## Overview

- 1. Finite-State Morphological Parsing – An Introduction
  - Lexicon, morphotactics and orthographic rules
- 2. The Lexicon and Morphotactics
- 3. Morphological parsing with Finite-State Transducers
  - Two-level morphology
  - Finite-State Transducer
- 4. Orthographic Rules and Finite-State Transducers
  - Spelling rules

Tell me and I forget. Teach me and I remember. Involve me and I learn.
-- *Benjamin Franklin*

## Finite-State Morphological Parsing

- Parsing English morphology
  - The second column contains the stem of each word + assorted morphological *features*
  - Features specify additional information about stem
  - +N means word is a noun; +SG means singular

| **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`) |

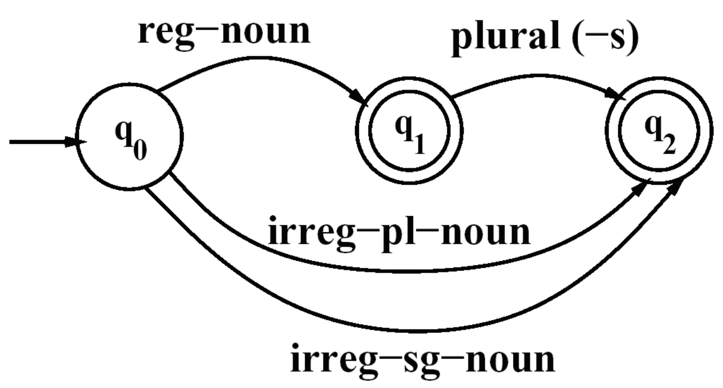
## Finite-State Morphological Parsing

- We need at least the following to build a morphological parser:
  - **Lexicon**: the *list of stems and affixes*, together with basic information about them (Noun stem or Verb stem, etc.)
  - **Morphotactics**: the model of *morpheme ordering* that explains which classes of morphemes can follow other classes of morphemes. E.g., the rule that English plural morpheme follows the noun rather than preceding it.
  - **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 changes *city* + *-s* to *cities*).

## 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. ***Inconvenient 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 modelling morphotactics is the finite-state automaton.

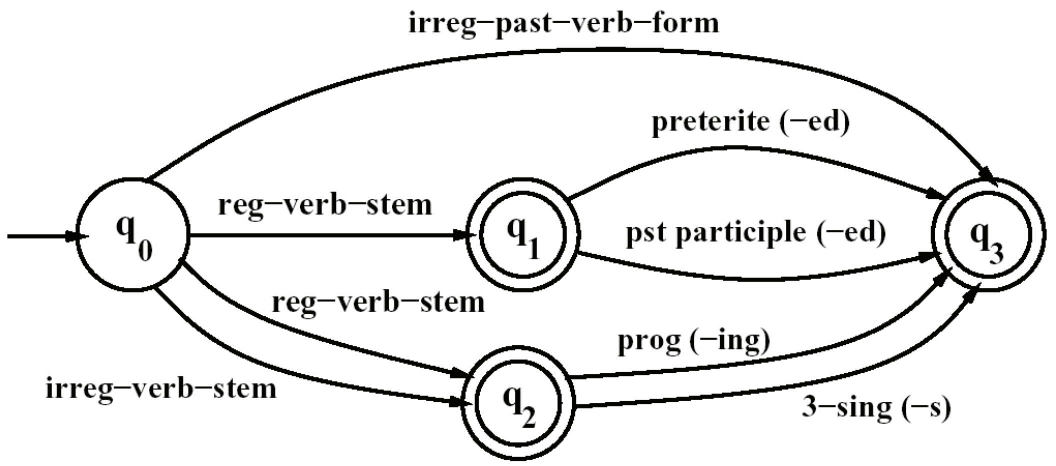
## The Lexicon and Morphotactics



*An FSA for English nominal inflection*

| Reg-noun | Irreg-pl-noun | Irreg-sg-noun | plural |
|---|---|---|---|
| fox | geese | goose | -s |
| cat | sheep | sheep | |
| table | mice | mouse | |
| book | | | |

