Chapter 3
Finite State Morphological Parsing

April 4, 2011

Overview

- Morphology primer
- Using FSAs to recognize morphologically complex words
- FSTs (definition, cascading, composition)
- FSTs for morphological parsing
- Next time: More on FSTs, morphological analysis and an XFST demo

Morphology Primer

- Words consist of stems and affixes.
- Affixes may be prefixes, suffixes, circumfixes or infixes.
  Examples?
- (Also: root and pattern morphology)
  Examples?
- Phonological processes can sometimes apply to combinations of morphemes.

Phonology at Morpheme Boundaries

Examples:

English	/s/	Spanish	/s/
Singular	Plural	Singular	Plural
[kæt] ‘cat’	[kæts] ‘cats’	[ninjo] Eng: ‘boy’	[ninjos]
[dɑg] ‘dog’	[dɑgz] ‘dogs’		
[fɪnʧ] ‘finch’	[fɪnʧəz] ‘finches’	[kaɾakol] Eng: ‘snail’	[kaɾakoles]

More on Morphology

- Languages vary in the richness of their morphological systems.
- Languages also vary in the extent to which phonological processes apply at (and sometimes blur) morpheme boundaries.
- English has relatively little inflectional morphology, but fairly rich (if not perfectly productive) derivational morphology.
- Turkish has more than 200 billion word forms.

Questions

- More examples of complex morphemes?
- What underlying representations might we want?
- Why would we want to get to those underlying representations?
- How do things change when we consider orthographic rules rather than phonological rules?

Morphological Parsing

- Parsing: Producing a linguistic structure for an input.
- Examples of morphological parsing:
  - Separating words into stems/roots and affixes

    e.g., input: cats   parse output: cat +s
  - Labeling morphemes with category labels

    e.g., input: cats   parse output: cat +N +PL

    ate   eat +V +PAST

List to Model Lexicon

- What about using a large list as a Lexicon?

a, aardvark, …
… bake, baked, baker, bakery, bakes, baking, …
… cat, catatonic, cats, catapult, …
… dog, dogged, dogs, …
… familiar, familiarity, familiarize, family, …

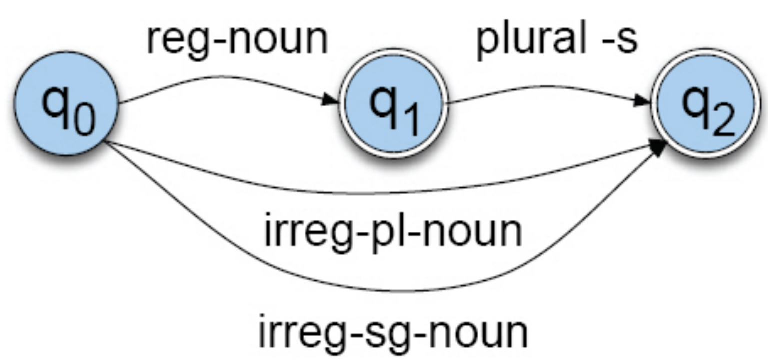
- Problem?

Using FSAs to recognize morphologically complex words

- Create FSAs for classes of word stems (word lists).
- Create FSA for affixes using word classes as stand-ins for the stem word lists.
- Concatenate FSAs for stems with FSAs for affixes.

