English Morphology

Morphological Processing and FSAs

Slides by James Martin, adapted by Diana Inkpen
for CSI 5386 @ uOttawa

English Morphology

- Morphology is the study of the ways that words are built up from smaller units called morphemes
  - The *minimal meaning-bearing* units in a language
- We can usefully divide morphemes into two classes
  - Stems: The core meaning-bearing units
  - Affixes: Bits and pieces that adhere to stems to change their meanings and grammatical functions

English Morphology

- We can further divide morphology up into two broad classes
  - Inflectional
  - Derivational

Word Classes

- By word class, we have in mind familiar notions like noun and verb
  - Also referred to as parts of speech and lexical categories
- We'll go into the gory details in Chapter 5
- Right now we're concerned with word classes because the way that stems and affixes combine is based to a large degree on the word class of the stem

Inflectional Morphology

- Inflectional morphology concerns the combination of stems and affixes where the resulting word....
  - Has *the same word class* as the original
  - And serves a grammatical/semantic purpose that is
    - *Different* from the original
    - But is nevertheless *transparently* related to the original
      - "walk" + "s" = "walks"

Inflection in English

- Nouns are simple
  - Markers for plural and possessive
- Verbs are only slightly more complex
  - Markers appropriate to the tense of the verb
- That’s pretty much it
  - Other languages can be quite a bit more complex
  - An implication of this is that hacks (approaches) that work in English will not work for many other languages

Regulars and Irregulars

- Things are complicated by the fact that some words misbehave (refuse to follow the rules)
  - Mouse/mice, goose/geese, ox/oxen
  - Go/went, fly/flew, catch/caught
- The terms *regular* and *irregular* are used to refer to words that follow the rules and those that don't

Regular and Irregular Verbs

- Regulars…
  - Walk, walks, walking, walked, walked
- Irregulars
  - Eat, eats, eating, ate, eaten
  - Catch, catches, catching, caught, caught
  - Cut, cuts, cutting, cut, cut

Inflectional Morphology

- So inflectional morphology in English is fairly straightforward
- But is somewhat complicated by the fact that are irregularities

Derivational Morphology

- Derivational morphology is the messy stuff that no one ever taught you
- In English it is characterized by
  - Quasi-systematicity
  - Irregular meaning change
  - Changes of word class

Derivational Examples

- Verbs and Adjectives to Nouns

-ation	computerize	computerization
-ee	appoint	appointee
-er	kill	killer
-ness	fuzzy	fuzziness

Derivational Examples

- Nouns and Verbs to Adjectives

-al	computation	computational
-able	embrace	embraceable
-less	clue	clueless

Example: *Compute*

- Many paths are possible…
- Start with compute
  - Computer -> computerize -> computerization
  - Computer -> computerize -> computerizable
- But not all paths/operations are equally good (allowable?)
  - Clue
    - Clue → clueless
    - Clue → ?clueful
    - Clue → *clueable

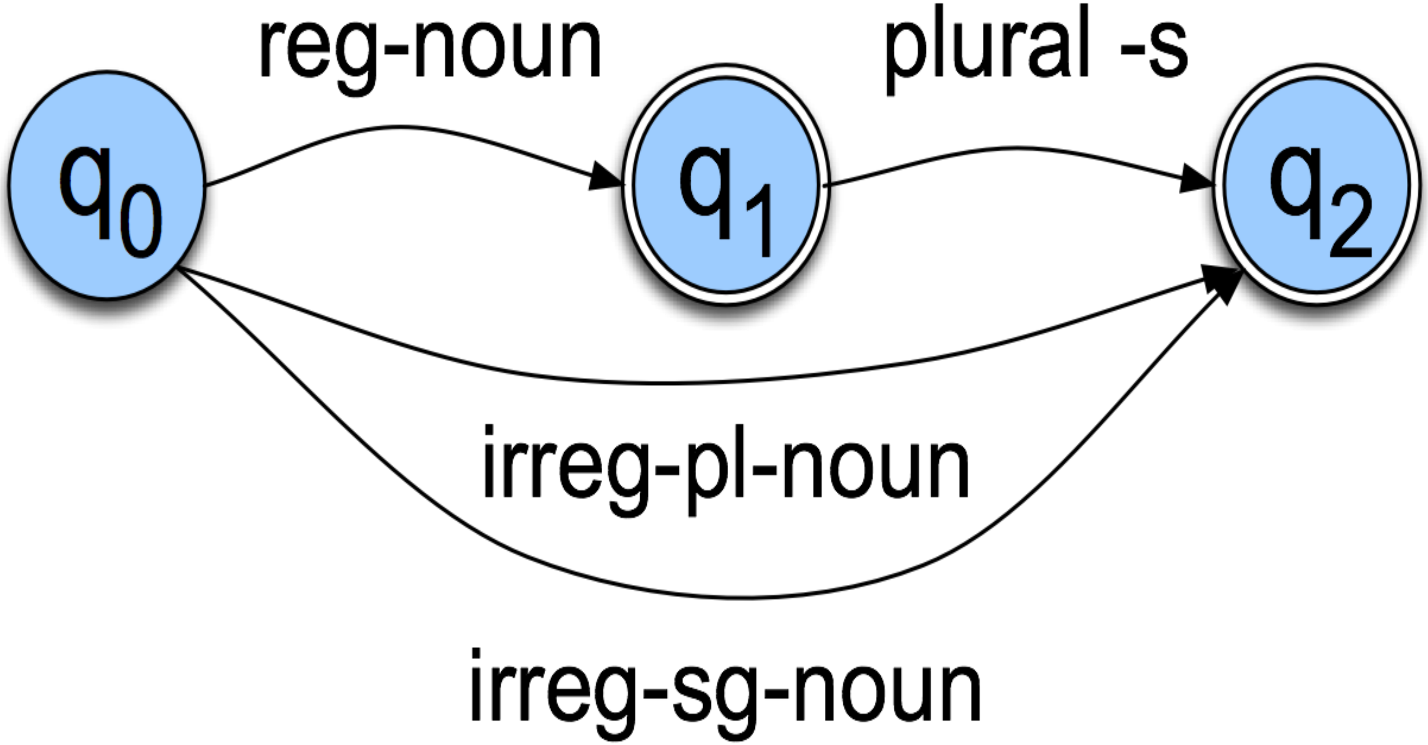
Morphology and FSAs

- We would like to use the machinery provided by FSAs to capture these facts about morphology
  - Accept strings that are in the language
  - Reject strings that are not
  - And do so in a way that doesn't require us to in effect list all the forms of all the words in the language
    - Even in English this is inefficient
    - And in other languages it is impossible

