

Chapter 3. Morphology and Finite-State Transducers

From: Chapter 3 of *An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*, by Daniel Jurafsky and James H. Martin

Background

- The problem of recognizing that *foxes* breaks down into the two morphemes *fox* and *-es* is called *morphological parsing*.
- Similar problem in the information retrieval domain: *stemming*
- Given the **surface** or **input form** *going*, we might want to produce the parsed form: VERB-go + GERUND-ing
- In this chapter
 - morphological knowledge and
 - The **finite-state transducer**
- It is quite inefficient to list all forms of noun and verb in the dictionary because the productivity of the forms.
- Morphological parsing is necessary for more than just IR(Information Retrieval), but also
 - Machine translation
 - Spelling checking

3.1 Survey of (Mostly) English Morphology

- Morphology is the study of the way words are built up from smaller meaning-bearing units, **morphemes**.
- Two broad classes of morphemes:
 - **The stems:** the “main” morpheme of the word, supplying the main meaning, while
 - **The affixes:** add “additional” meaning of various kinds.
- Affixes are further divided into **prefixes, suffixes, infixes, and circumfixes**.
 - Suffix: *eat-s*
 - Prefix: *un-buckle*
 - Circumfix: *ge-sag-t* (said) *sagen* (to say) (in German)
 - Infix: *hingi* (borrow) *humingi* (the agent of an action))in Philippine language Tagalog)

3.1 Survey of (Mostly) English Morphology

- Prefixes and suffixes are often called **concatenative morphology**.
- A number of languages have extensive **non-concatenative morphology**
 - The Tagalog infixation example
 - **Templatic morphology** or **root-and-pattern morphology**, common in Arabic, Hebrew, and other Semitic languages
- 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 stem, and usually filling some syntactic function like agreement, and
 - **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.

3.1 Survey of (Mostly) English Morphology

Inflectional 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

3.1 Survey of (Mostly) English Morphology

Inflectional 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*.

3.1 Survey of (Mostly) English Morphology

Inflectional 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

3.1 Survey of (Mostly) English Morphology

Derivational Morphology

- **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 (N)	clueless

3.1 Survey of (Mostly) English Morphology

Derivational Morphology

- Derivation in English is more complex than inflection because
 - Generally less productive
 - A nominalizing affix like *-ation* can not be added to absolutely every verb.
eatation(*)
 - There are subtle and complex meaning differences among nominalizing suffixes. For example, *sincerity* has a subtle difference in meaning from *sincereness*.

3.2 *Finite-State Morphological Parsing*

- Parsing English morphology

Input	Morphological parsed output
cats	cat +N +PL
cat	cat +N +SG
cities	city +N +PL
geese	goose +N +PL
goose	(goose +N +SG) or (goose +V)
gooses	goose +V +3SG
merging	merge +V +PRES-PART
caught	(caught +V +PAST-PART) or (catch +V +PAST)

Stems and morphological features

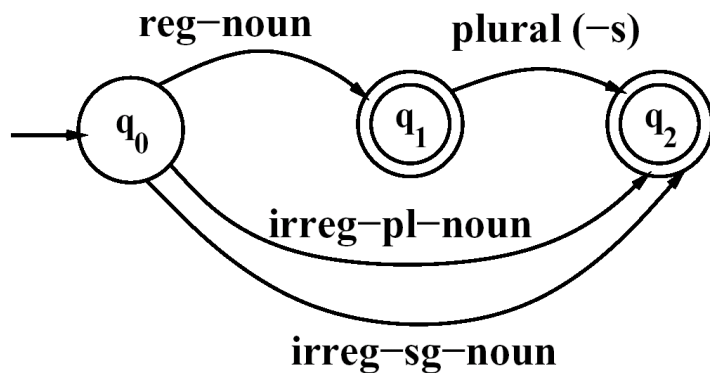
3.2 *Finite-State Morphological Parsing*

- We need at least the following to build a morphological parser:
 1. **Lexicon:** the list of stems and affixes, together with basic information about them (Noun stem or 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. E.g., the rule that English plural morpheme follows the noun rather than preceding it.
 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 $y \rightarrow ie$ spelling rule changes *city* + *-s* to *cities*).