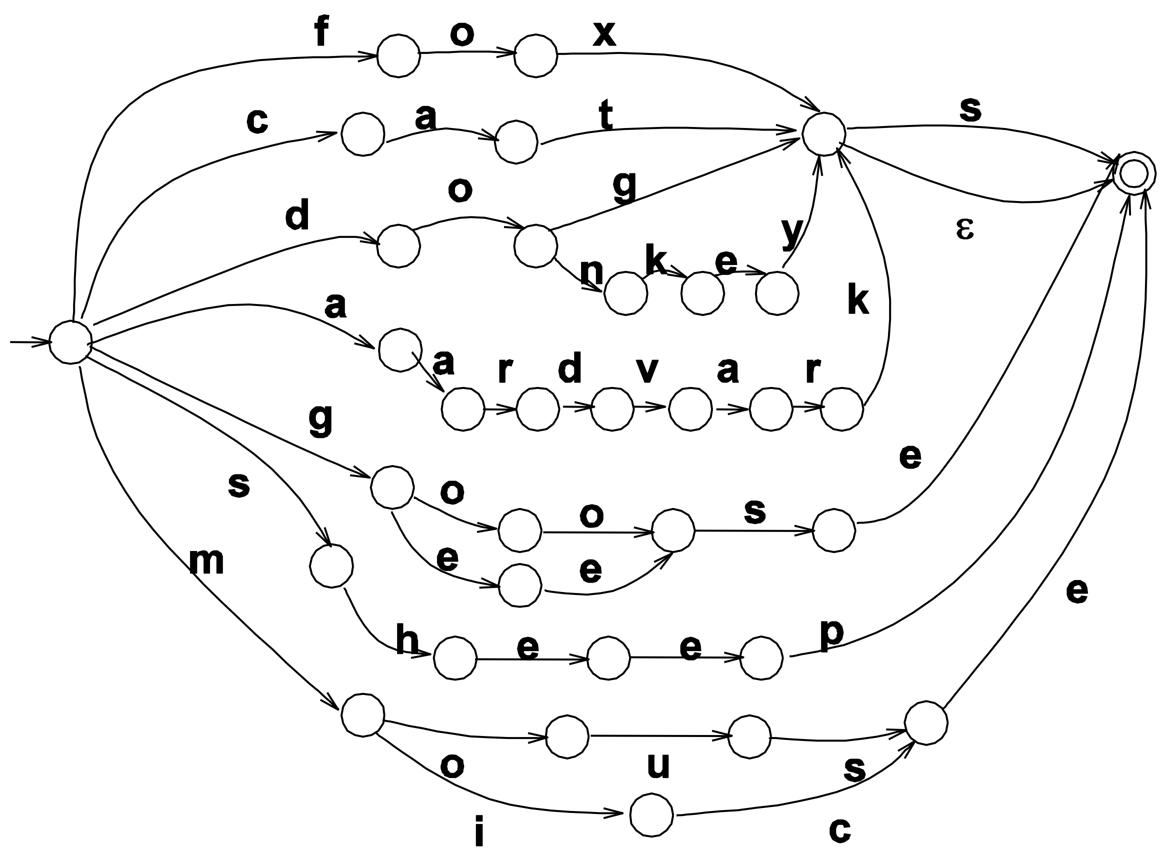
FSA Example using Word Classes

Defining morpheme selection and ordering for singular and plural English nouns:



J&M text, Fig 3.3

A variation with some words:



Note: Orthographic issues are not addressed.