Start Simple

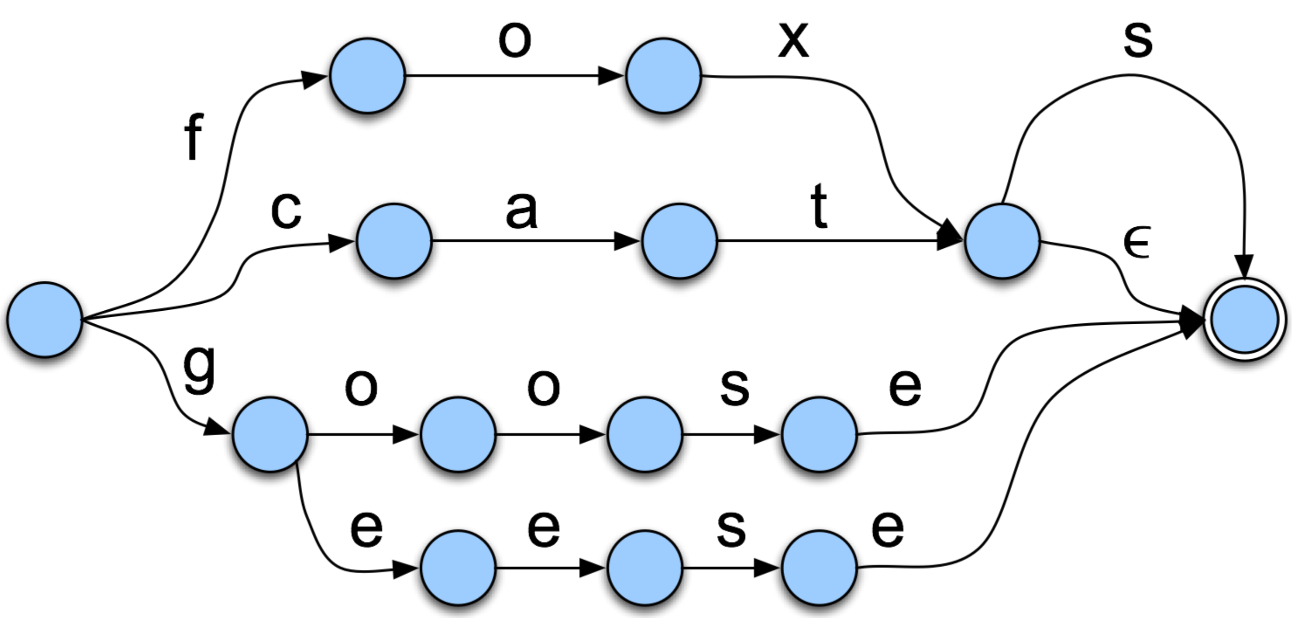
- Regular singular nouns are ok as is
  - They are in the language
- Regular plural nouns have an -s on the end
  - So they're also in the language
- Irregulars are ok as is

Simple Rules



Now Plug in the Words Spelled Out

Replace the class names like “reg-noun” with FSAs that recognize all the words in that class.



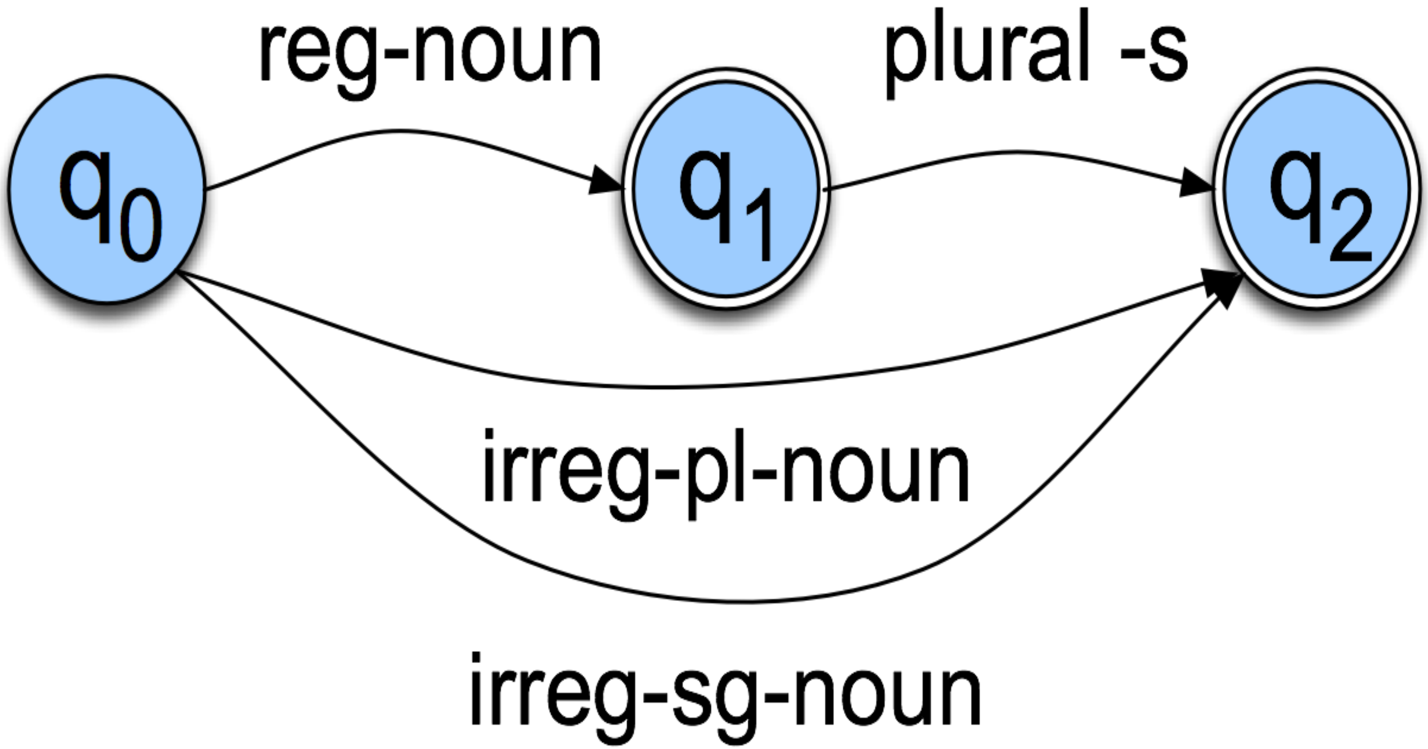
Morphology and FSAs

- We would like to use the machinery provided by FSAs to capture facts about morphology
  - Accept strings (words) that are legitimate words in the language
  - Reject those that are not
  - And do so in a way that doesn't require us to list all the forms of all the words in the language
    - Even in English this is inefficient
    - And in other languages it is impossible

