Words, Morphology and FST

Agenda

- Words
- Survey of English Morphology
- Morphological Parsing
- Regular Expressions
- Finite State Transducers

Introduction

- **Morphology**: the **study of word structure**, the way **words are formed** and the **way their form interacts with other aspects** of grammar such as phonology and syntax
- Example: Plural
  - Boy → Boys
  - Girl → Girls
  - Fox → Foxes
  - Fish → fish
  - Goose → geese
- Example: Tense
  - Play → Playing, Played, Plays
  - Read → reading, read, reads

Morphological Parsing

- **Morphological Parsing:** the process of determining the morphemes from which a given word is constructed
- The problem of recognizing that foxes breaks down into the two morphemes fox and -es
- It is **quite inefficient to list all forms of noun and verb** in the dictionary because the productivity of the forms. → Parsers required to process
- Morphological parsing is necessary in
  - Machine translation
  - Spelling checking
  - Information retrieval (for searching different inflection of user query)

Morphology: Challenges

- New words in open class (noun, verb) are getting added day by day
- A word can have at least 3 different morphological forms → storing all morphological variants is not practical
- Morphologically rich languages: Turkish, Indian Languages (Sanksrit-- Sandhi and samasa)

key algorithm for morphological parsing, the finite state transducer

Survey of English Morphology

- **Morphology** is the study of the way words are built up from smaller meaning bearing units
- **Morphemes** are the smallest meaning bearing units
- How many morphemes in fox, cats
- Morphemes can be **stems or affixes**
- **Stem** is the main morpheme while **affixes** add additional meaning
- Affixes can be suffixes, prefixes, infixes, and circumfixes

Survey of English Morphology

- Affixes can be suffixes, prefixes, infixes, and circumfixes
- Suffix: happily, believable
- Prefix: unhappy, inefficient
- No infix and circumfix in English (other languages have.. Just explore)
- More than one affix is possible
  - Unconditionally
- Agglutinative Languages allow many number of affixes

Survey of English Morphology

- Two broad classes of ways to form words from morphemes:
- **Inflection:** the combination of a word stem with a grammatical morpheme, usually resulting in a word of the same class as the original tem, and usually filling some syntactic function like agreement

  boy (N)+s → boys (N)

- **Derivation:** the combination of a word stem with a grammatical morpheme, usually resulting in a word of a different class, often with a meaning hard to predict exactly.

  Compute (v) +ional → computational (N)

Survey of English Morphology

In English, only nouns, verbs, and sometimes adjectives can be inflected, and the number of affixes is quite small.

- Inflections of nouns in English:

– An affix marking **plural,**

- `cat(-s), thrush(-es), ox (oxen), mouse (mice)`
- `ibis(-es), waltz(-es), finch(-es), box(-es),`

`butterfly(-lies)`

– An affix marking **possessive**

- `llama’s, children’s, llamas’, Euripides’ comedies`

Survey of English Morphology

- Verbal inflection is more complicated than nominal inflection.
  - English has three kinds of verbs:
    - **Main verbs**, *eat*, *sleep*, *impeach*
    - **Modal verbs**, *can will, should*
    - **Primary verbs**, *be, have, do*
  - Morphological forms of regular verbs

stem	walk	merge	try	map
-*s* form	walks	merges	tries	maps
-*ing* principle	walking	merging	trying	mapping
Past form or –*ed* participle	walked	merged	tried	mapped

  - These regular verbs and forms are significant in the morphology of English because of their *majority* and being *productive*.

Survey of English Morphology

– Morphological forms of irregular verbs

stem	eat	catch	cut
-*s* form	eats	catches	cuts
-*ing* principle	eating	catching	cutting
Past form	ate	caught	cut
–*ed* participle	eaten	caught	cut

Survey of English Morphology (Derivation)

Nominalization in English:

– The formation of new nouns, often from verbs or adjectives

Suffix	Base Verb/Adjective	Derived Noun
-action	computerize (V)	computerization
-ee	appoint (V)	appointee
-er	kill (V)	killer
-ness	fuzzy (A)	fuzziness

– Adjectives derived from nouns or verbs

Suffix	Base Noun/Verb	Derived Adjective
-al	computation (N)	computational
-able	embrace (V)	embraceable
-less	clue (A)	clueless

Survey of English Morphology

Cliticization

Clitic is a unit whose meaning lies in between that of an affix and word

Will → ‘ll
have→ ‘ve
am→ ‘m
has→ ‘s

Ambiguous:

She’s → she is or she has

Survey of English Morphology

Agreement

the subject noun and the main verb in English have to agree in number, meaning that the two must either be both singular or both plural.

Some languages have gender agreement (Eg: Hindi, Tamil)

Gender is sometimes marked explicitly on a noun

Eg: Tamil

Assignment (ungraded)

Try to analyse morphological properties of Indian languages (Sanskrit, Malayalam, Tamil, Kannada, Hindi, Telugu,...)