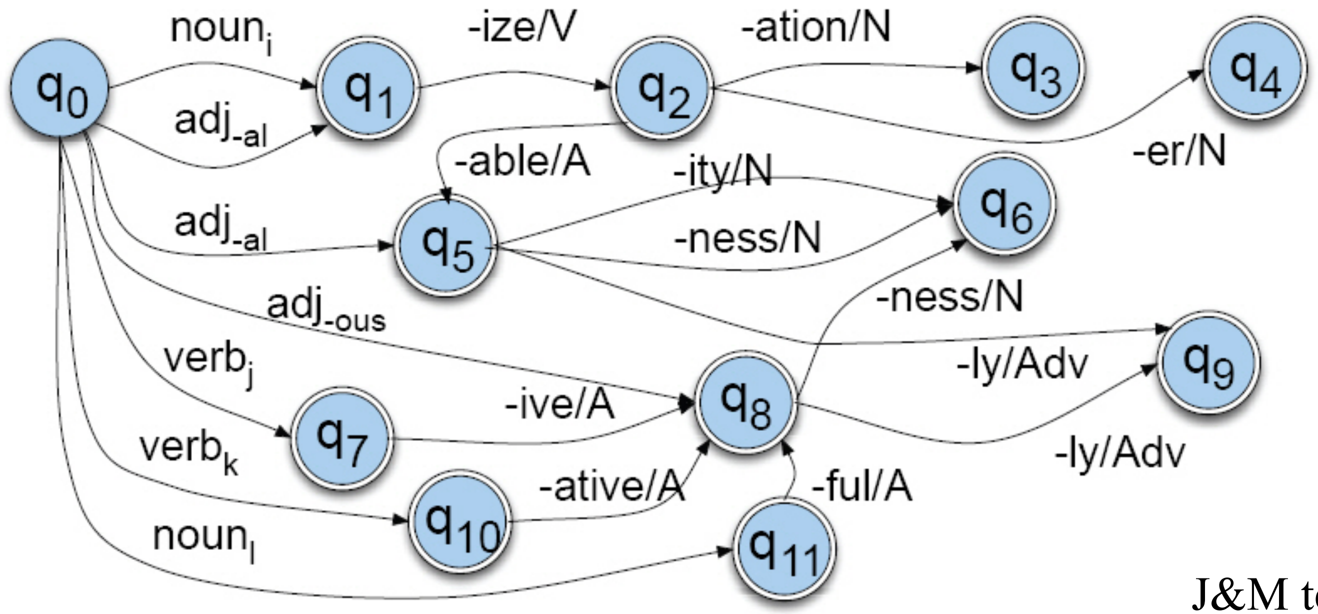
More Generalizations

… formal, formalize, formalization, …

… fossil, fossilize, fossilization, …

- These represent sets of related words.
- New forms are built with the addition of derivational morphology.
  - ADJ + -ity ➡ NOUN
  - ADJ or NOUN + -ize ➡ VERB

Derivational Rules



J&M text, Fig 3.7

Note: What string would this recognize? Is that really what we want?

Morphological Parsing

- A parsing task:
  - Recognize a string
  - Output information about the stem and affixes of the string
- Something like this:
  - Input: cats
  - Output: cat+N+PL
- We will use Finite-State Transducers to accomplish this.

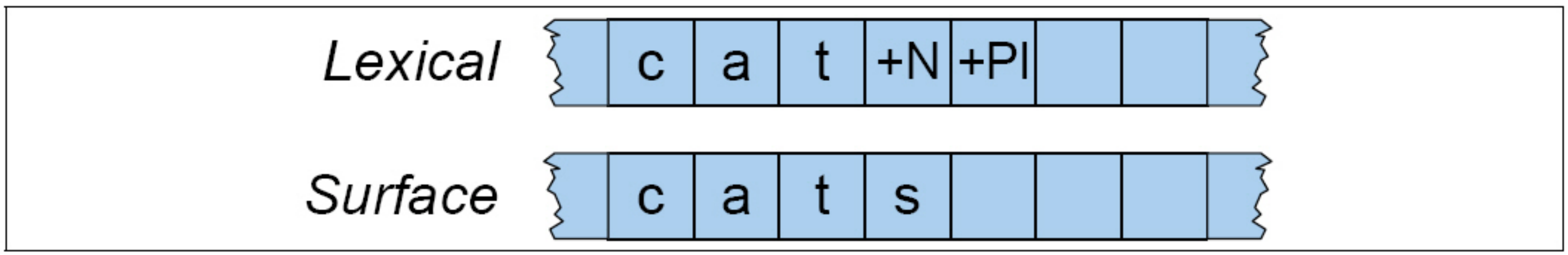
Finite-State Transducer (FST)

An FST: (see text pg 58 for formal definition)

- is like an FSA but defines *regular relations,* not regular languages
- has two alphabet sets
- has a transition function relating input to states
- has an output function relating state and input to output
- can be used to recognize, generate, translate or relate sets

Visualizing FTSs

- FSTs can be thought of as having an upper tape and a lower tape (output).