Start Simple

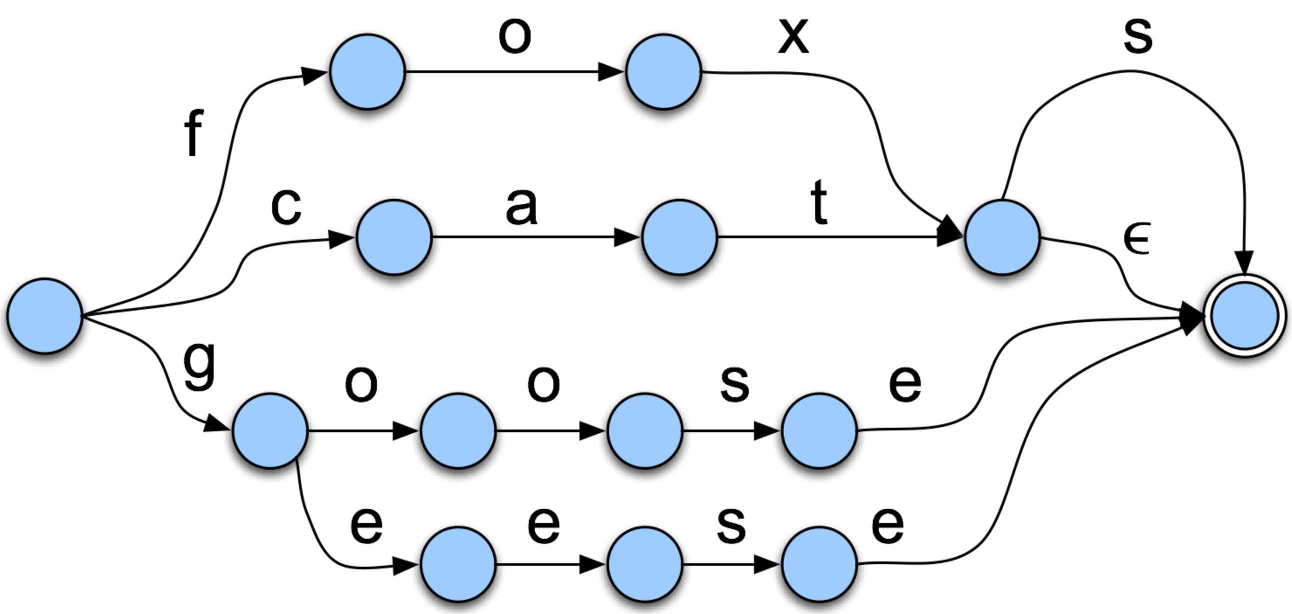
- Regular singular nouns are ok as is
  - They are in the language
- Regular plural nouns have an -s on the end
  - So they're also in the language
- Irregulars are ok as is
  - Irregulars with regular endings (-s) need to be blocks

Simple Rules

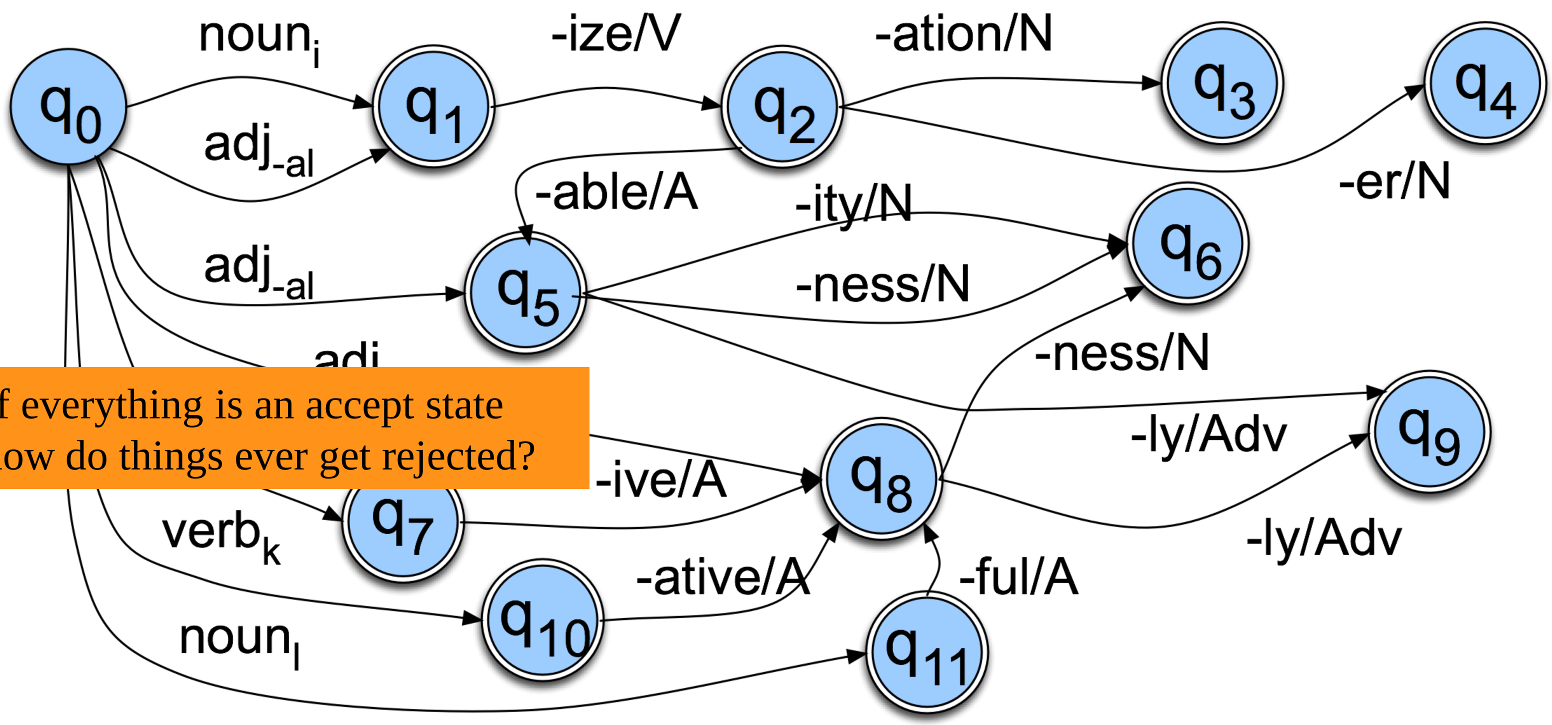


Now Plug in the Words Spelled Out

Replace the class names like “reg-noun” with FSAs that recognize all the words in that class.



Derivational Rules



If everything is an accept state
how do things ever get rejected?

Lexicons

- So the big picture is to store a lexicon (list of words you care about) as an FSA. The base lexicon is embedded in larger automata that captures the inflectional and derivational morphology of the language.
- So what? Well, the simplest thing you can do with such an FSA is *spell checking*
  - If the machine rejects, the word isn't in the language
  - Without listing every form of every word

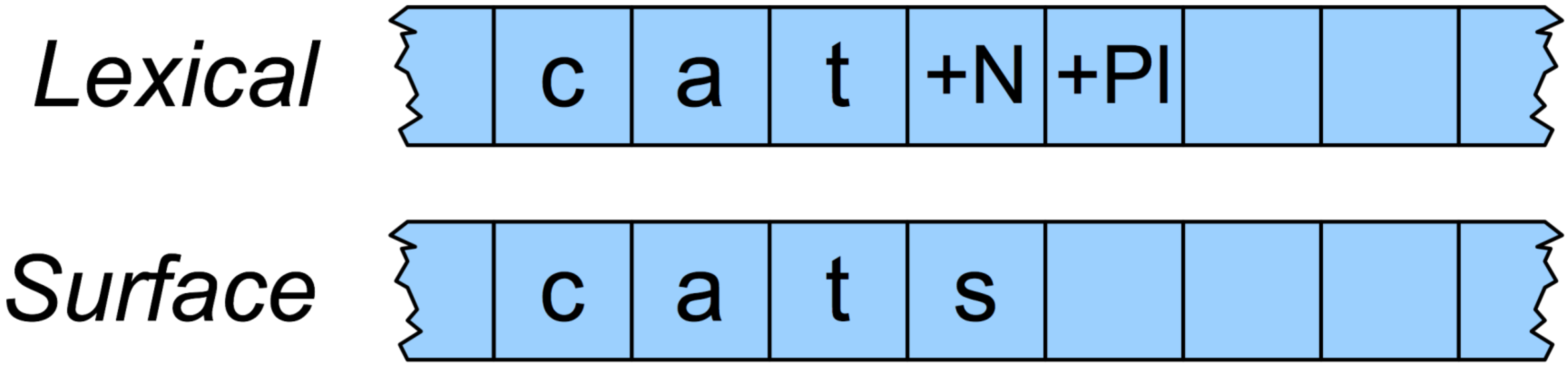
Parsing/Generation vs. Recognition

- We can now run strings through these machines to recognize strings in the language
- But recognition is usually not quite what we need
  - Often if we find some string in the language we might like to assign a structure to it (parsing)
  - Or we might start with some structure and want to produce a surface form for it (production/generation)
- For that we'll move to finite state transducers
  - Add a second tape that can be written to