Give examples for inflections and derivations

Give examples for infix, suffix, prefix cases

Give examples for regular and irregular nouns

Give examples for regular and irregular verbs

Finite State Morphological Parsing

Goal:

English		Spanish		
Input	Morphologically Parsed Output	Input	Morphologically Parsed Output	Gloss
cats	cat +N +PL	pavos	pavo +N +Masc +Pl	‘ducks’
cat	cat +N +SG	pavo	pavo +N +Masc +Sg	‘duck’
cities	city +N +Pl	bebo	beber +V +PInd +1P +Sg	‘I drink’
geese	goose +N +Pl	canto	cantar +V +PInd +1P +Sg	‘I sing’
goose	goose +N +Sg	canto	canto +N +Masc +Sg	‘song’
goose	goose +V	puse	poner +V +Perf +1P +Sg	‘I was able’
gooses	goose +V +1P +Sg	vino	venir +V +Perf +3P +Sg	‘he/she came’
merging	merge +V +PresPart	vino	vino +N +Masc +Sg	‘wine’
caught	catch +V +PastPart	lugar	lugar +N +Masc +Sg	‘place’
caught	catch +V +Past			

Stem + assorted morphological features

Finite State Morphological Parsing

In order to build a morphological parser, we'll need at least the following:

1. **lexicon**: the list of stems and affixes, together with basic information about them (whether a stem is a Noun stem or a Verb stem, etc.).

2. **morphotactics**: the model of morpheme ordering that explains which classes of morphemes can follow other classes of morphemes inside a word. For example, the fact that the English plural morpheme follows the noun rather than preceding it is a morphotactic fact.

3. **orthographic rules**: these spelling rules are used to model the changes that occur in a word, usually when two morphemes combine (e.g., the y → ie spelling rule discussed above that changes city + -s to cities rather than citys)

Building Lexicon

Lexicon is the repository of words

List of all possible words is impossible

Practical: store all stems and morphemes and morphotactics

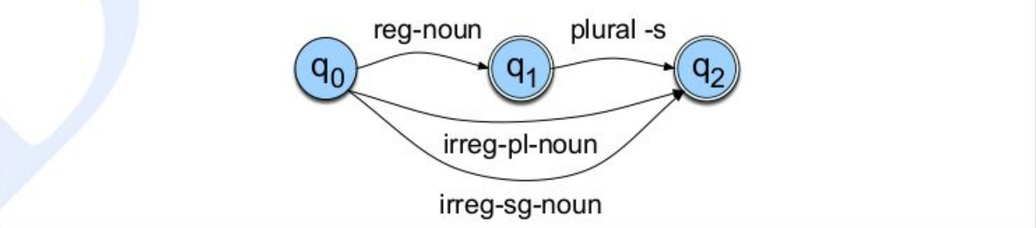


Figure 3.3 A finite-state automaton for English nominal inflection.

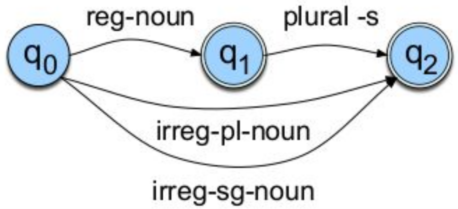


Figure 3.3 A finite-state automaton for English nominal inflection.

Lexicon includes regular nouns (reg-noun) that take the regular -s plural (e.g., cat, dog, fox, aardvark). These are the vast majority of English nouns since for now we will ignore the fact that the plural of words like fox have an inserted e: foxes. The lexicon also includes irregular noun forms that don’t take -s, both singular irreg-sg-noun (goose, mouse) and plural irreg-pl-noun (geese, mice).

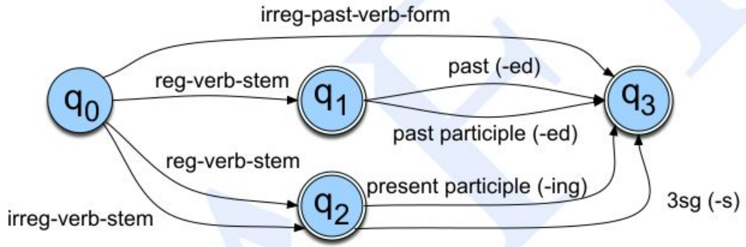


Figure 3.4 A finite-state automaton for English verbal inflection

This lexicon has three stem classes (reg-verb-stem, irreg-verb-stem, and irreg-past-verb-form), plus four more affix classes (*-ed* past, *-ed* participle, *-ing* participle, and third singular *-s*):

reg-verb-stem	**irreg-verb-stem**	**irreg-past-verb**	**past**	**past-part**	**pres-part**	**3sg**
walk fry talk impeach	cut speak sing	caught ate eaten sang	-ed	-ed	-ing	-s