3.2 Finite-State Morphological Parsing

The Lexicon and Morphotactics

- A lexicon is a repository for words.
 - The simplest one would consist of an explicit list of every word of the language. *Incovenient or impossible!*
 - Computational lexicons are usually structured with
 - a list of each of the stems and
 - Affixes of the language together with a representation of morphotactics telling us how they can fit together.
 - The most common way of modeling morphotactics is the finite-state automaton.

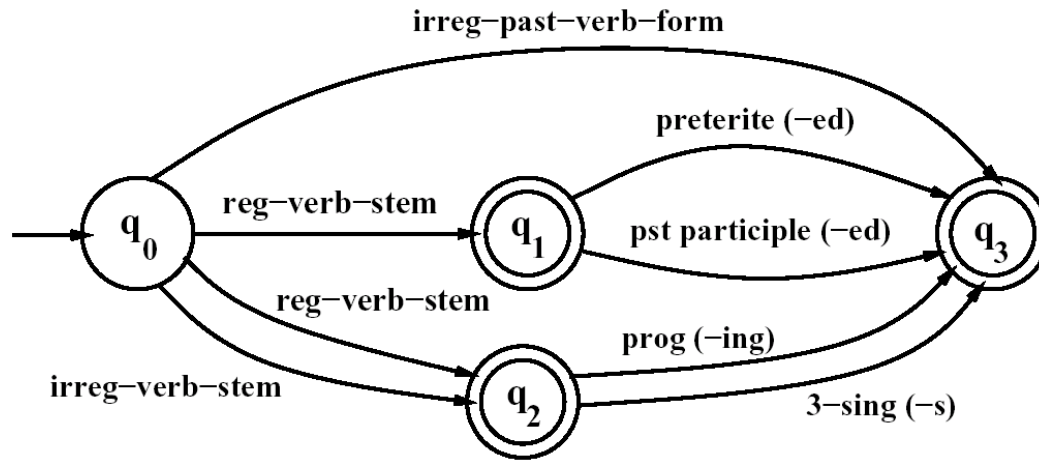


An FSA for English nominal inflection

Reg-noun	Irreg-pl-noun	Irreg-sg-noun	plural
fox	geese	goose	-s
fat	sheep	sheep	
fog	Mice	mouse	
fardvark			

3.2 Finite-State Morphological Parsing

The Lexicon and Morphotactics



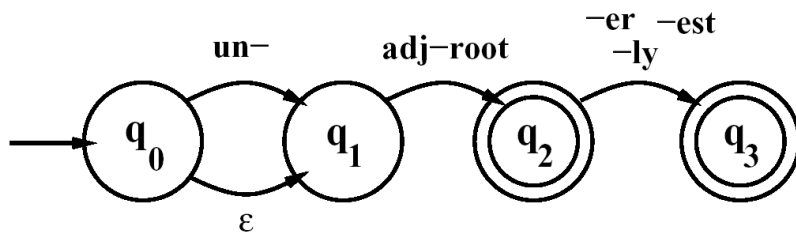
An FSA for English verbal inflection

Reg-verb-stem	Irreg-verb-stem	Irreg-past-verb	past	Past-part	Pres-part	3sg
walk	cut	caught	-ed	-ed	-ing	-s
fry	speak	ate				
talk	sing	eaten				
impeach	sang					
	spoken					

3.2 *Finite-State Morphological Parsing*

The Lexicon and Morphotactics

- English derivational morphology is more complex than English inflectional morphology, and so automata of modeling English derivation tends to be quite complex.
 - Some even based on CFG
- A small part of morphosyntactics of English adjectives