Finite State Transducers

- The simple story
  - Add another tape
  - Add extra symbols to the transitions
  - On one tape we read "cats", on the other we write "cat +N +PL"
    - +N and +PL are elements in the alphabet for one tape that represent underlying linguistic features

FSTs



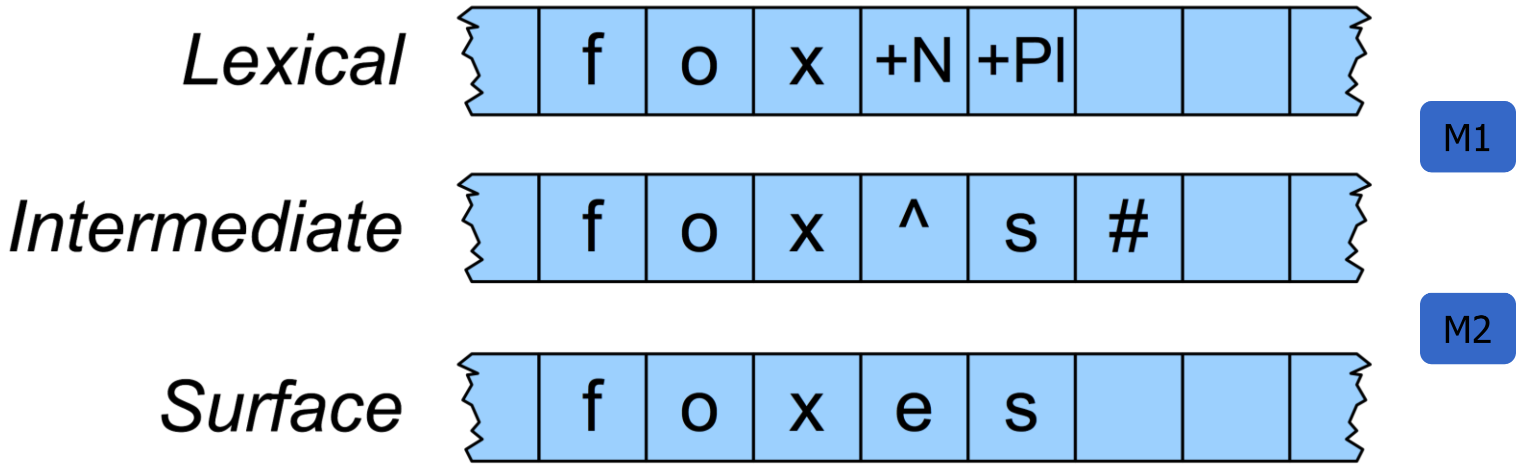
The Gory Details

- Of course, its not as easy as
  - “cat +N +PL” <-> “cats”
- As we saw earlier there are geese, mice and oxen
- But there are also a whole host of spelling/pronunciation changes that go along with inflectional changes
  - Cats vs Dogs (‘s’ sound vs. ‘z’ sound)□
  - Fox and Foxes (that ‘e’ got inserted)
    - And doubling consonants (swim, swimming)
    - adding k’s (picnic, picnicked)
    - deleting e’s,...

Multi-Tape Machines

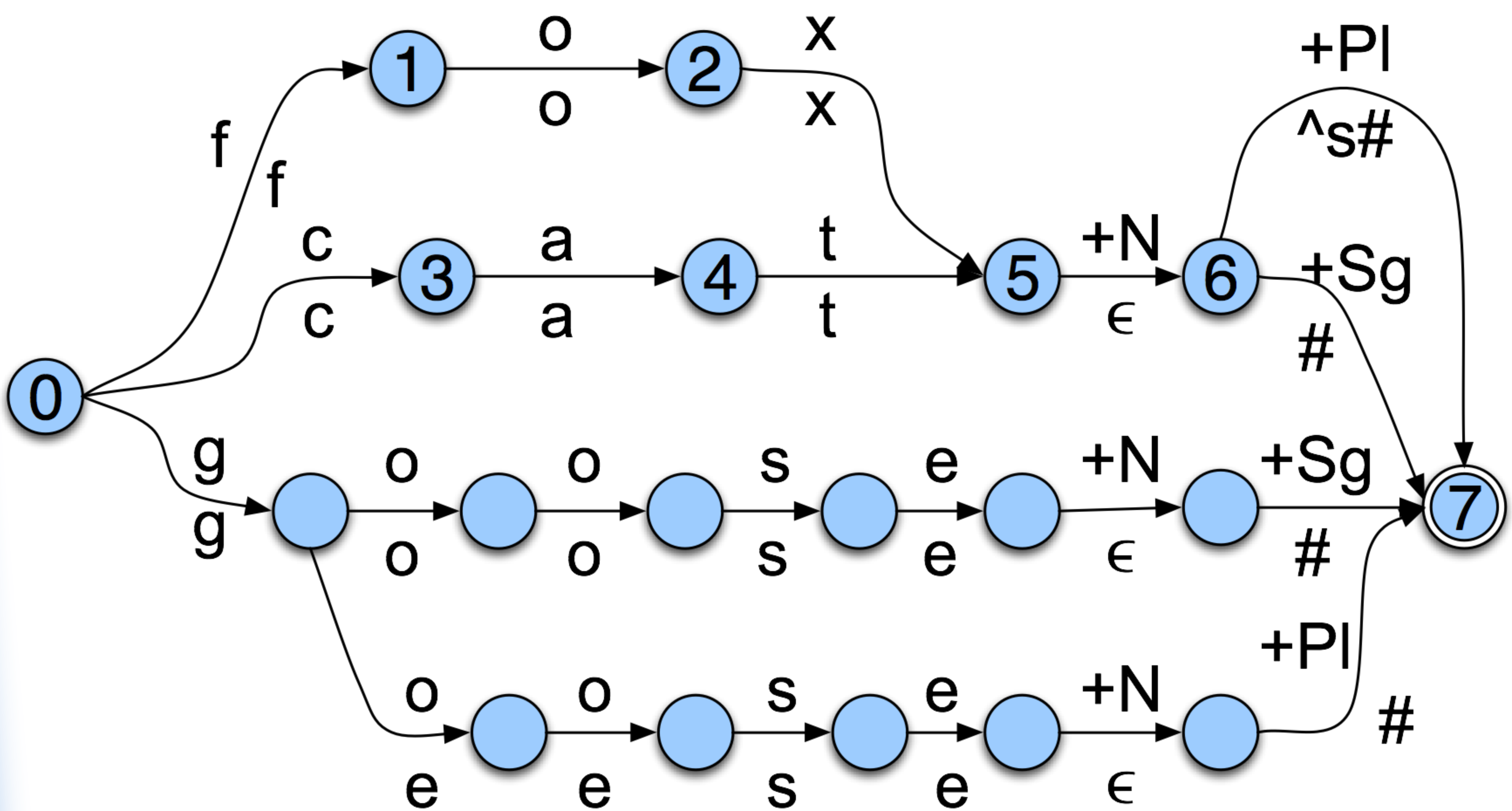
- To deal with these complications, we will add even more tapes and use the output of one tape machine as the input to the next
- So, to handle irregular spelling changes we will add intermediate tapes with intermediate symbols