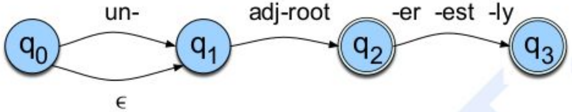
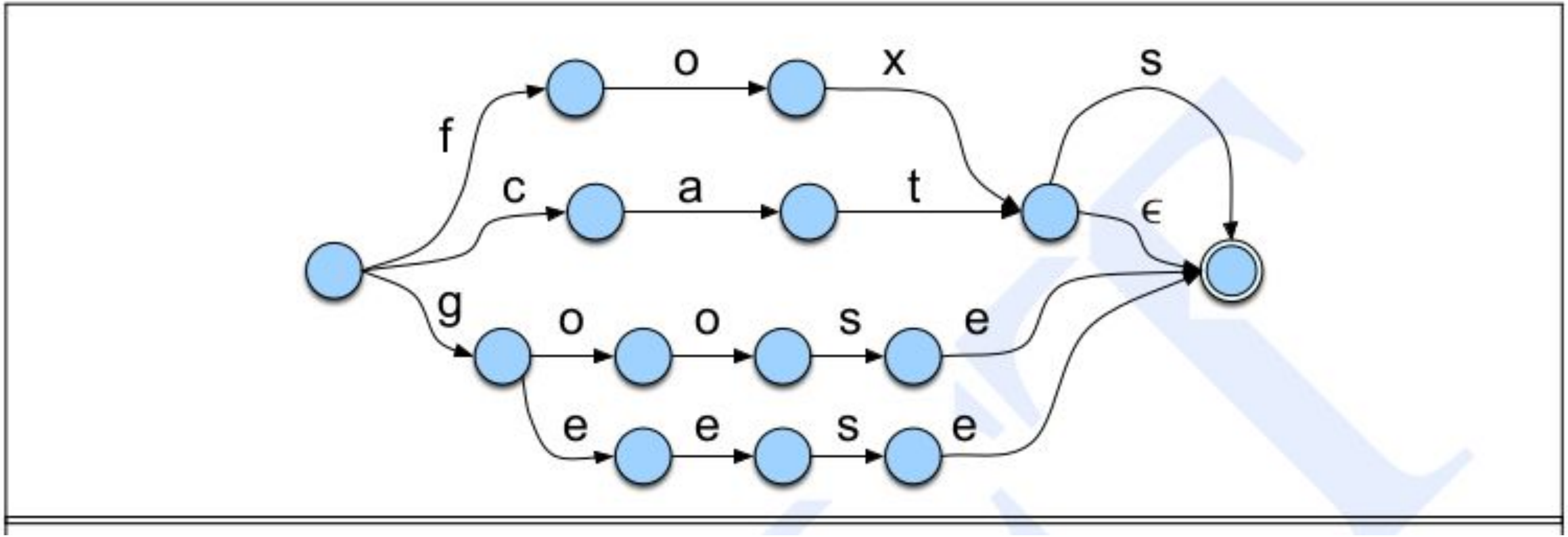


Figure 3.5 An FSA for a fragment of English adjective morphology: Antworth's Proposal #1.

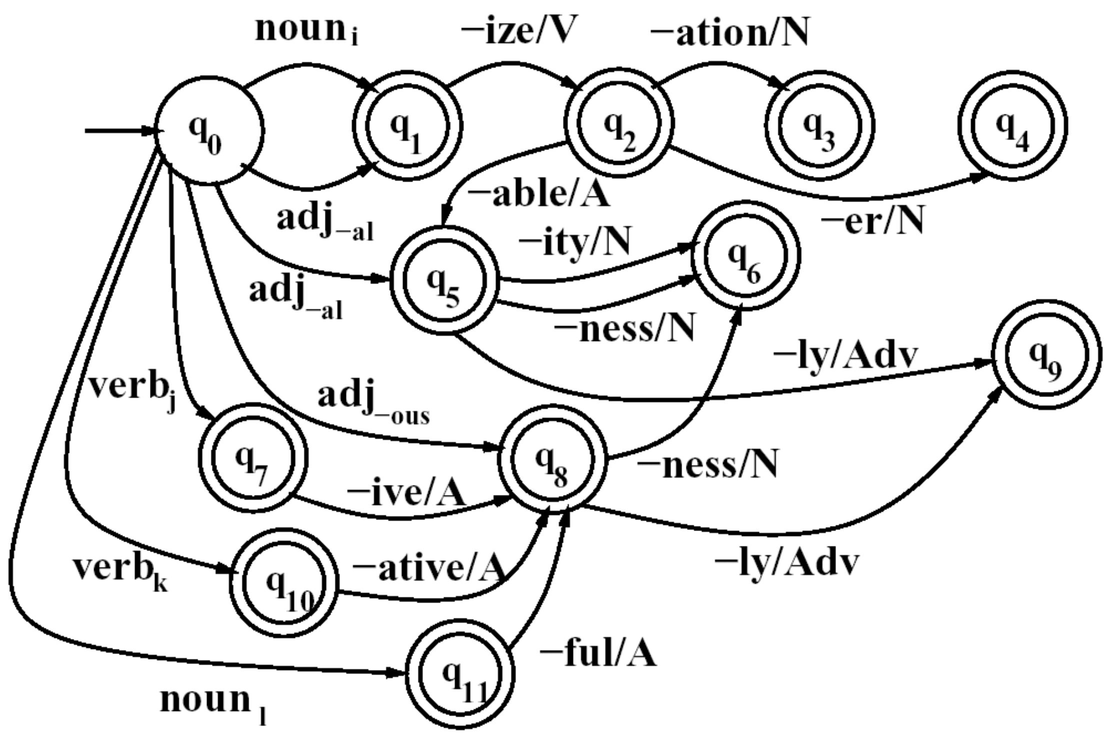
Alas, while this FSA will recognize all the adjectives in the table above, it will also