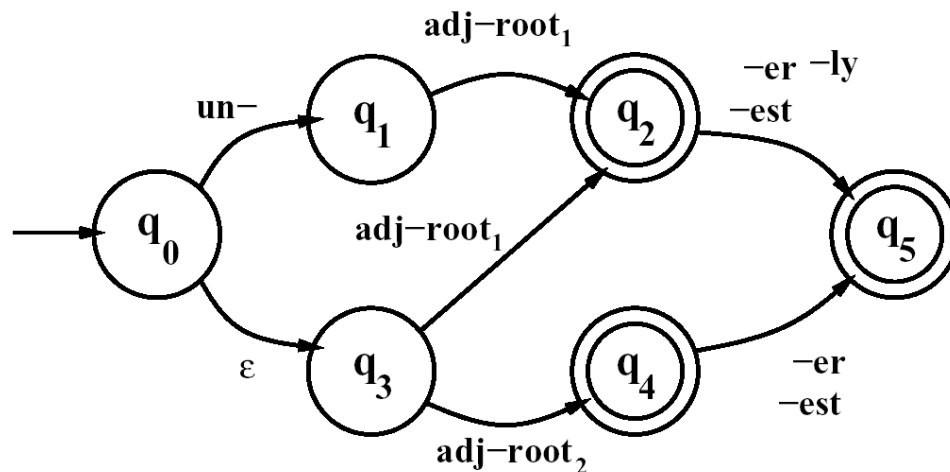


An FSA for a fragment of English adjective Morphology #1

big, bigger, biggest
cool, cooler, coolest, coolly
red, redder, reddest
clear, clearer, clearest, clearly, unclear, unclearly
happy, happier, happiest, happily
unhappy, unhappier, unhappiest, unhappily
real, unreal, really

