Two Level Morphology

Alexander Fraser & Liane Guillou
{fraser,liane}@cis.uni-muenchen.de

CIS, Ludwig-Maximilians-Universität München

Computational Morphology and Electronic Dictionaries
SoSe 2016
2016-05-09

Outline

- Today we will briefly discuss two-level morphology
- Then Luisa will present an exercise showing how to use these concepts

Credits

- Adapted from a lecture by Ching-Long Yeh, Tatung University
- Which was adapted from:
- Chapter 3 Morphology and Finite-State Transducers
- *Speech and Language Processing*
- *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*
- Daniel Jurafsky and James H. Martin

Two-Level Morphology

- Two-level morphology is a key idea for dealing with morphology in a finite state framework
- The critical generalization is that it is difficult to deal with things like orthographic rules in English with a single transducer
- The key to making this work will be to use **two transducers**
- Recall that we can **compose** transducers
  - Composing intuitively means we feed the output of the first transducer as the input to the second transducer

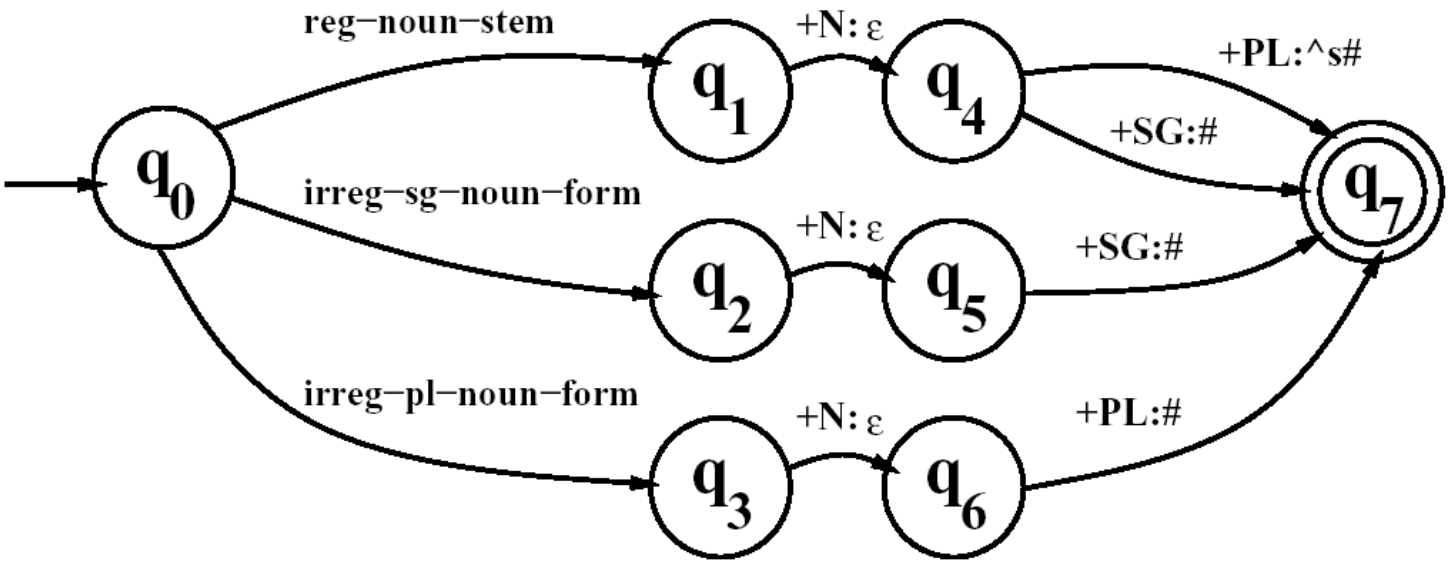
3.2 Finite-State Morphological Parsing

Morphological Parsing with FST

- Composition is useful because it allows us to take two transducers than run in series and replace them with one complex transducer.
  - $T_1 \circ T_2(S) = T_2(T_1(S))$

