Ling/CSE 472: Introduction to Computational Linguistics

4/4/17: Morphology & FST 1

Announcements

- Assignment 1, part 2: Find a partner/have Canvas do it?
- Final project specs are up
- Questions to Canvas, rather than email, please
- More questions!

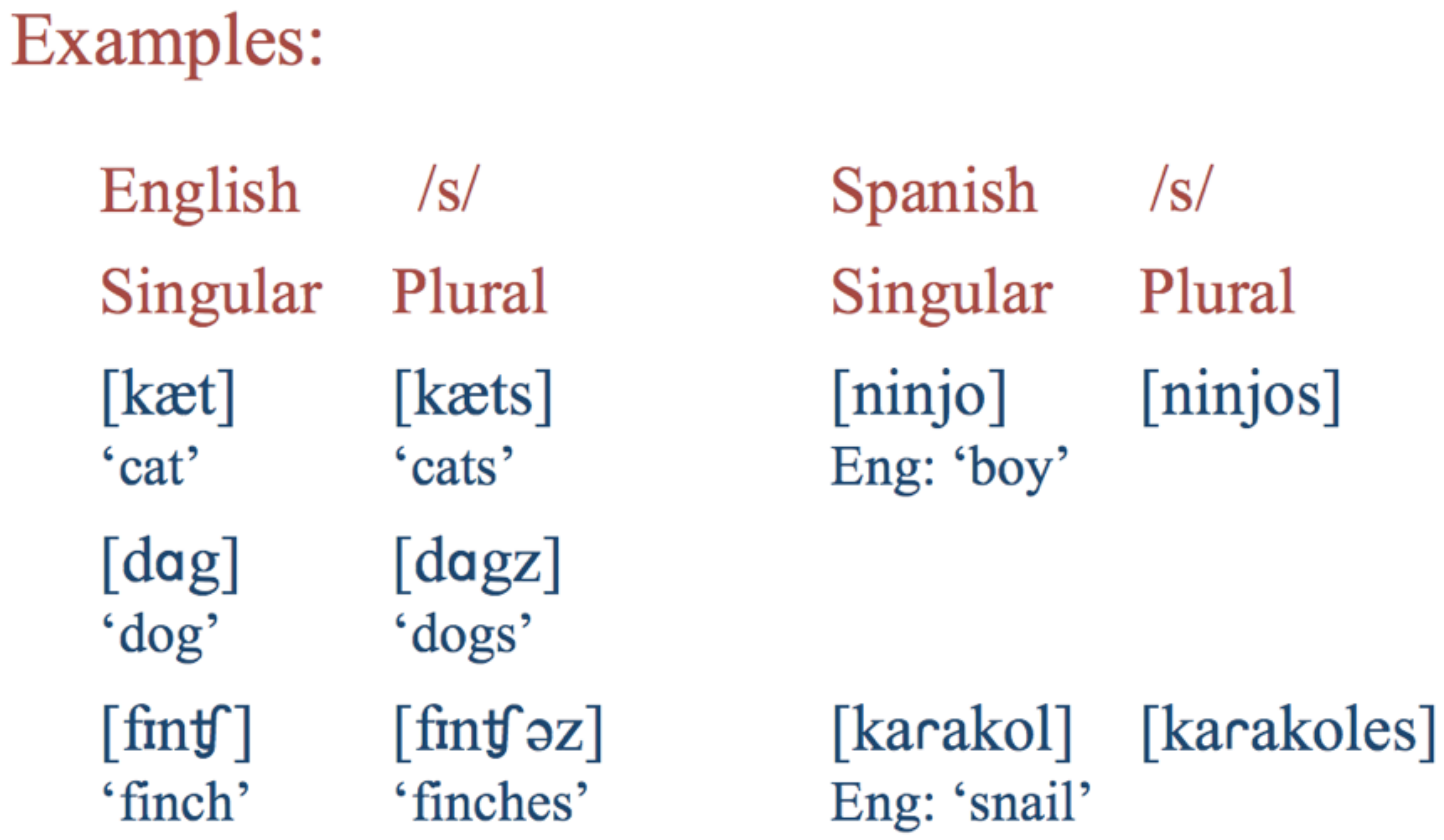
Overview

- Morphology primer
- Using FSAs to recognize morphologically complex words
- FSTs (definition, cascading, composition)
- FSTs for morphological parsing
- Reading questions
- Next time: More on FSTs, morphological analysis, 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’		
[finʧ] ‘finch’	[finʧəz] ‘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