FSA for morphological recognition



Next: FST

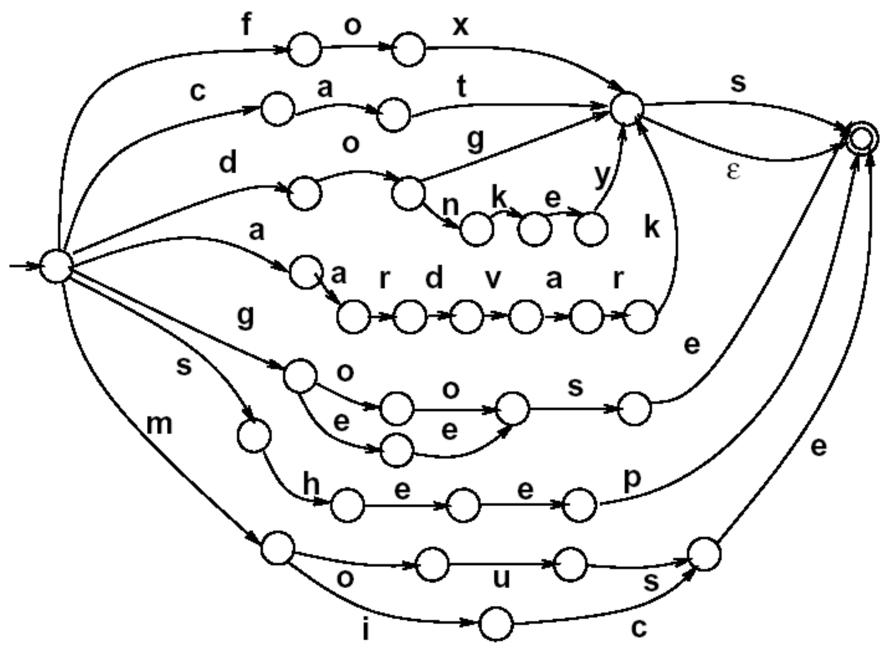
Finite-State Morphological Parsing



An FSA for another fragment of English derivational morphology

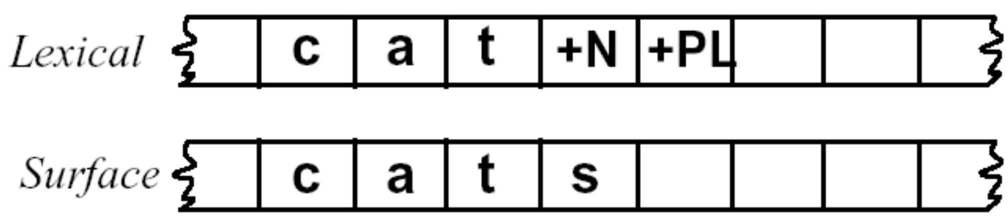
Finite-State Morphological Parsing

- We can now use these FSAs to solve the problem of **morphological recognition**:
  - Determining whether an input string of letters makes up a legitimate English word or not
  - We do this by taking the morphotactic FSAs, and plugging in each “sub-lexicon” into the FSA.
  - The resulting FSA can then be defined as the level of the individual letter.



Finite-State Morphological Parsing
Morphological Parsing with FST

- Given the input, for example, *cats*, we would like to produce `cat +N +PL`.
- Two-level morphology, by Koskenniemi (1983)
  - Representing a word as a correspondence between a **lexical level**
    - Representing a simple concatenation of morphemes making up a word, and
  - The **surface level**
    - Representing the actual spelling of the final word.
- Morphological parsing is implemented by building mapping rules that maps letter sequences like *cats* on the surface level into morpheme and features sequence like `cat +N +PL` on the lexical level.



Finite-State Morphological Parsing

Morphological Parsing with FST

- The automaton we use for performing the mapping between these two levels is the **finite-state transducer** or **FST**.
  - A transducer maps between one set of symbols and another;
  - An FST does this via a finite automaton.
- Thus an FST can be seen as a two-tape automaton which **recognizes** or **generates** ***pairs*** of strings.
- The FST has a more general function than an FSA:
  - An FSA defines a formal language
  - An FST defines a relation between sets of strings.
- Another view of an FST:
  - A machine reads one string and generates another.