## The Lexicon and Morphotactics



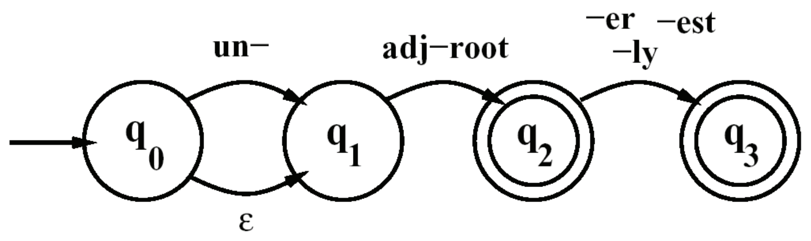
*An FSA for English verbal inflection*

| 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 | | | | |

## The Lexicon and Morphotactics

- English derivational morphology is more complex than inflectional morphology
- So automata of modeling English derivation tends to be quite complex
    - Some even based on CFG
  - [Antworth,1990] A small part of morphotactics of English adjectives
  - Adjectives can have an optional prefix (*un-*)
  - An obligatory root (*big*, *cool*, *etc*)
  - And an optional suffix (-*er*, -*est*, *or* -*ly*)

## The Lexicon and Morphotactics

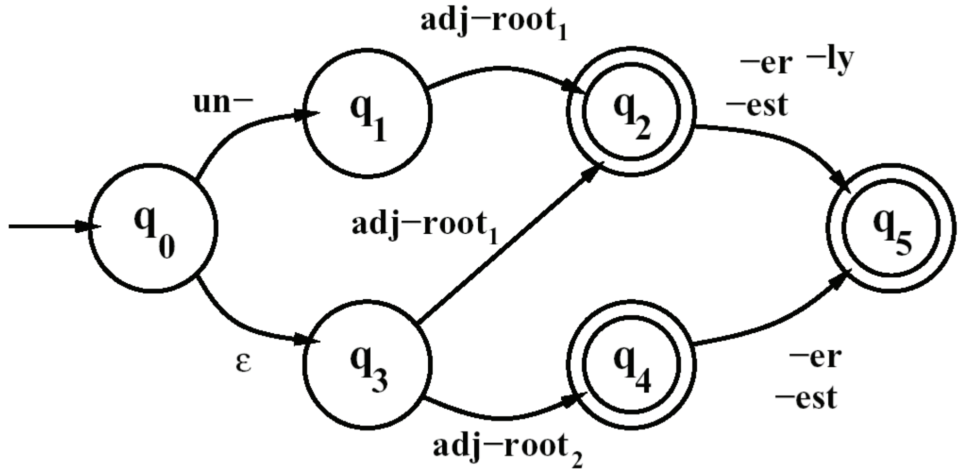


*An FSA for a fragment of English adjective Morphology #1*