J&M text, Fig 3.12

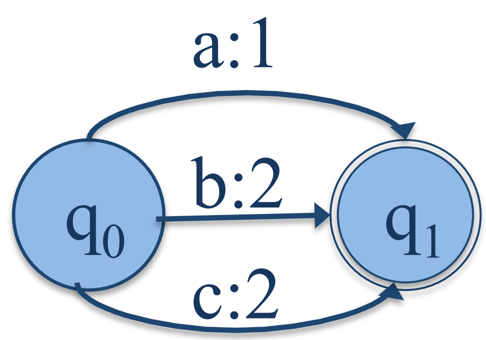
Regular Relations

- Regular language: a set of strings
- Regular relation: a set of pairs of strings
- E.g., Regular relation = {a:1, b:2, c:2}

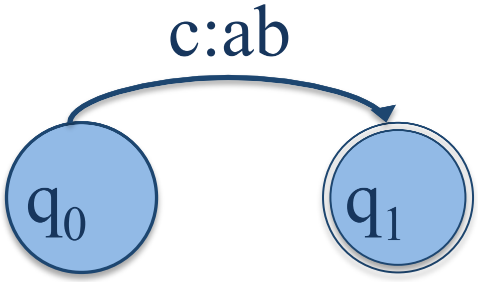
  Input Σ = {a,b,c}

  Output ={1, 2}

  FST:

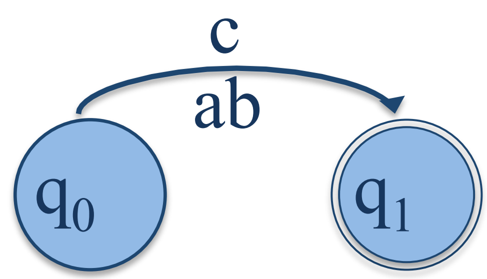


FST conventions

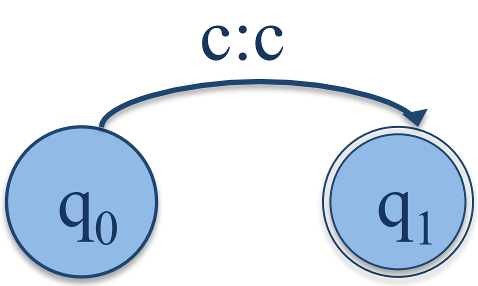


Complex input element

=

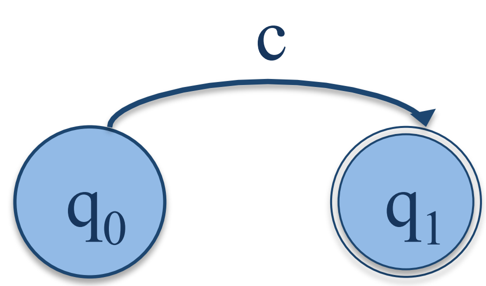


Divided into upper and lower

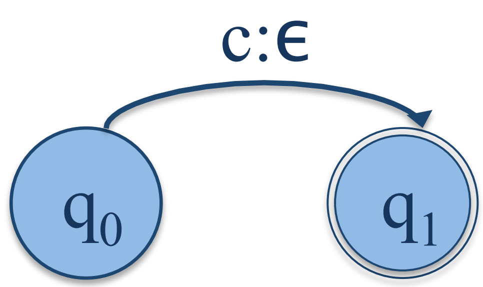


Default pair

=



Default pair - shortcut



c on upper, nothing on lower

FSTs: Not just fancy FSAs

- *Regular languages* are closed under difference, complementation and intersection; *regular relations* are (generally) not.
- *Regular languages* and *regular relations* are both closed under union.
- But *regular relations* are closed under composition and inversion; not defined for *regular languages*.