Finite-State Morphological Parsing
Morphological Parsing with FST

- **FST as recognizer:**
  - a transducer that takes a pair of strings as input and output *accept* if the string-pair is in the string-pair language, and a *reject* if it is not.
- **FST as generator:**
  - a machine that outputs pairs of strings of the language. Thus the output is a yes or no, and a pair of output strings.
- **FST as transducer:**
  - A machine that reads a string and outputs another string.
- **FST as set relater:**
  - A machine that computes relation between sets.

Finite-State Morphological Parsing

Morphological Parsing with FST

- A formal definition of FST (based on the **Mealy machine** extension to a simple FSA):
  - Q: a finite set of N states $q_0, q_1, \ldots, q_N$
  - Σ: a finite alphabet of complex symbols. Each complex symbol is composed of an input-output pair $i : o$; one symbol I from an input alphabet I, and one symbol o from an output alphabet O, thus $\Sigma \subseteq I \times O$. I and O may each also include the epsilon symbol ε.
  - q_0: the start state
  - F: the set of final states, $F \subseteq Q$
  - $\delta(q, i{:}o)$: the transition function or transition matrix between states. Given a state $q \in Q$ and complex symbol $i{:}o \in \Sigma$, $\delta(q, i{:}o)$ returns a new state q' $\in Q$. δ is thus a relation from $Q \times \Sigma$ to Q.

Finite-State Morphological Parsing

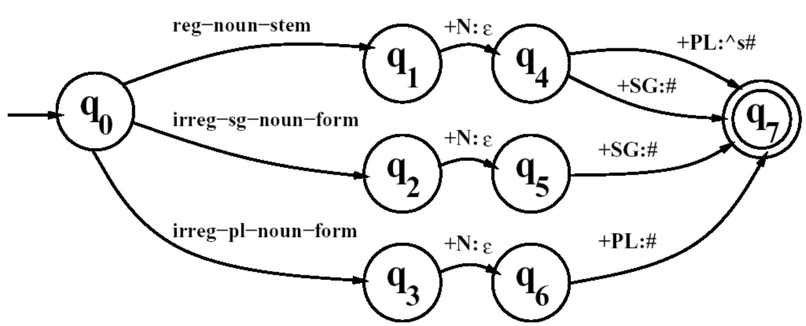
Morphological Parsing with FST

- FSAs are isomorphic to regular languages, FSTs are isomorphic to **regular relations.**
- Regular relations are sets of pairs of strings, a natural extension of the regular language, which are sets of strings.
- FSTs are closed under union, but generally they are not closed under difference, complementation, and intersection.
- Two useful closure properties of FSTs:
  - **Inversion:** If T maps from I to O, then the inverse of T, T^{-1} maps from O to I.
  - **Composition:** If T_1 is a transducer from I_1 to O_1 and T_2 a transducer from I_2 to O_2, then $T_1 \circ T_2$ maps from I_1 to O_2

Finite-State Morphological Parsing

Morphological Parsing with FST

- Inversion is useful because it makes it easy to convert a FST-as-parser into an FST-as-generator.
- Composition is useful because it allows us to take two transducers than run in series and replace them with one complex transducer.
  - $T_1 \circ T_2(S) = T_2(T_1(S))$