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

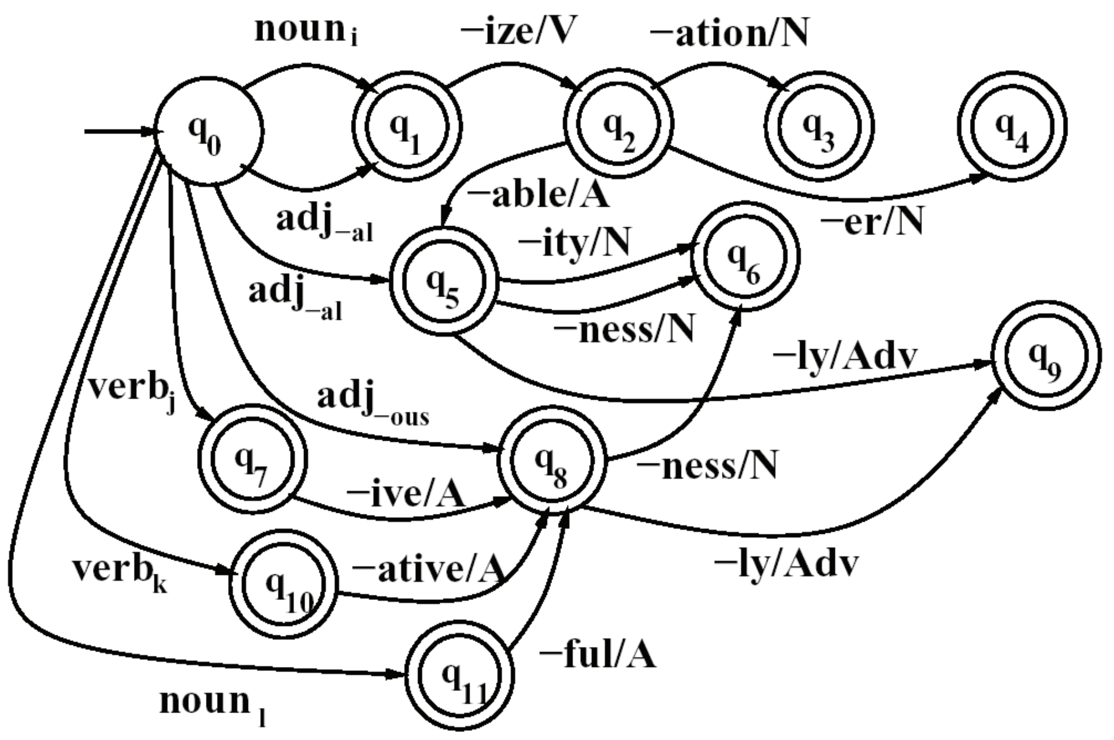
## The Lexicon and Morphotactics

- The FSA#1 recognizes all the listed adjectives, and ungrammatical forms like *unbig*, *unfast*, *redly*, *smally* 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*

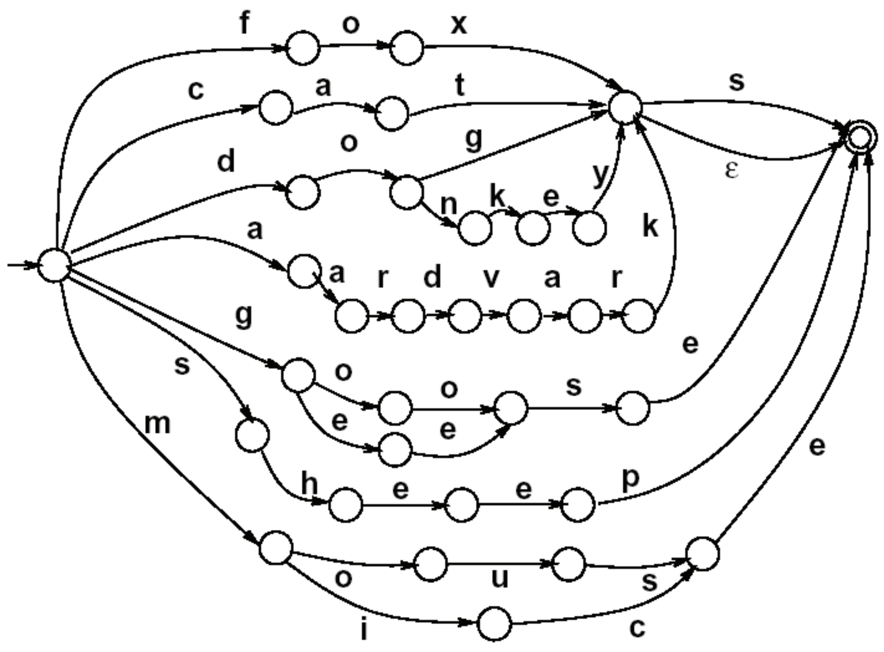
## The Lexicon and Morphotactics



*An FSA for another fragment of English derivational morphology*

## The Lexicon and Morphotactics

- FSA can be used to solve the problem of **morphological recognition**:
  - Determining whether an input string of letters makes up a *legitimate English word* or not
  - The resulting FSA can then be defined as the level of the individual letter.