Inversion

- FSTs are closed under inversion, i.e., the inverse of an FST is an FST.
- Inversion just switches the input and output labels.

  e.g., if T_1 maps ‘a’ to ‘1’, then T_1^{-1} maps ‘1’ to ‘a’

- Consequently, an FST designed as a parser can easily be changed into a generator.

Composition

- It is possible to run input through multiple FSTs by using the output of one FST as the input of the next. This is called Cascading.
- Composing is equivalent in effect to Cascading but combines two FSTs and creates a new, more complex FST.
- $T_1 \circ T_2 = T_2(T_1(\mathrm{s}))$
  where s is the input string

Composition Example

- Very simple example:

  $T_1 = \{a{:}1\}$

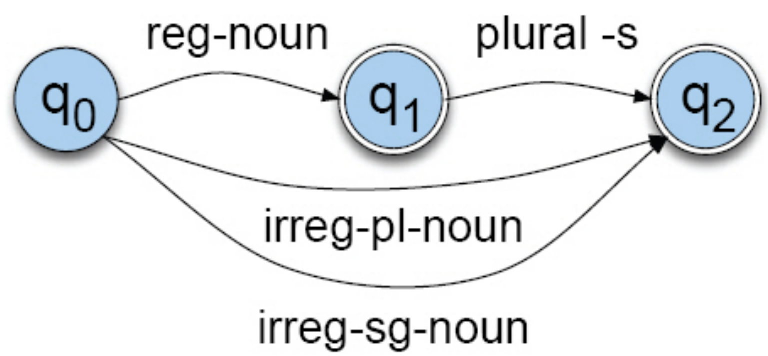
  $T_2 = \{1{:}one\}$

  $T_1 \circ T_2 = \{a{:}one\}$

  $T_2\,(T_1\,(a)\,) = one$
- Note that order matters: $T_1(T_2(a)) \neq$ one
- Composing will be useful for adding orthographic rules.

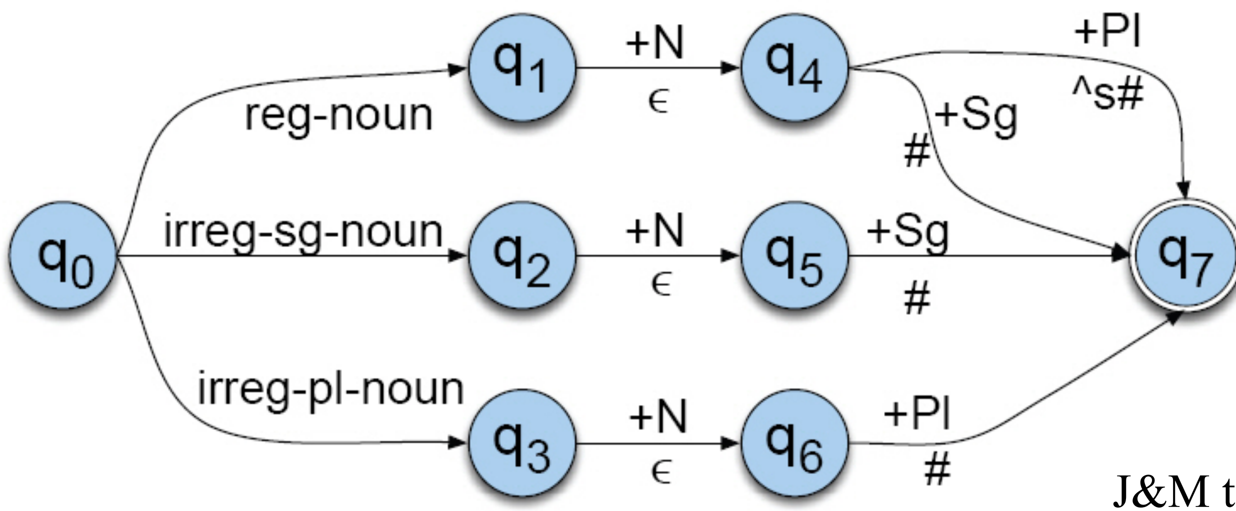
Comparing FSA Example with FST

Recall this FSA singular and plural recognizer:



J&M text, Fig 3.3

An FST to parse English Noun Number Inflection

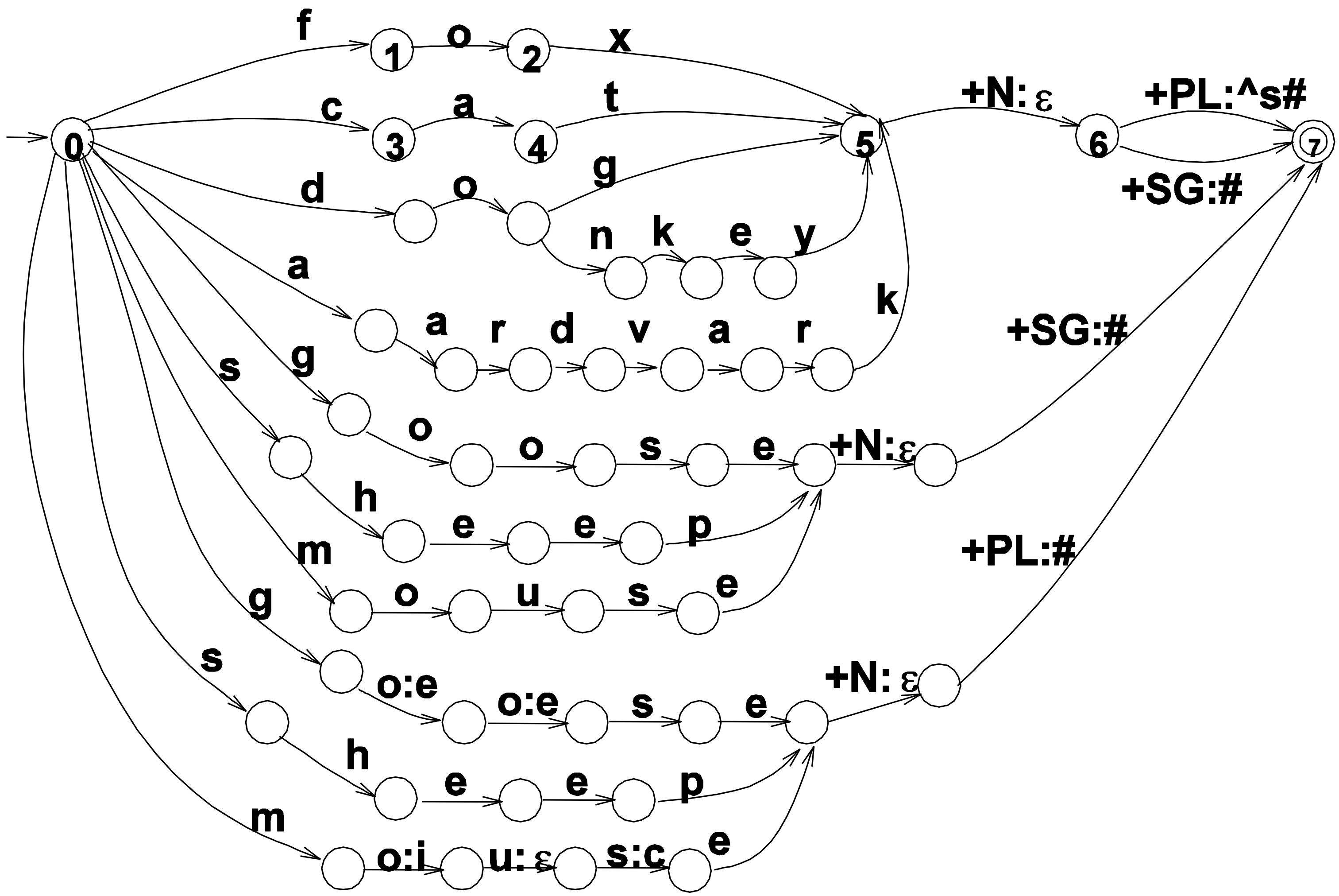


J&M text, Fig. 3.13

^ = morpheme boundary
# = word boundary

What are the benefits of this FST over the previous FSA? What is the input alphabet? What does the output look like?