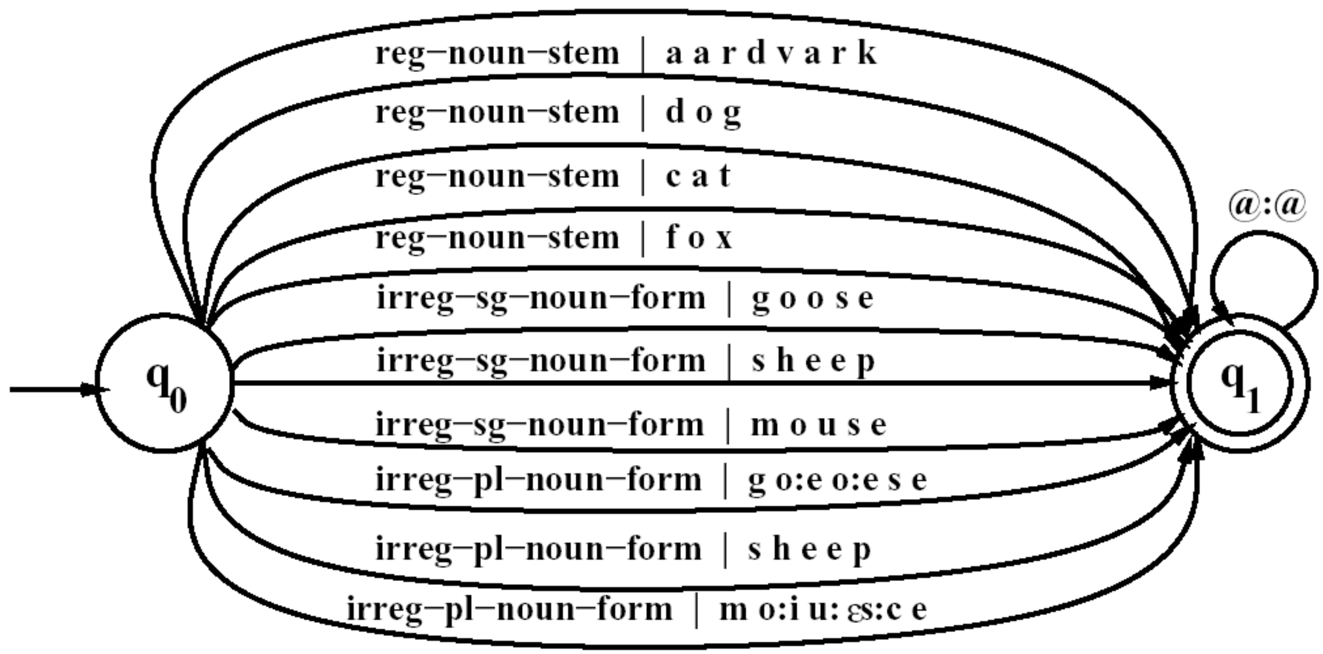


A transducer for English nominal number inflection T_{num}

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

Finite-State Morphological Parsing

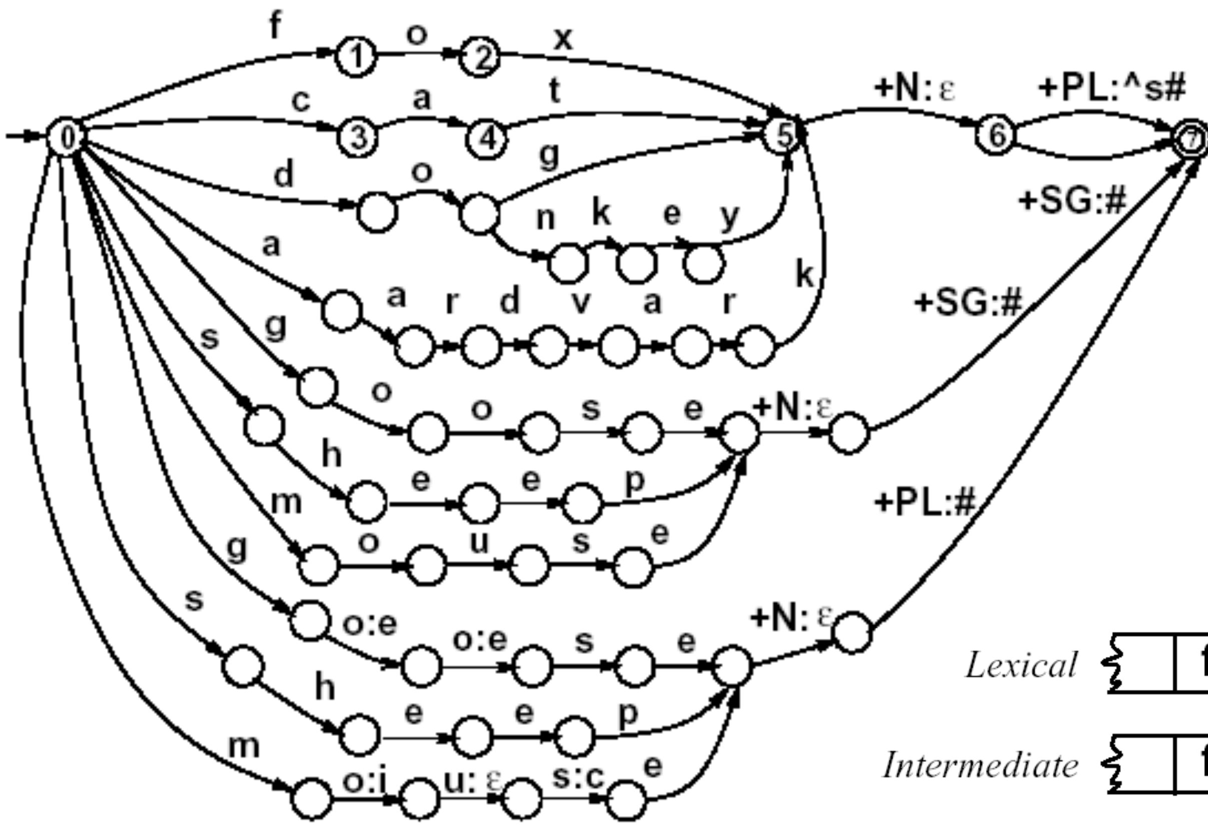
Morphological Parsing with FST



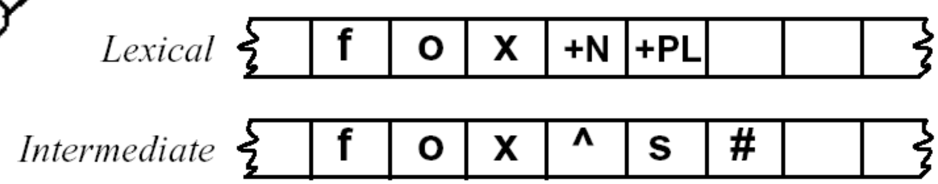
The transducer T_{stems}, which maps roots to their root-class