Multi-Level Tape Machines



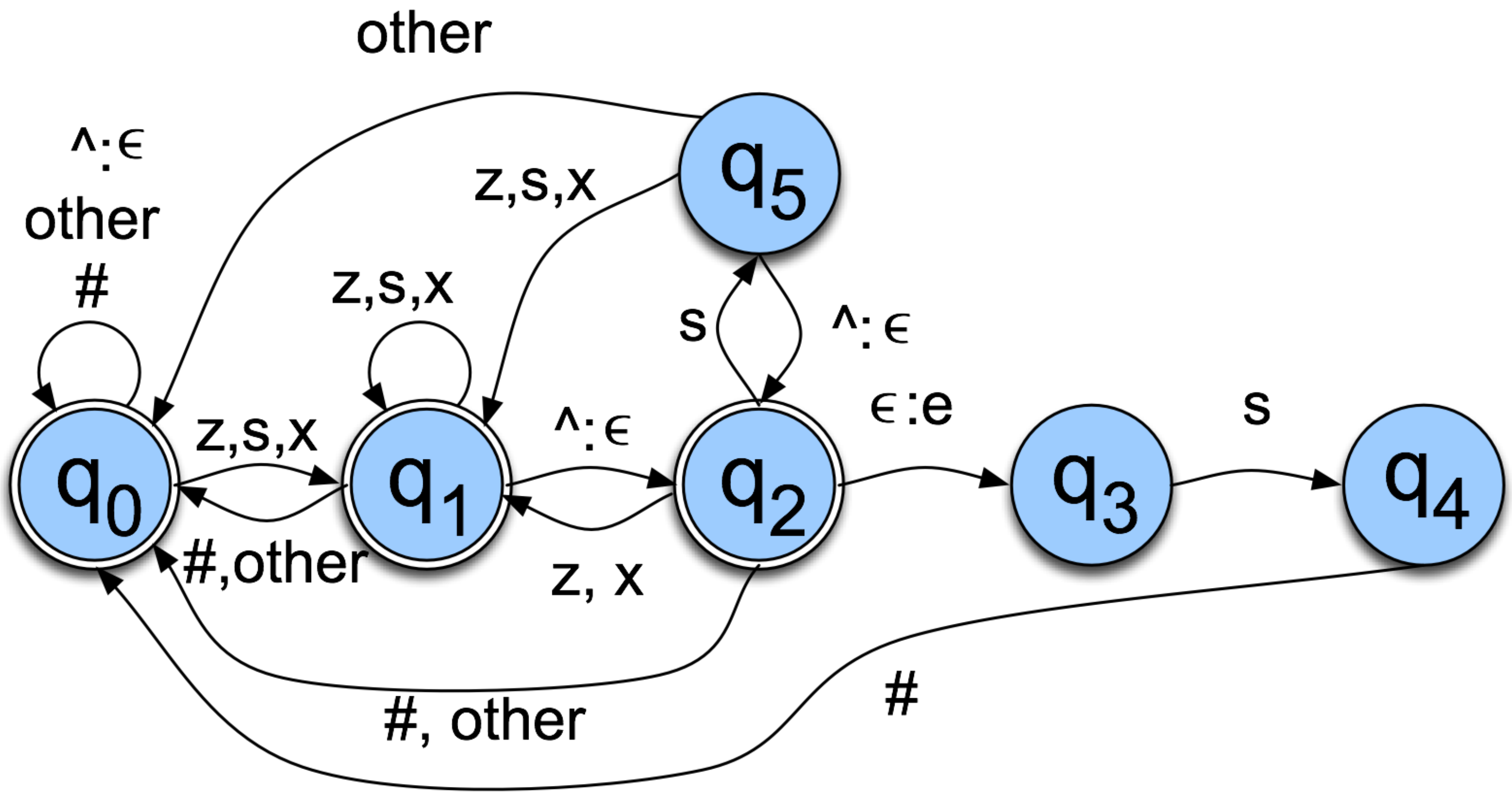
- We use one machine to transduce between the lexical and the intermediate level (M1), and another (M2) to handle the spelling changes to the surface tape
  - M1 knows about the particulars of the lexicon
  - M2 knows about weird English spelling rules