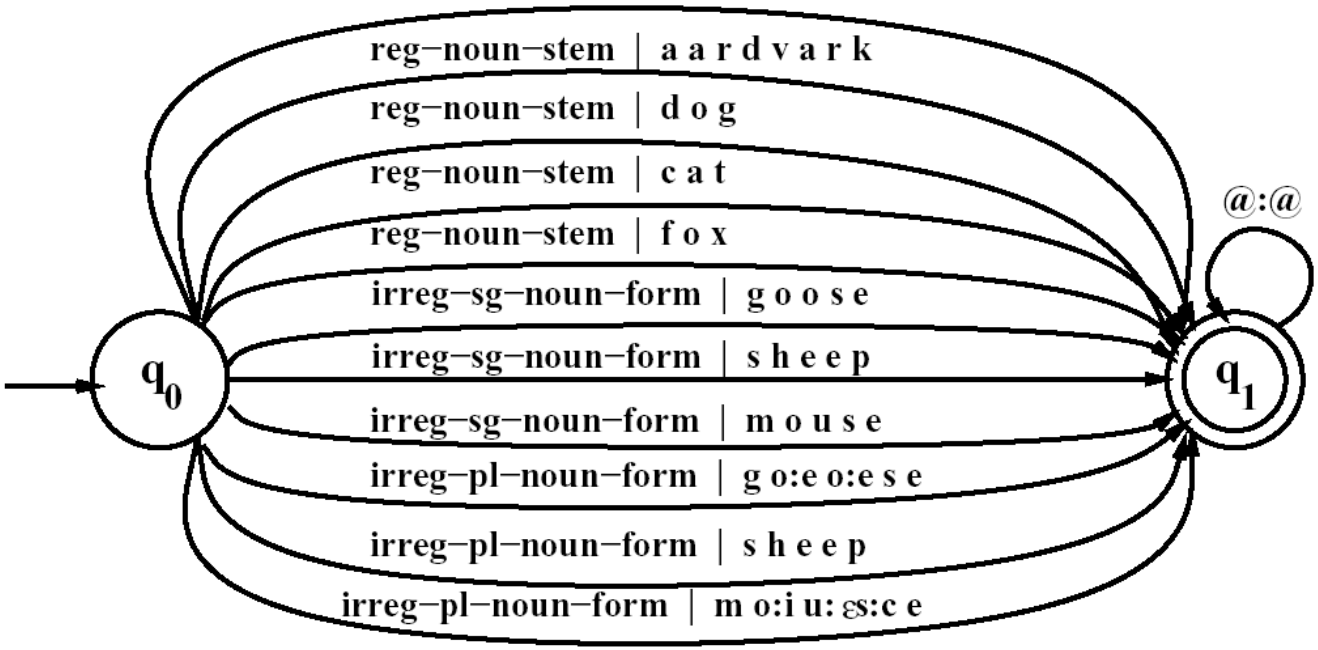
Reg-noun	Irreg-pl-noun	Irreg-sg-noun
fox	g o:e o:e s e	goose
cat	sheep	sheep
fog	m o:i u:ɛs:c e	mouse
aardvark		