Finite-State Morphological Parsing

Morphological Parsing with FST



^: morpheme boundary
#: word boundary



A fleshed-out English nominal inflection FST

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

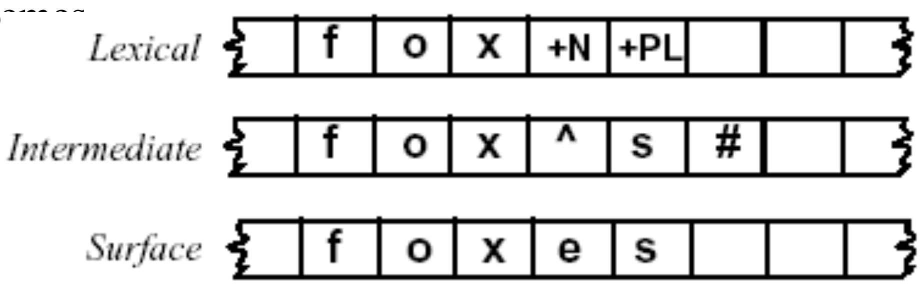
Finite-State Morphological Parsing

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 morph[…]



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*"

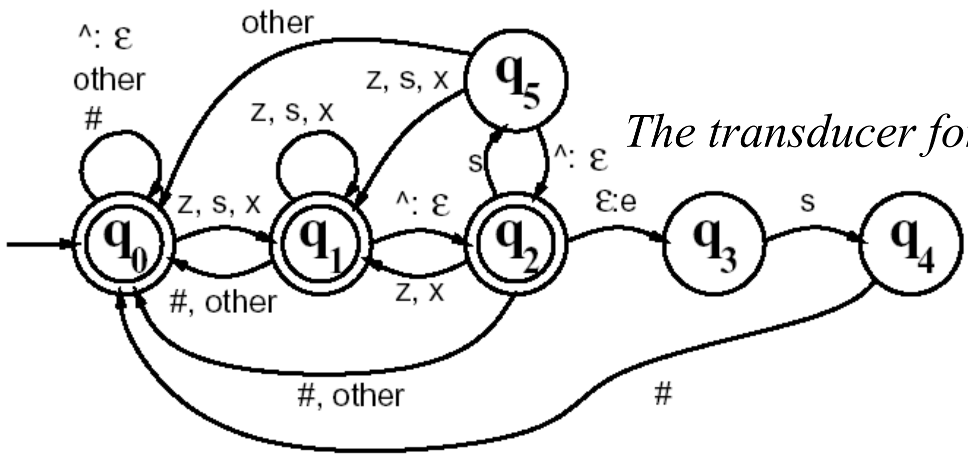
$$\varepsilon \rightarrow \mathrm{e} / \left\{ \begin{matrix} x \\ s \\ z \end{matrix} \right\} \_\_\ s\#$$