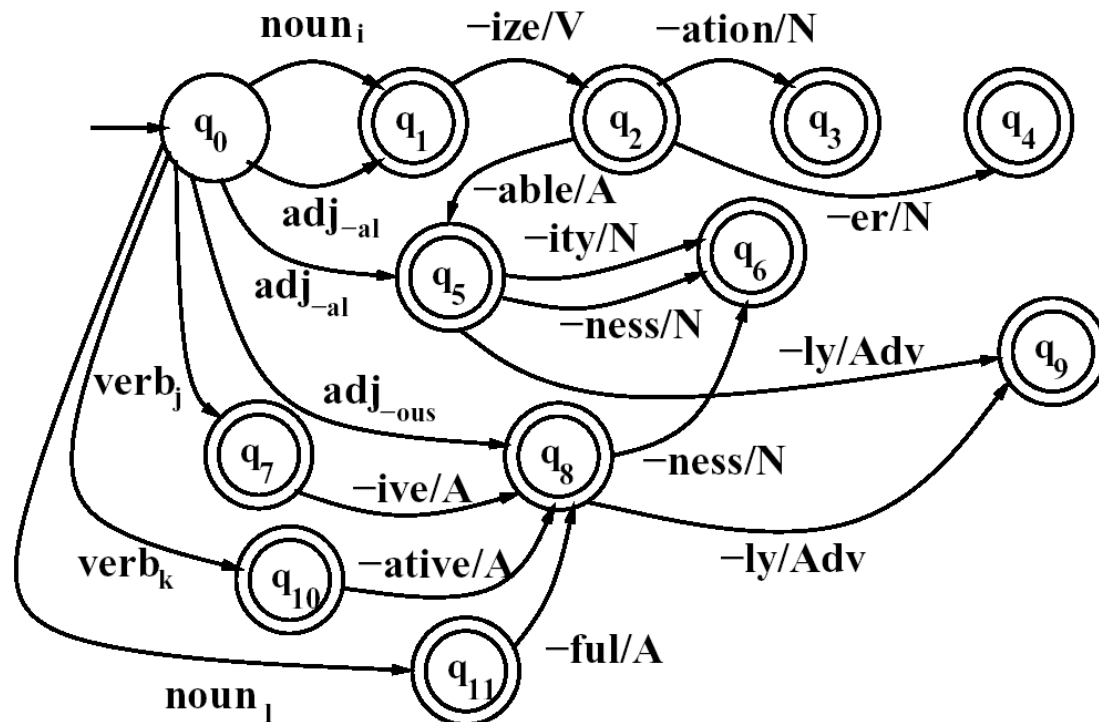
3.2 Finite-State Morphological Parsing

- The FSA#1 recognizes all the listed adjectives, and ungrammatical forms like *unbig*, *redly*, and *realest*.
- Thus #1 is revised to become #2.
- The complexity is expected from English derivation.



An FSA for a fragment of English adjective Morphology #2

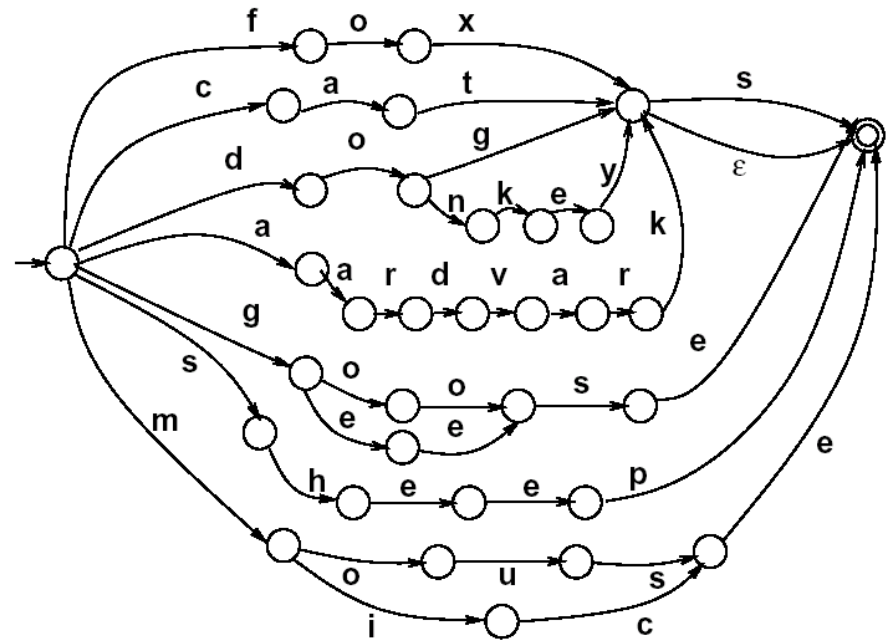
3.2 Finite-State Morphological Parsing



An FSA for another fragment of English derivational morphology

3.2 Finite-State Morphological Parsing

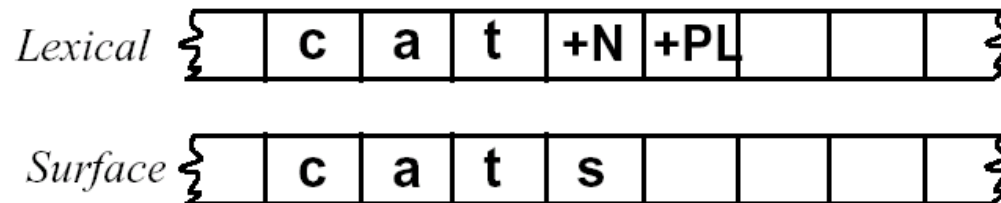
- We can now use these FSAs to solve the problem of **morphological recognition**:
 - Determining whether an input string of letters makes up a legitimate English word or not
 - We do this by taking the morphotactic FSAs, and plugging in each “sub-lexicon” into the FSA.
 - The resulting FSA can then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



3.2 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.



3.2 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.

3.2 *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.

3.2 *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, \dots, 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 .