*An FSA for English nominal inflection*

## Morphological Parsing with FST

- Two-level morphology
- Finite-State Transducer [FST]

## Morphological Parsing with FST

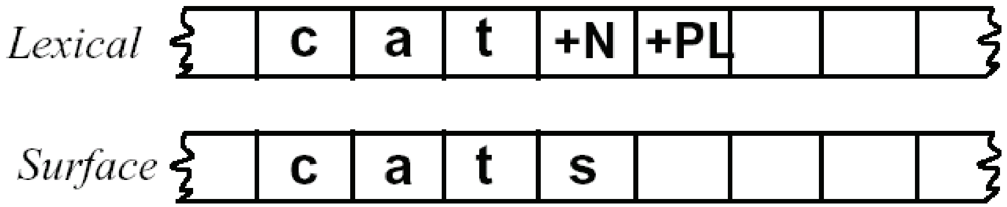
- Given the input, for example, *cats*, we would like to produce :

  `cat +N +PL`

  - Two-level morphology, by Koskenniemi (1983)
    - Represent a word as a correspondence between a **lexical level**
      - represents a concatenation of morphemes making up a word
    - and the **surface level**
      - Represents the actual spelling of the final word.

## Morphological Parsing with FST

- Morphological parsing is implemented by building mapping rules:
  - that <u>maps letter sequences</u> like *cats* on the surface level
  - <u>into morpheme and features sequence</u> like `cat +N +PL` on the lexical level.



## 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.

## 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 translator:**
  - A machine that reads a string and outputs another string.
- **FST as set relater:**
  - A machine that computes relation between sets.

## 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 $q_0, q_1, \ldots, q_N$
  - $\Sigma$: 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 $\Sigma \subseteq I$ □$O$. $I$ and $O$ may each also include the epsilon symbol $\varepsilon$.
  - $q_0$: 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 $q' \in Q$. $\delta$ is thus a relation from $Q \times \Sigma$ to $Q$.

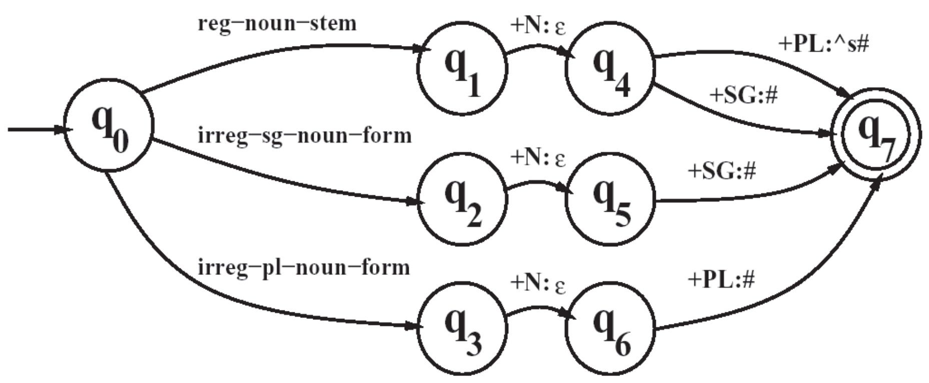
## Morphological Parsing with FST

- 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 $O_2$

## 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. $T_1 \circ T_2(S) = T_2(T_1(S))$
- View an FST as having two tapes:
  - *upper* or *lexical tape*: characters from the left side of the *a:b* pairs
  - *lower* or *surface tape*: characters from the right side of the *a:b* pairs
  - Each symbol *a:b* expresses how the symbol *a* from one tape is mapped to the symbol *b* on the another tape