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

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

Finite-State Morphological Parsing

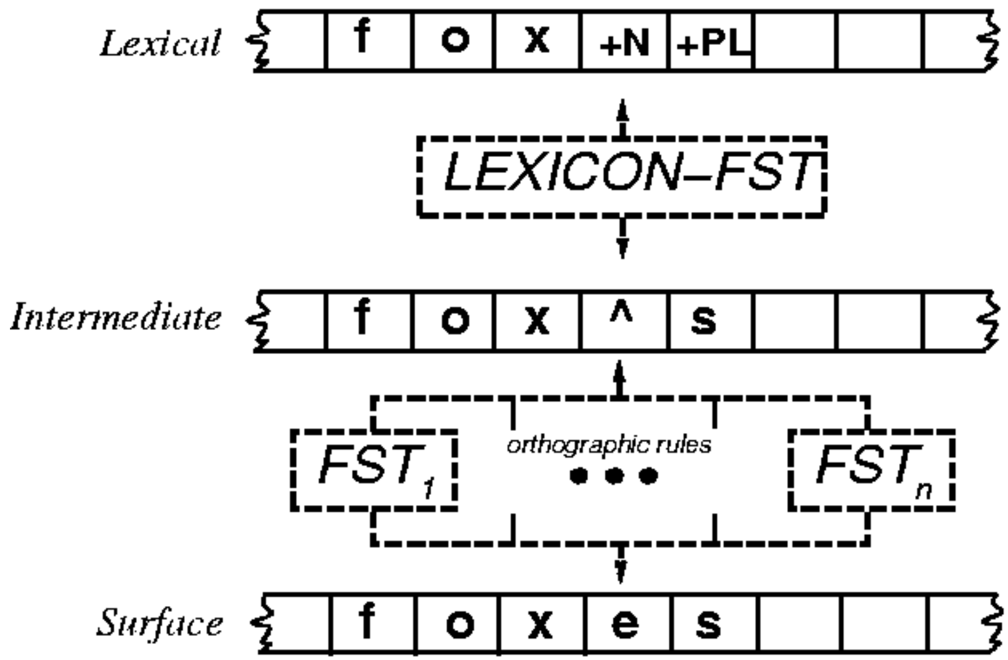
Orthographic Rules and FSTs



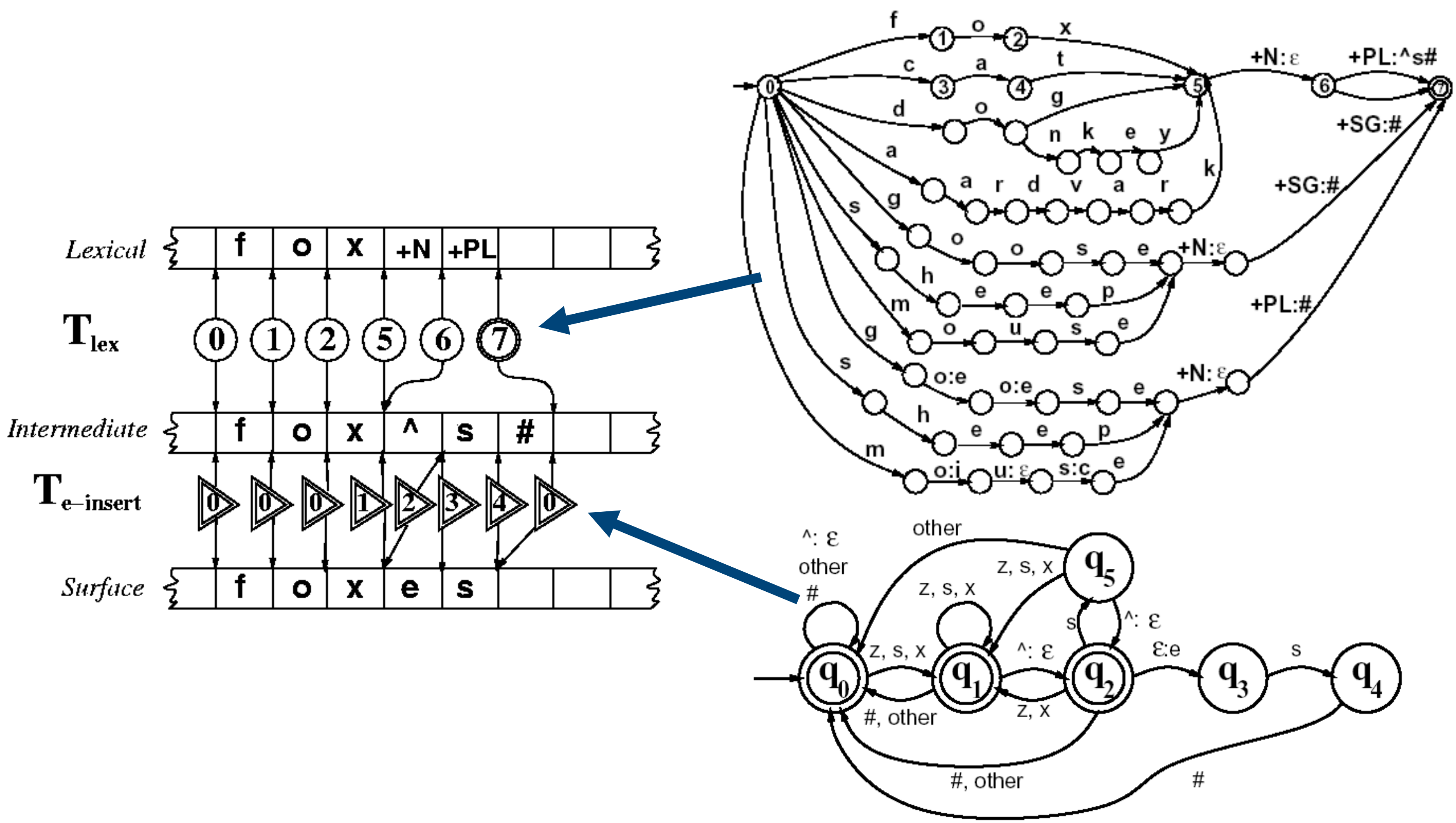
The transducer for the E-insertion rule

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

Combining FST Lexicon and Rules



Combining FST Lexicon and Rules



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`

Lexicon-Free FSTs: the Porter Stemmer

- Information retrieval
- One of the mostly widely used **stemmming** algorithms is the simple and efficient Porter (1980) algorithm, which is based on a series of simple cascaded rewrite rules.
  - ATIONAL → ATE (e.g., relational → relate)
  - ING → εif stem contains vowel (e.g., motoring → motor)
- Problem:
  - Not perfect: error of commision, omission
- Experiments have been made
  - Some improvement with smaller documents
  - Any improvement is quite small