3.2 *Finite-State Morphological Parsing*

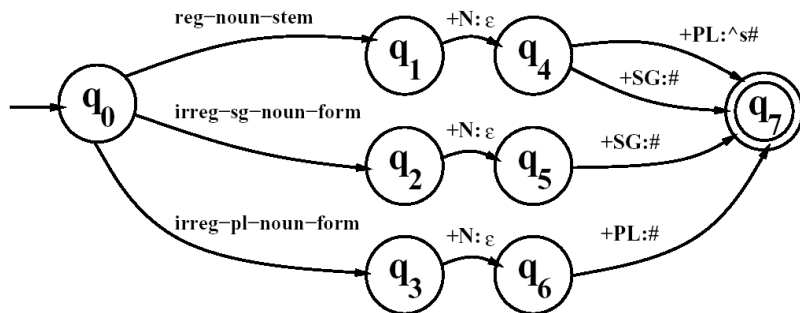
Morphological Parsing with FST

- 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

3.2 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.
 - $T_1 \circ T_2(S) = T_2(T_1(S))$

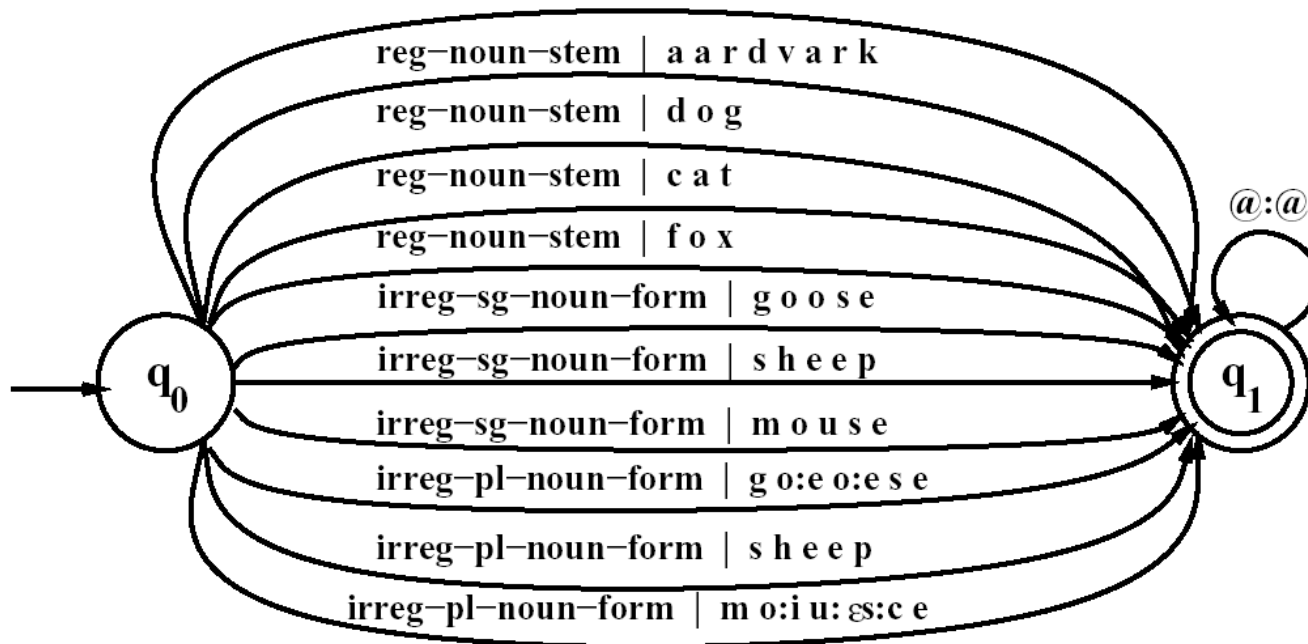


A transducer for English nominal number inflection T_{num}

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:es:c e	mouse
aardvark		

3.2 Finite-State Morphological Parsing

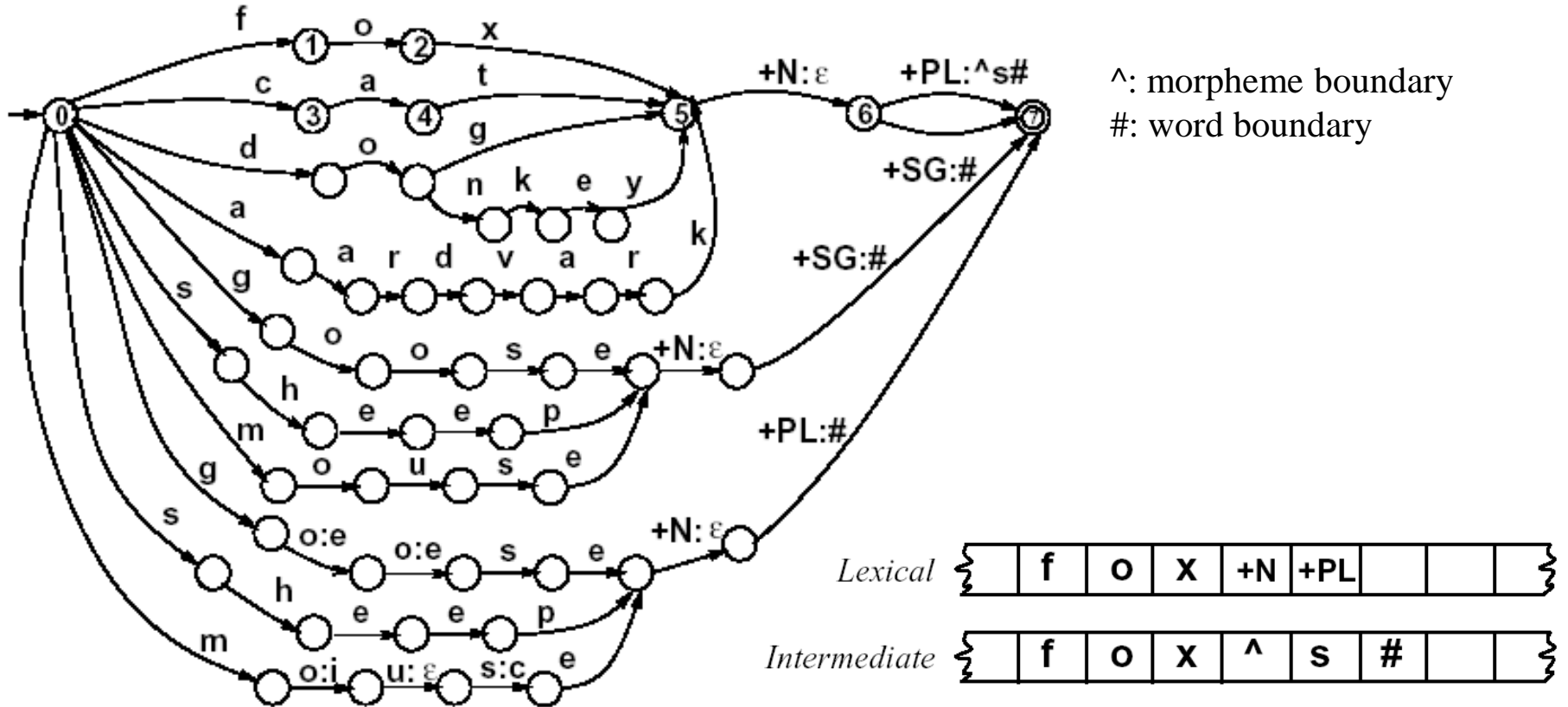
Morphological Parsing with FST