A transducer for English nominal number inflection T_{num}



3.2 Finite-State Morphological Parsing

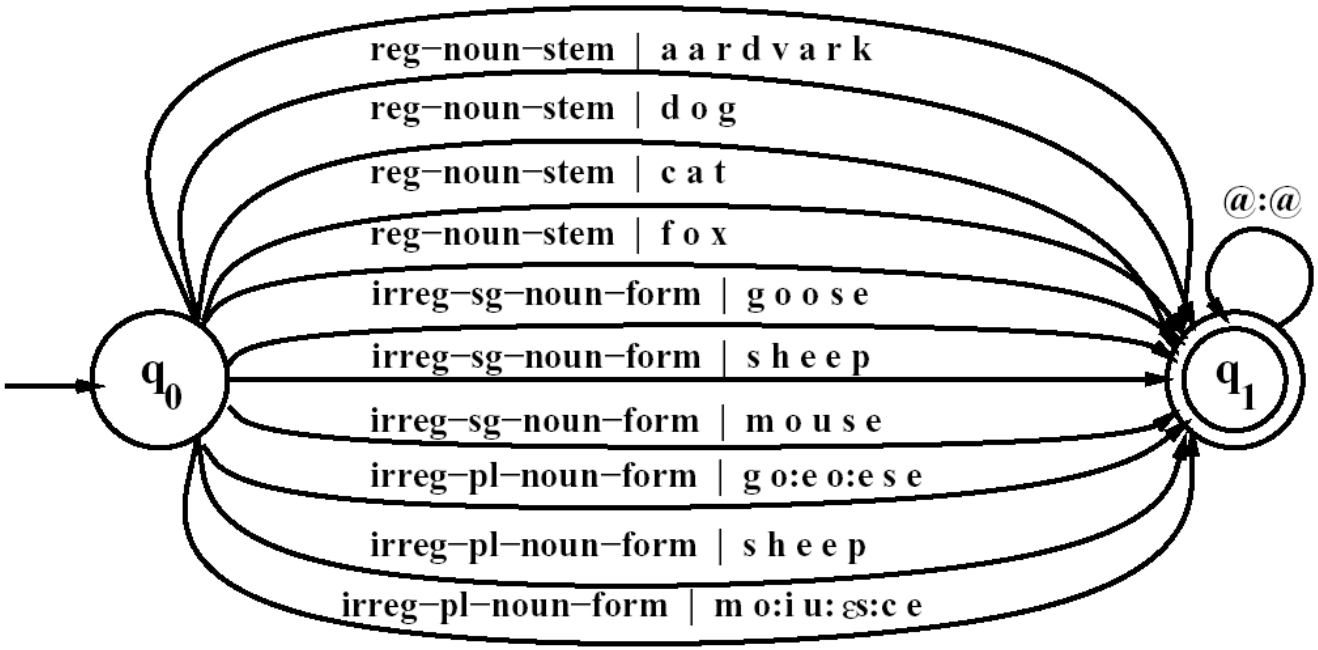
Morphological Parsing with FST



The transducer T_{stems}, which maps roots to their root-class

3.2 Finite-State Morphological Parsing

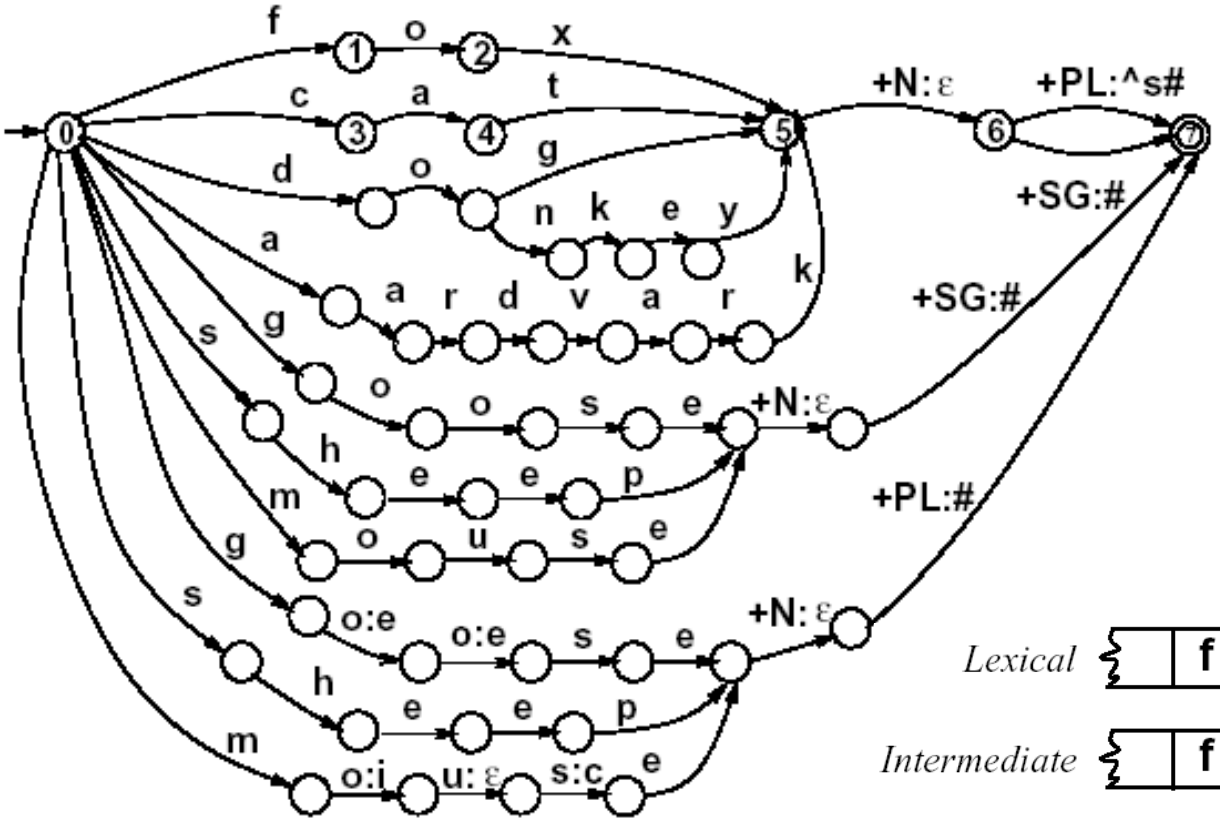
Morphological Parsing with FST



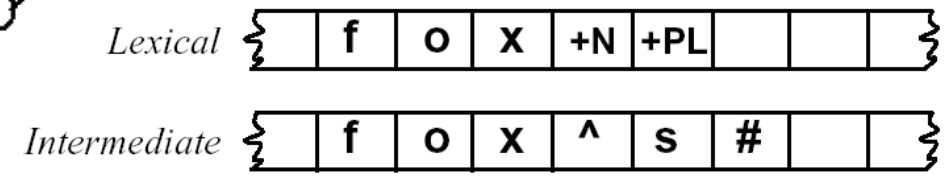
The transducer T_{stems}, which maps roots to their root-class

3.2 Finite-State Morphological Parsing

Morphological Parsing with FST



^: morpheme boundary
#: word boundary



A fleshed-out English nominal inflection FST

$T_{lex} = T_{num} \circ T_{stems}$

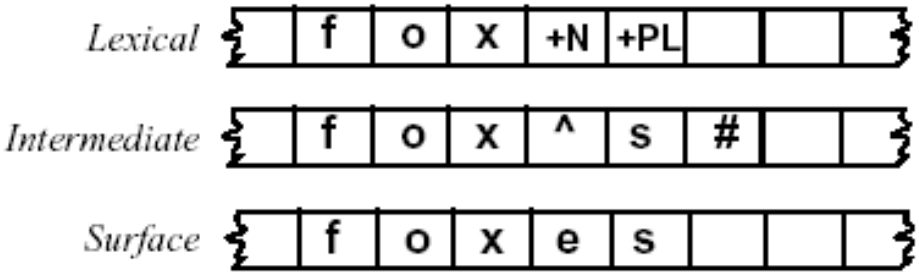
3.2 Finite-State Morphological Parsing

Orthographic Rules and FSTs

- **Spelling rules** (or **orthographic rules**)

Name	**Description of Rule**	**Example**
Consonant doubling	1-letter consonant doubled before *-ing/-ed*	beg/begging
E deletion	Silent e dropped before *-ing* and *-ed*	make/making
E insertion	**e added after *-s, -z, -x, -ch, -sh,* before *-s***	**watch/watches**
Y replacement	*-y* changes to *-ie* before *-s*, *-i* before *-ed*	try/tries
K insertion	Verb ending with *vowel* + *-c* add *-k*	panic/panicked

– These spelling changes can be thought as taking as input a simple concatenation of morphemes and producing as output a slightly-modified concatenation of morphemes.



3.2 Finite-State Morphological Parsing

Orthographic Rules and FSTs

- "insert an *e* on the surface tape just when the lexical tape has a morpheme ending in *x* (or *z*, etc) and the next morphemes is –*s*"

$$\varepsilon \rightarrow e / \left\{ \begin{matrix} x \\ s \\ z \end{matrix} \right\} \hat{}\_\_ s\#$$