Lexical to Intermediate Level

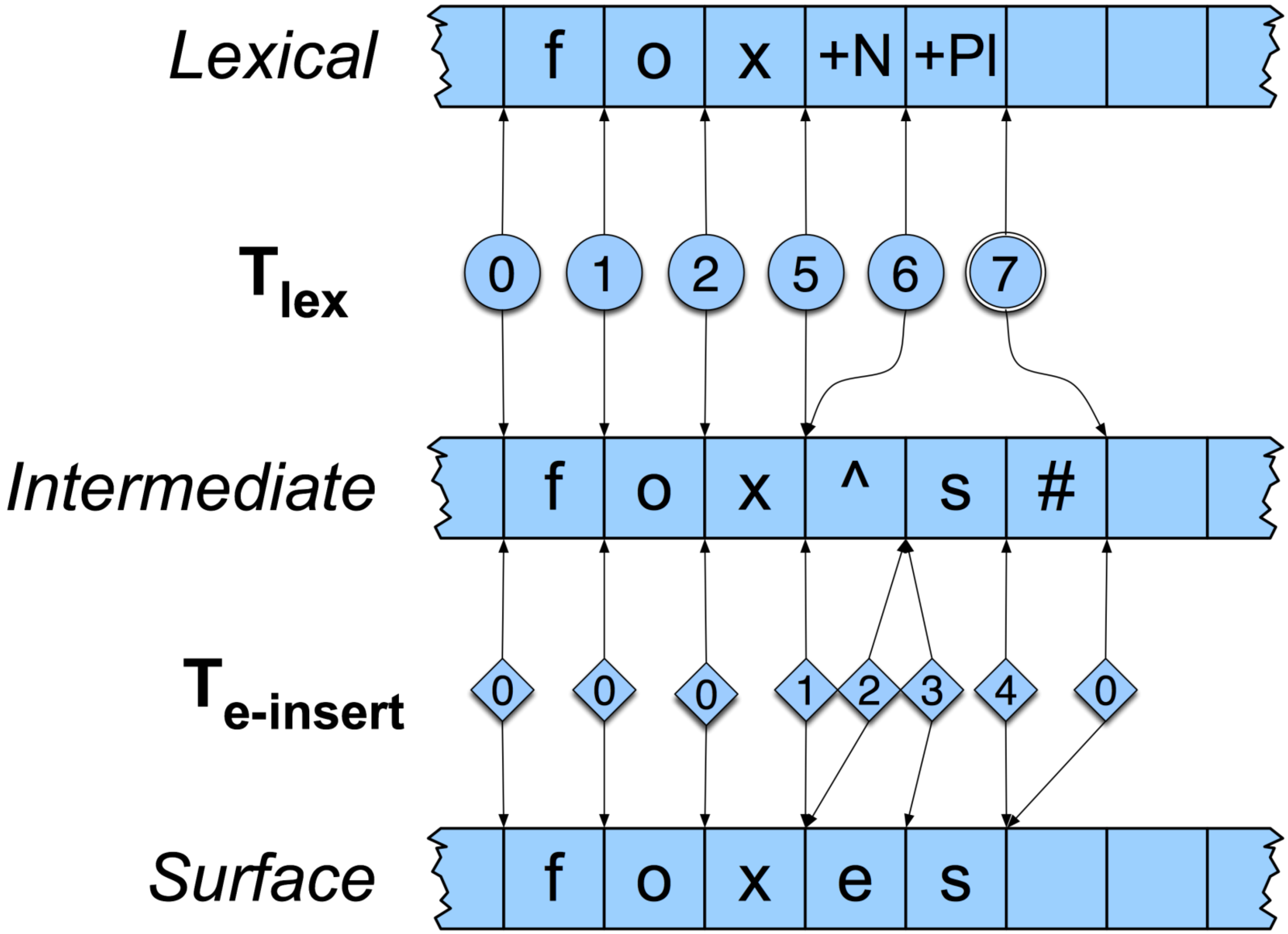


Intermediate to Surface

- The add an "e" English spelling rule as in fox^s# <-> foxes#

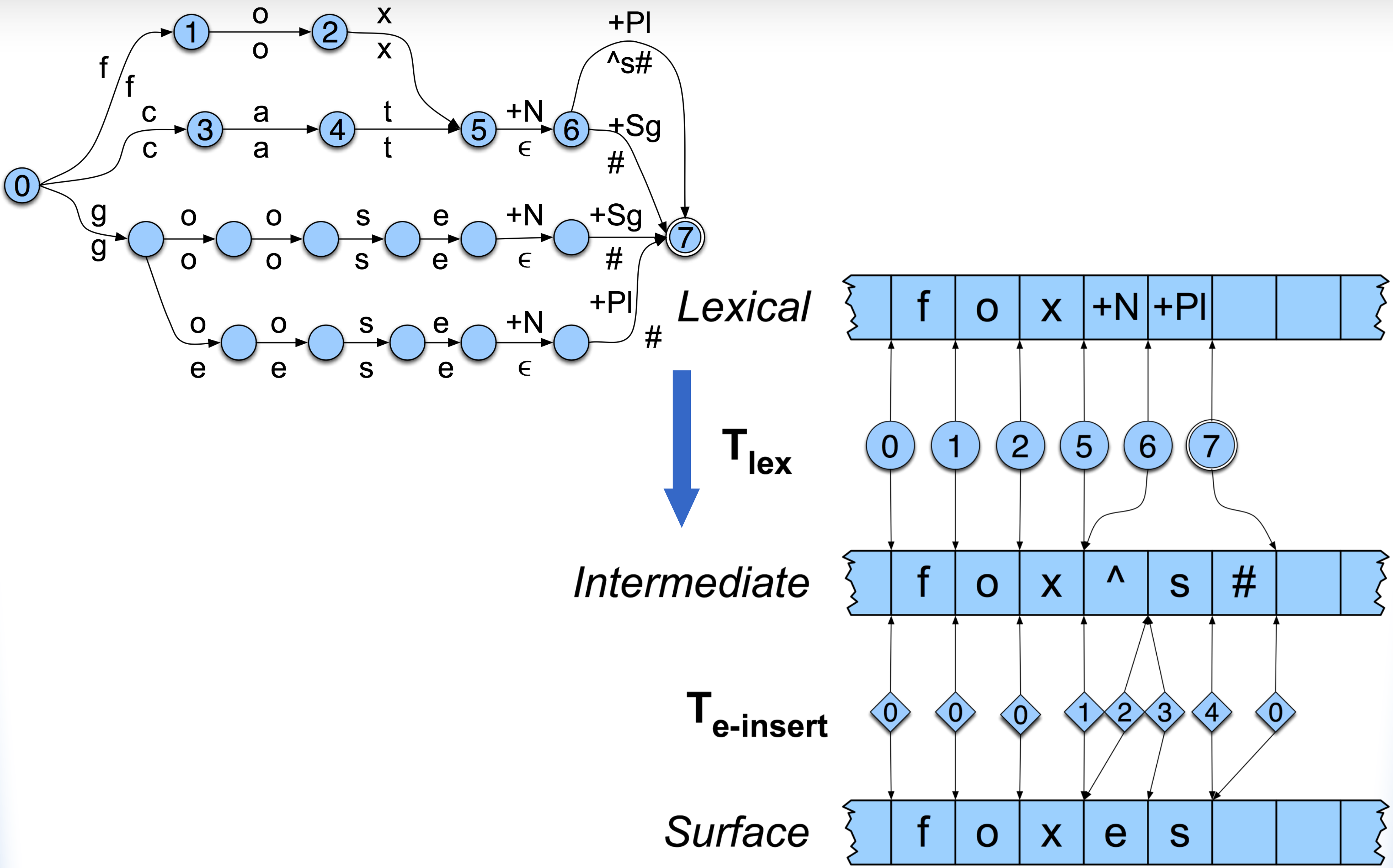


Foxes



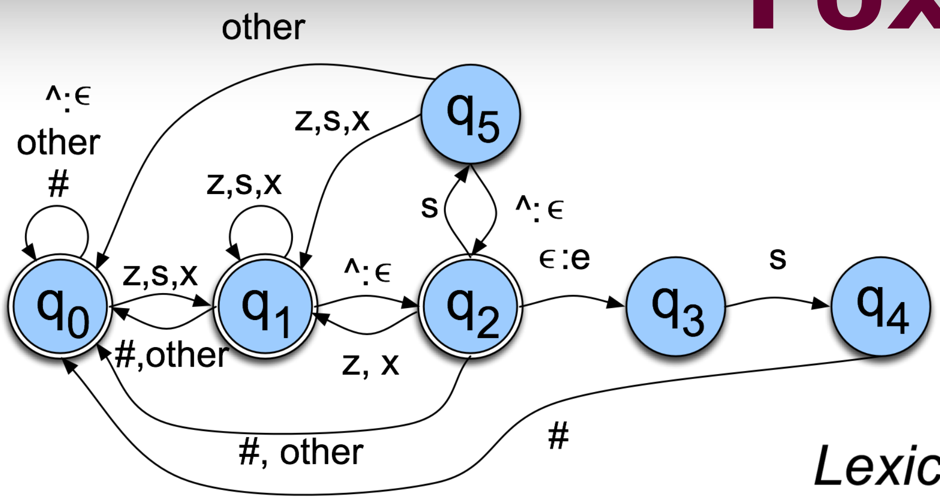
Lexical
f
o
x
+N
+Pl
Tlex
0
1
2
5
6
7
Intermediate
f
o
x
^
s
#
Te-insert
0
0
0
1
2
3
4
0
Surface
f
o
x
e
s