## Morphological Parsing with FST



*A lexical transducer for English nominal inflection $T_{num}$*

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

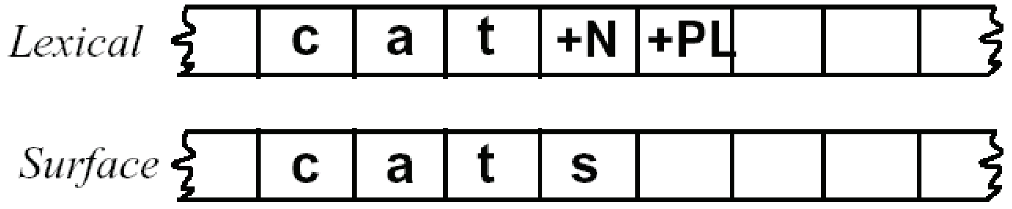
## Morphological Parsing with FST

- The transducer will map:
  - plural nouns = stem + morphological marker *+PL*
  - singular nouns = stem + morphological marker *+SG*
- Thus a surface `cats` will map to `cat +N +PL`

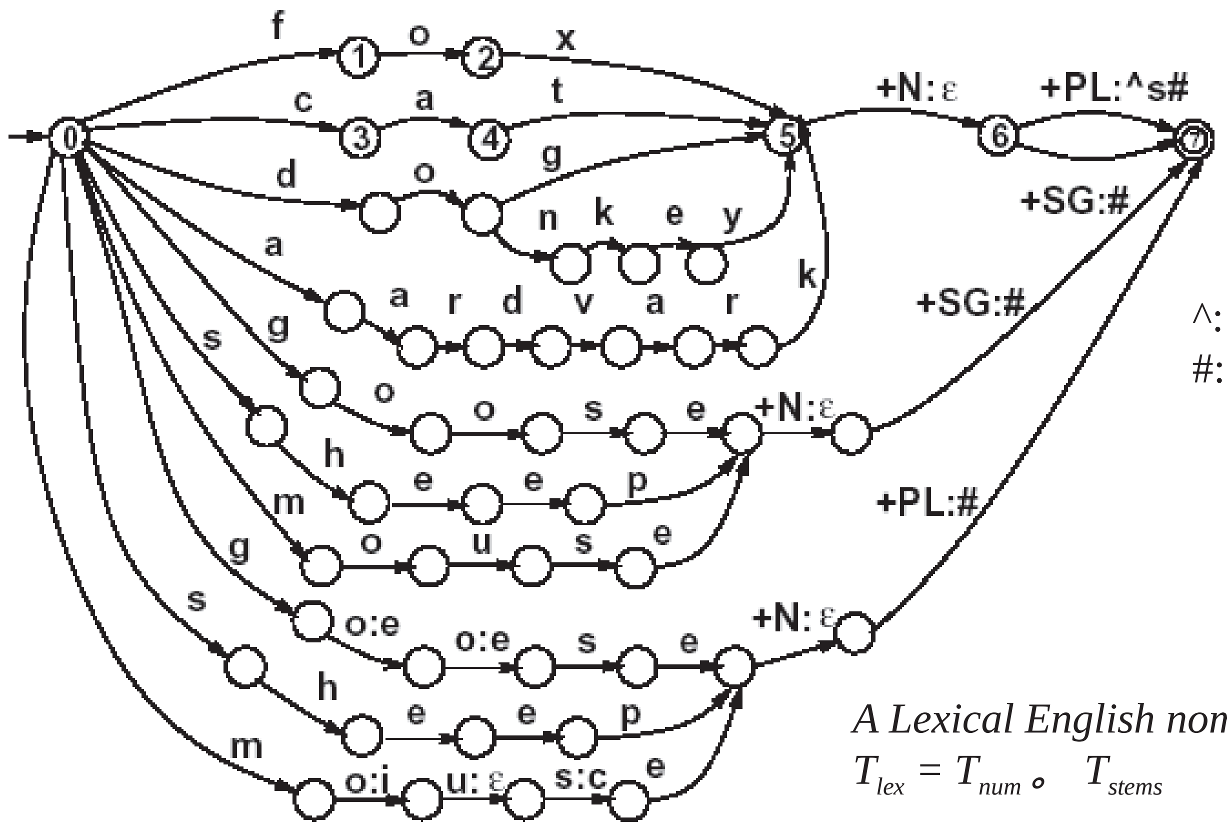
  –`c:c a:a t:t +N:epsilon  +PL:^s#`

  –*c* maps to itself, as do *a* and *t*, while morphological feature `+N` maps to nothing, and the feature `+PL` maps to ^s

  –The symbol ^ indicates *morpheme boundary*, while symbol # indicates a *word boundary*



## Morphological Parsing with FST



^: morpheme boundary
#: word boundary

*A Lexical English nominal inflection FST*