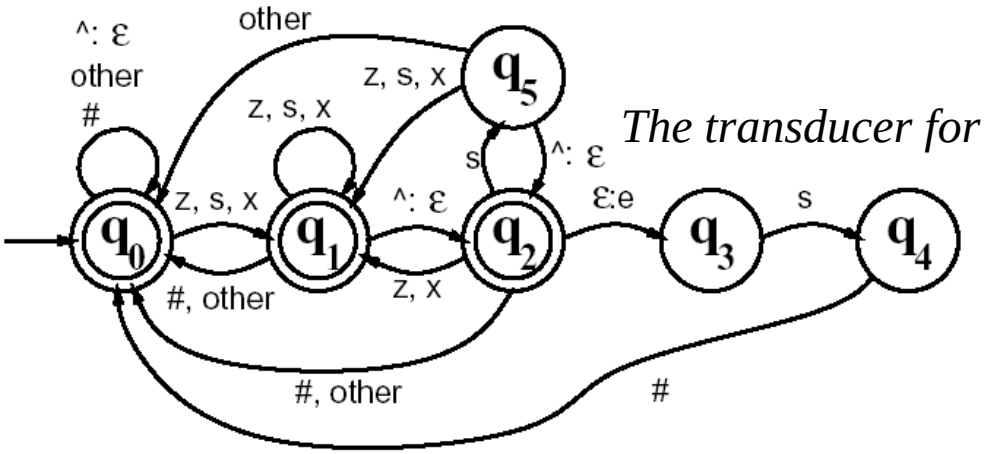
- "rewrite *a* as *b* when it occurs between *c* and *d*"

$$a \rightarrow b / c \_ d$$

- This syntax is from the seminar paper of Chomsky and Halle (1968)
- Note that ^ is used as a morpheme boundary, and # means that we talking about a word-final "-s"