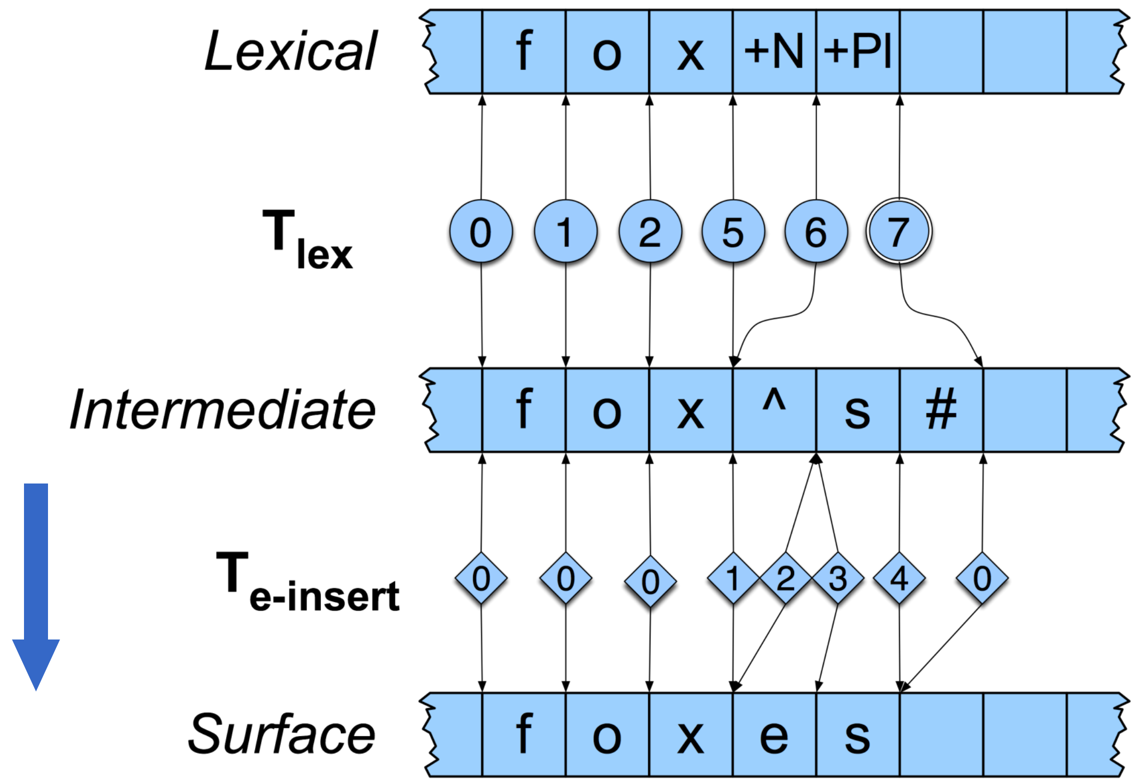
Foxes



0
1
2
3
4
5
6
7
f
f
o
o
x
x
c
c
a
a
t
t
+N
ϵ
+Pl
^s#
+Sg
#
g
g
o
o
s
s
e
e
Lexical
f o x +N +Pl
T lex
0 1 2 5 6 7
Intermediate
f o x ^ s #
T e-insert
0 0 0 1 2 3 4 0
Surface
f o x e s

Foxes


other
^:ϵ
other
#
z,s,x
z,s,x
s
^:ϵ
ϵ:e
s
z,s,x
^:ϵ
#,other
z, x
#, other
#
q0
q1
q2
q3
q4
q5


Lexical
f o x +N +Pl
T lex
0 1 2 5 6 7
Intermediate
f o x ^ s #
T e-insert
0 0 0 1 2 3 4 0
Surface
f o x e s

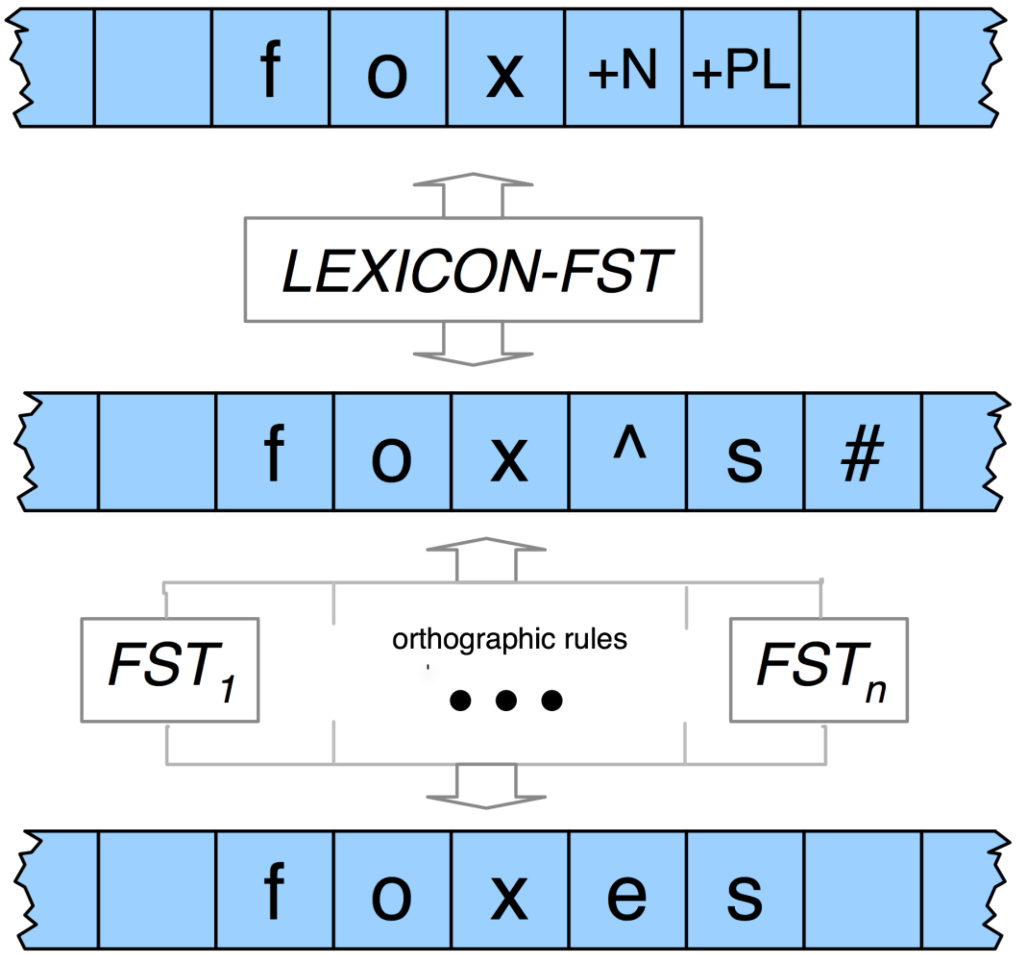
Note

- A key feature of this lower machine is that it has to do the right thing for inputs to which it doesn't apply. So…
  - fox^s# → foxes
  - but bird^s# → birds
  - and cat# → cat