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

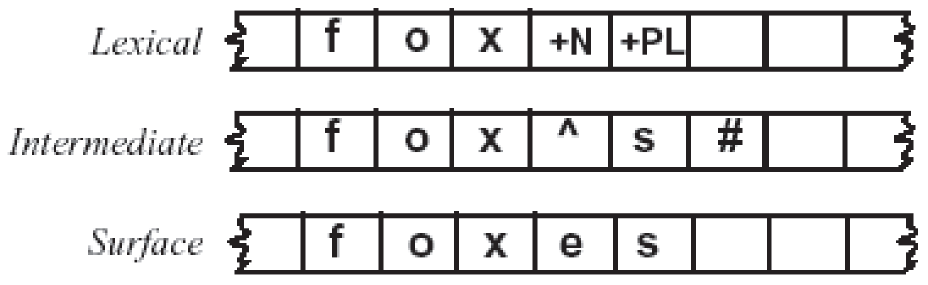
| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| *Lexical* | f | o | x | +N | +PL | | | |
| *Intermediate* | f | o | x | ^ | s | # | | |

## 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 |

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



## Orthographic Rules and FSTs

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

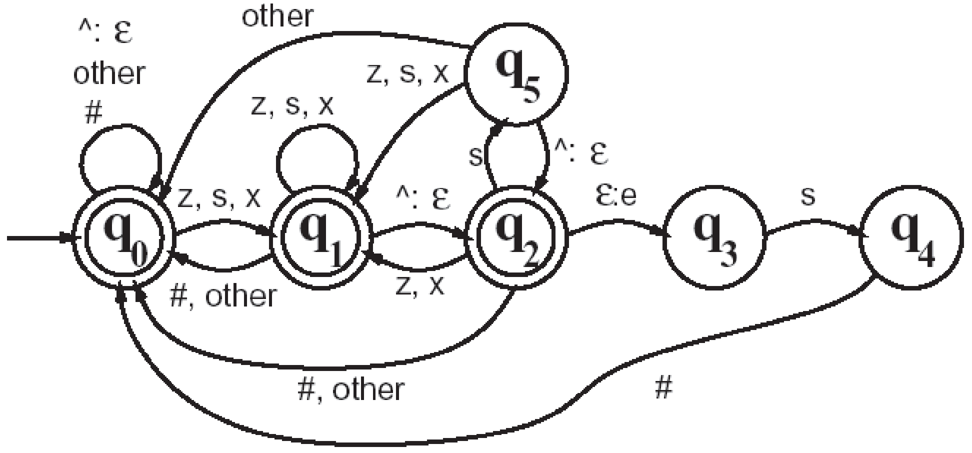
$$\varepsilon \rightarrow e \,/ \left\{ \begin{matrix} x \\ s \\ z \end{matrix} \right\} \wedge \_\_ s\#$$

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

  [Chomsky and Halle (1968)]

  $a \rightarrow b \,/\, c \,\_d$

## 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 |

## References

- *Speech and Language Processing*, [Chapter 3]
  Daniel Jurafsky and James H. Martin

THANK YOU