Lexical to Intermediate Level

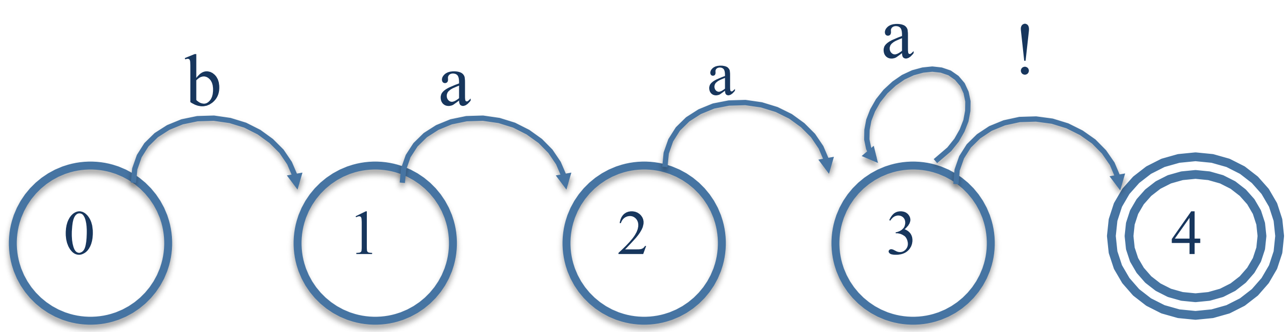


Overview

- Morphology primer
- Using FSAs to recognize morphologically complex words
- FSTs (definition, cascading, composition)
- FSTs for morphological parsing
- Next time: More on FSTs, morphological analysis and an XFST demo

Note: FSA as a generator

- Not only can an FSA be used as a recognizer – it can also generate. Back to sheep language:



•Begin at the start state (0).
•Emit each arc label.
•Output examples: baa! baaaa! baaaaaaaaaa!

Finite-State Transducers: Mealy machines

- Q: a finite set of states $q_0, q_1, \ldots, q_N$
- Σ: a finite alphabet of complex symbols $i : o$ such that $i \in I$ and $o \in O$. $\Sigma \subseteq I$ x O. I and O may each include ϵ.
- q_0: the start state
- F: the set of final states, $F \subseteq Q$
- $\delta(q, i : o)$: the transition matrix.

Regular Relations: Non-linguistic example

- Father-of relation: $\{\langle$ Larry, David $\rangle$, $\langle$ Ed, Cora $\rangle$, $\langle$ David, Henry $\rangle$, $\langle$ David, Simon $\rangle\}$
- Parent-of relation: $\{\langle$ Larry, David $\rangle$, $\langle$ Ed, Cora $\rangle$, $\langle$ David, Henry $\rangle$, $\langle$ David, Simon $\rangle$, $\langle$ Andrea, David $\rangle$, $\langle$ Geri, Cora $\rangle$, $\langle$ Cora, Henry $\rangle$, $\langle$ Cora, Simon $\rangle\}$
- Grandfather-of relation: $\{\langle$ Larry, Henry $\rangle$, $\langle$ Larry, Simon $\rangle$, $\langle$ Ed, Henry $\rangle$, $\langle$ Ed, Simon $\rangle\}$
- Paternal-grandfather-of relation: $\{\langle$ Larry, Henry $\rangle$, $\langle$ Larry, Simon $\rangle\}$