Overall Scheme

- We now have one FST that has explicit information about the lexicon (actual words, their spelling, facts about word classes and regularity).
  - Lexical level to intermediate forms
- We have a larger set of machines that capture orthographic/spelling rules.
  - Intermediate forms to surface forms
  - One machine for each spelling rule

Overall Scheme

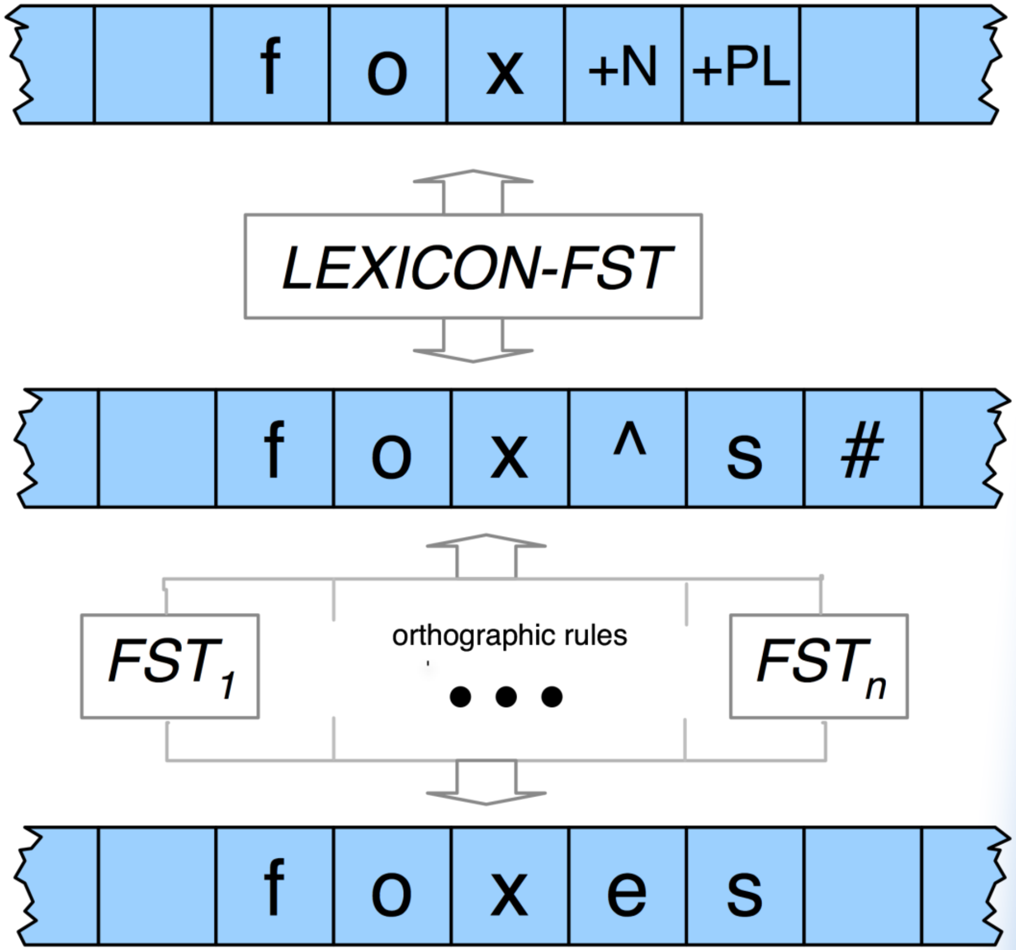


Separate FSTs for each of the English spelling rules we want to capture.

Cascades

- This is an architecture that we'll see again and again
  - Overall processing is divided up into distinct rewrite steps
  - The output of one layer serves as the input to the next
  - The intermediate tapes may or may not end up being useful in their own right

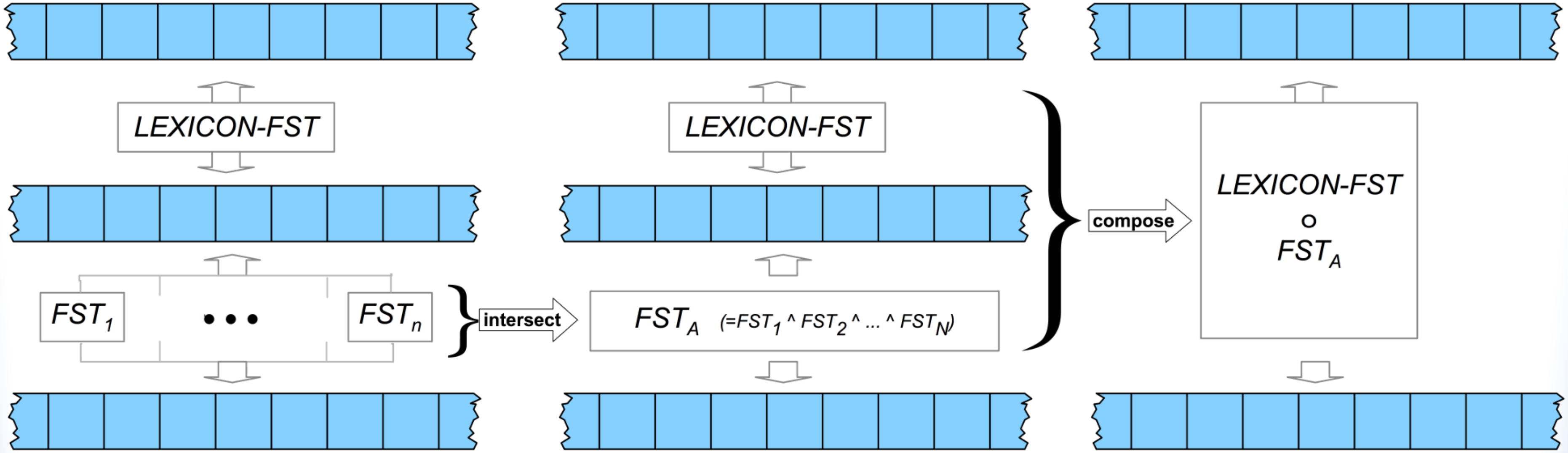
Overall Plan



Unfortunately, as an architecture this is a little unwieldy. We don't really want to mess with multiple tapes.

And we really don't want to mess with multiple machines reading and writing the same tapes in parallel.

Final Scheme



Intersecting FSTs

- Recall that for FSAs it was ok to take the intersection of two regular languages and have the result still be regular
  - *Regular languages are closed under intersection*
- There's no such guarantee for FSTs
  - *Regular relations are not closed under intersection in general*
  - *But interesting subsets are*

Composing FSTs

1. Create a set of new states that correspond to each pair of states from the original machines (New states are called {x,y}, where x is a state from M1, and y is a state from M2)
2. Create a new FST transition table for the new machine according to the following intuition…

Composition

- There should be a transition between two states in the new machine if it is the case that the output for a transition from a state from M1, is the same as the input to a transition from M2 or…

Then

- Once we've used composition to eliminate the intermediate tapes (machines), we can then determinize and minimize the resulting machine.
- Such minimized automata/transducers are used to represent large lexicons efficiently

Finally

- Now we can compose the fox^s machine with the e-insertion machine.
- That gives us a composed FST that in effect represents the path traversed by the input tape
- Then we can "project" to take only the output symbols from that composed machine... Giving us what we want.