3.2 Finite-State Morphological Parsing

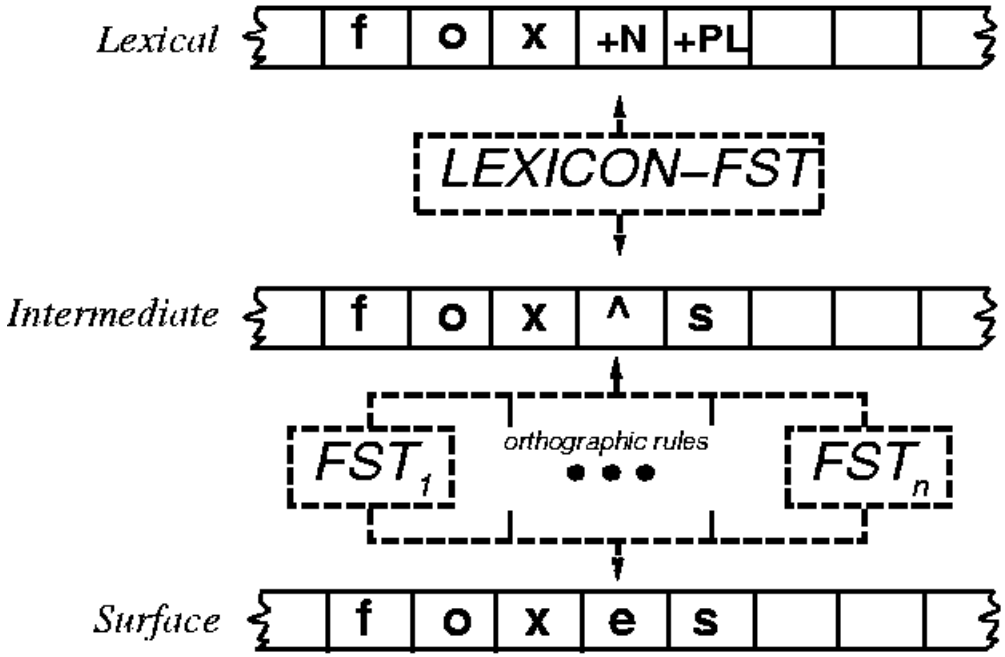
Orthographic Rules and FSTs



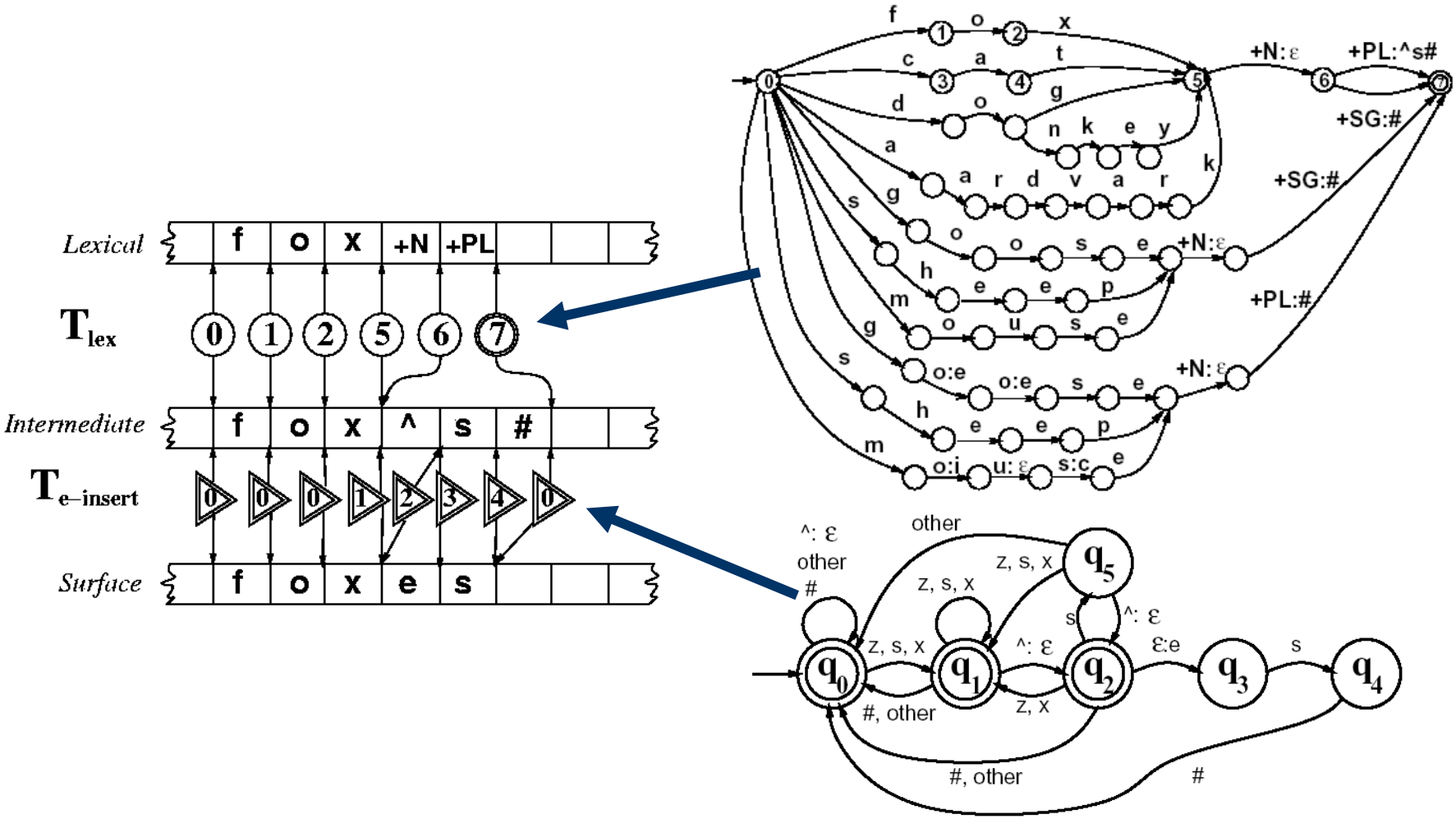
The transducer for the E-insertion rule

State \ Input	s : s	x : x	z : z	^:ε	ε : e	#	other
q_0:	1	1	1	0	-	0	0
q_1:	1	1	1	2	-	0	0
q_2:	5	1	1	0	3	0	0
q_3	4	-	-	-	-	-	-
q_4	-	-	-	-	-	0	-
q_5	1	1	1	2	-	-	0

3.3 Combining FST Lexicon and Rules



3.3 Combining FST Lexicon and Rules



3.3 Combining FST Lexicon and Rules

- The power of FSTs is that the exact same cascade with the same state sequences is used
  - when machine is generating the surface form from the lexical tape, or
  - When it is parsing the lexical tape from the surface tape.
- Parsing can be slightly more complicated than generation, because of the problem of **ambiguity.**
  - For example, *foxes* could be `fox +V +3SG` as well as `fox +N +PL`

Summary

- Two-level morphology depends on using two composed transducers to capture complex morphological phenomena
- The example we looked at involved the orthography of realizing the plural morpheme "-s" in English
- Two-level morphology is the technology behind most morphological analysis systems

- Thank you for your attention