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

3.2 Finite-State Morphological Parsing

Morphological Parsing with FST



A fleshed-out English nominal inflection FST

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

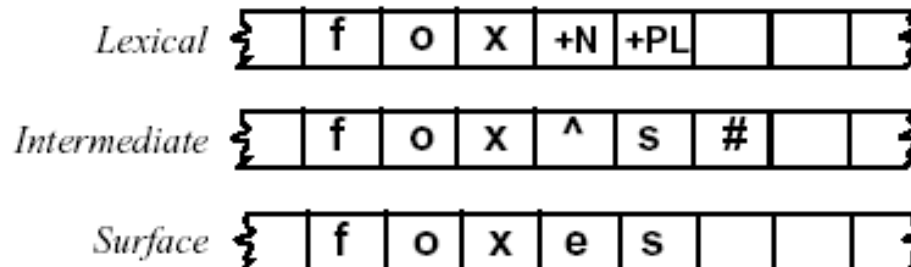
3.2 *Finite-State Morphological Parsing*

Orthographic Rules and FSTs

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

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

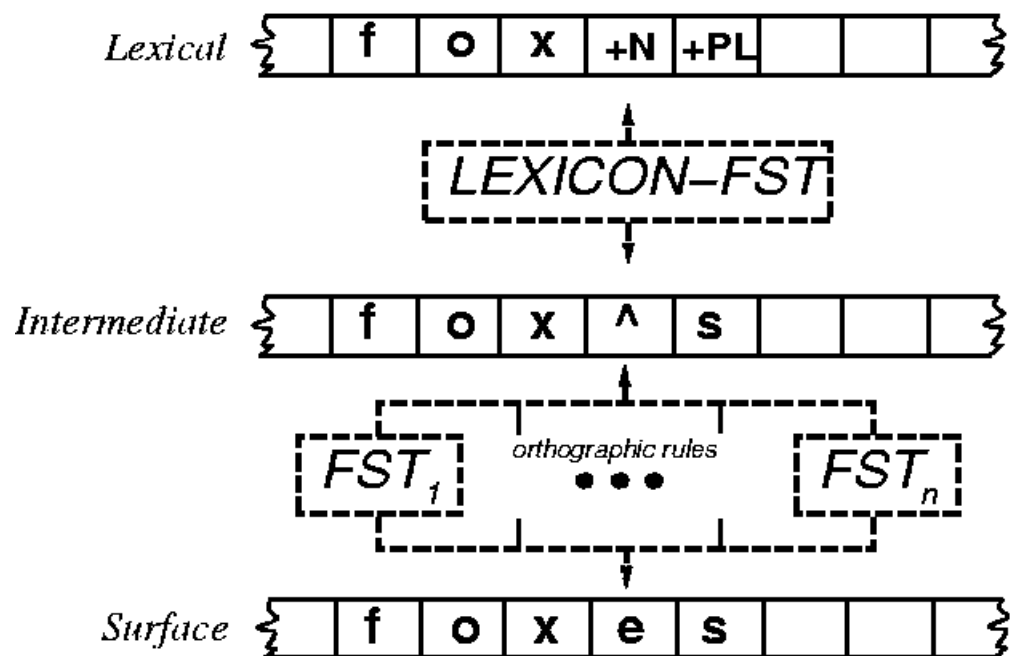
Orthographic Rules and FSTs

- “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 $-s$ ”

$$\varepsilon \rightarrow e / \left\{ \begin{array}{c} x \\ s \\ z \end{array} \right\} \uparrow \underline{s\#}$$

- “rewrite a and b when it occurs between c and d ”
 $a \rightarrow b / c _ d$

3.3 Combining FST Lexicon and Rules



3.3 Combining FST Lexicon and Rules

- Parsing can be slightly more complicated than generation, because of the problem of **ambiguity**.
 - For example, *foxes* could be $f_{OX} +V +3SG$ as well as $f_{OX} +N +PL$

3.4 Lexicon-Free FSTs: the Porter Stemmer

- Information retrieval
- One of the mostly widely used **stemming** 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