
- What happens if you give an FST an input on only one “tape”?
- What happens if the input has symbols outside the FST’s alphabet?
- Do the upper and lower tape strings always have the same length?

Recall this FST

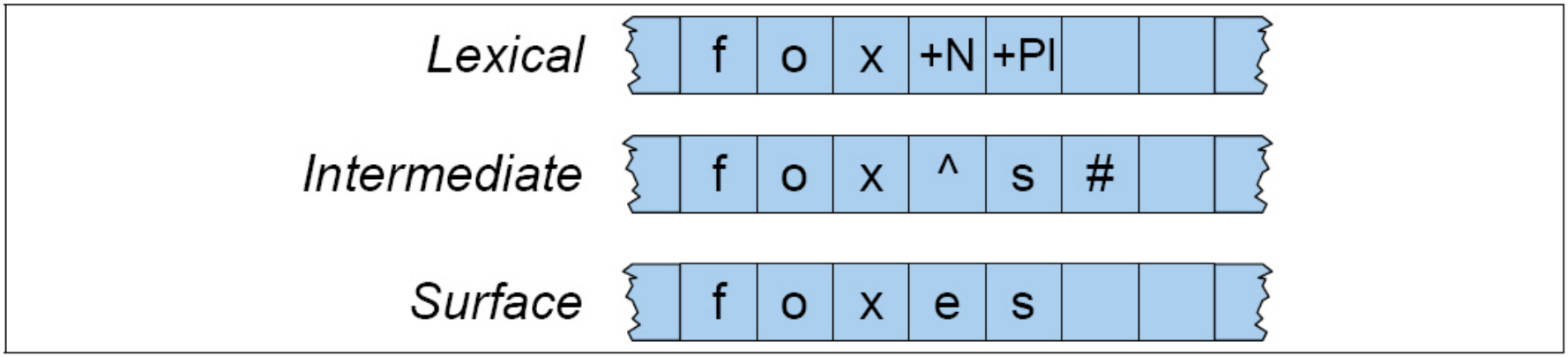


J&M text, Fig. 3.14

Cascade with an FST to handle spelling

- A spelling change rule would insert an e only in the appropriate environment:

$$\epsilon \rightarrow e \;/\; \{x,s,z\}\hat{}\; \_\_\_ s\#$$

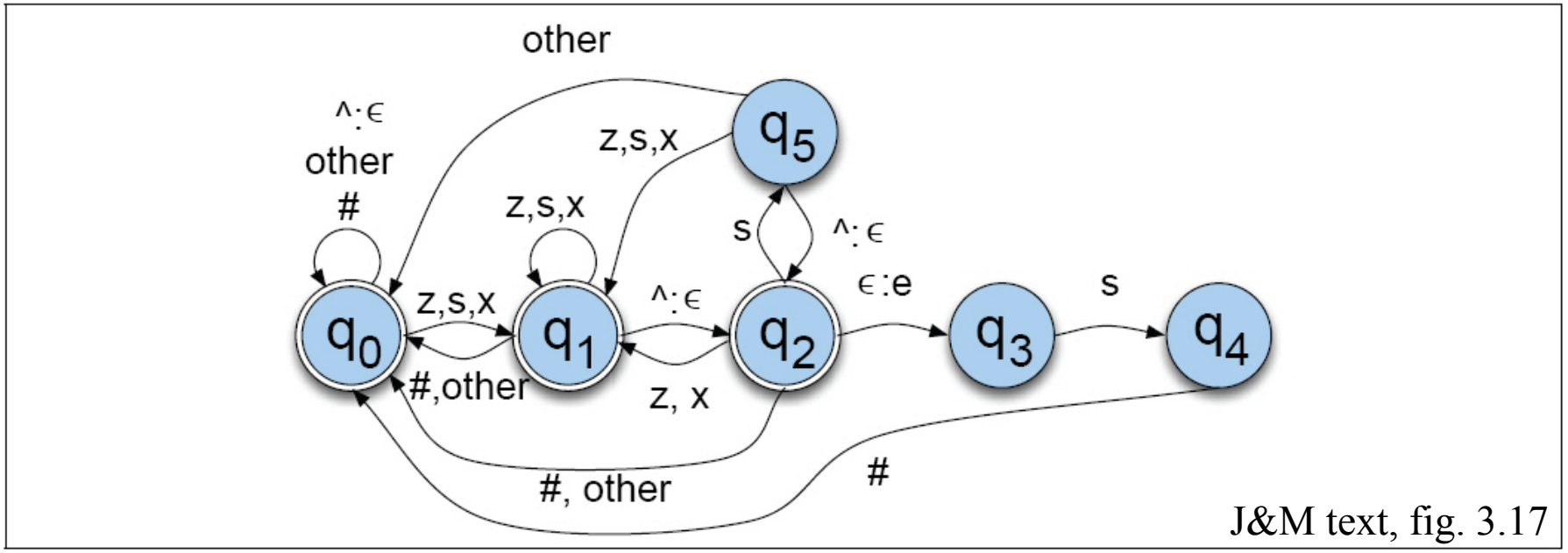


J&M text, Fig 3.16

- Note that you can read down from the top tape or up from the bottom tape.

Sample e-insertion FST

The idea is to add *e* only in the proper environments while letting all other sequences pass through.



J&M text, fig. 3.17

- But it is not necessary to hand-write FSTs like this.
- Many tools are available that compile FSTs from rules.

A Few Words about Ambiguity

- Ambiguity can be an issue in parsing;
- Example: *foxes*
  - *fox* can be a noun or a verb
  - the affix *s* can mark plural or 3rd sg present tense
- This kind of ambiguity (global) cannot be resolved with a transducer.
- However, transducer design must handle local ambiguity such as whether the *e* in the string *asse* is an inserted *e* (*asses*) or part of a stem (*assess*).

Xerox Finite-State Tool (`xfst`)

- Karttunen, Gaál & Kempe, 1997
- http://www.cis.upenn.edu/~cis639/docs/xfst.html
- Abstract: "Xerox finite-state tool is a general-purpose utility for computing with finite-state networks. It enables the user to create simple automata and transducers from text and binary files, regular expressions and other networks by a variety of operations. The user can display, examine and modify the structure and the content of the networks. The result can be saved as text or binary files."

XFST syntax

`*` = Kleene *

`+` = Kleene +

`0` = epsilon (empty string)

`%` = escape character

␣ (space) = concatenation

`\` = negation

`|` = disjunction

`(` `)` = optionality

`?` = wild card

`[` `]` = grouping

A note on ?

- In regular expressions, it's ANY.
- In arc labels, it's UNKNOWN … any symbol not otherwise represented in the FST.
- `xfst` takes a regular expression and returns an FST so note that ? means something slightly different character in each.

XFST demo

- Concatenation
- Kleene *, Kleene +
- Symbol pairs (‘:’)
- Iteration
- Wildcard
- +-removal
- Composition
- Apply up, apply down
- Print upper, print lower
- Print net

Spelling change rule FST 1

```
define Rule1 [ ?* e:0 %+:0 [e|i] ?*];
```

- Draw an FST corresponding to Rule1.
- What are the upper and lower languages of Rule1?
- What linguistic work is this rule supposed to do?
- If the upper tape has `expect+ed`, what goes on the lower tape?

Spelling change rule FST 2

```
define Rule2 [[?* e:0 %+:0 [e|i] ?*] |
              [?* e %+:0 (\[e|i])] |
              [?* \e %+ ?*] |
              [\[%+]*]]
```

- What are the upper and lower languages of Rule2?
- What linguistic work is each part of this rule supposed to do?
- If the upper tape has expect+ed, what goes on the lower tape?
- If the upper tape has write+ing, what goes on the lower tape?

What if you don't have a lexicon?

- Why might you not have a (big enough) lexicon?
- Why might you still want to do morphological parsing?
- The Porter stemmer is a cascade of rewrite affixation rules sensitive to orthographic properties of words, but without knowledge of any particular lexicon.
- Robust systems combine lexicon-based morphological parsing with techniques for handling unknown words. E.g., Chasen – morpological parser of Japanese text.

Detection and correction of spelling errors

- Integral part of many word processors and search engines
- Important for correcting errors in OCR and handwriting recognition
- Three problems (in order of difficulty):
  - Non-word error detection
  - Isolated-word error correction
  - Context-dependent error detection and correction (including real-word errors)

FSAs as spell-check dictionaries

- Non-word error *detection* is usually based on a large dictionary.
- An FST morphological parser is inherently a word recognizer.
- An FSA recognizer can be made by projecting the lower tape from an FST morphological parser.
- Non-word error correction algorithms use some form of distance metric to select between possible word candidates.

Overview

- Review: Finite state methods in morphology
- Ambiguity
- XFST demo
- FSTs for spelling change rule
- Lexicon-free morphology
- Detection and correction of spelling errors