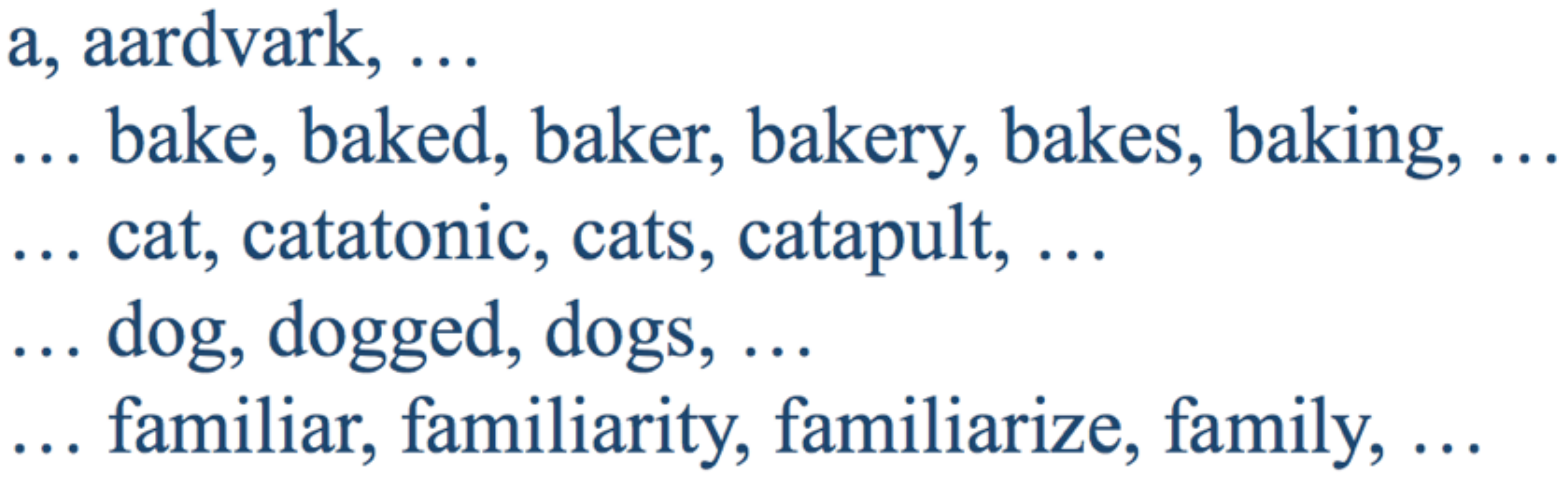
- More examples of complexity in morphology?
- What underlying representations might we want?
- Why would we want to get to those underlying representations?
- How do things change when we consider orthographic rules rather than phonological rules?

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
    - input: ate parse output: eat +V +PAST

Can we just model a lexicon as a list?

- 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, …

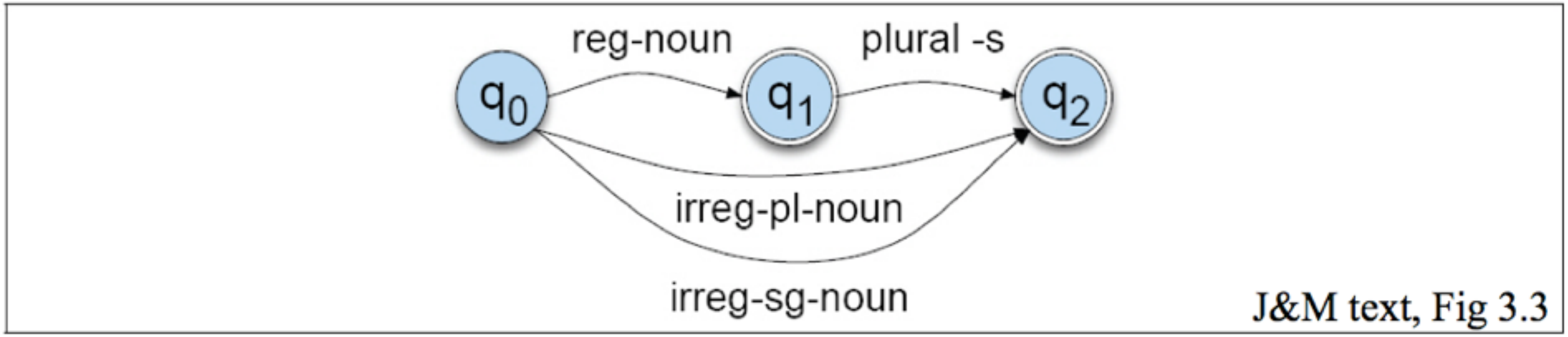
- Problem?

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.

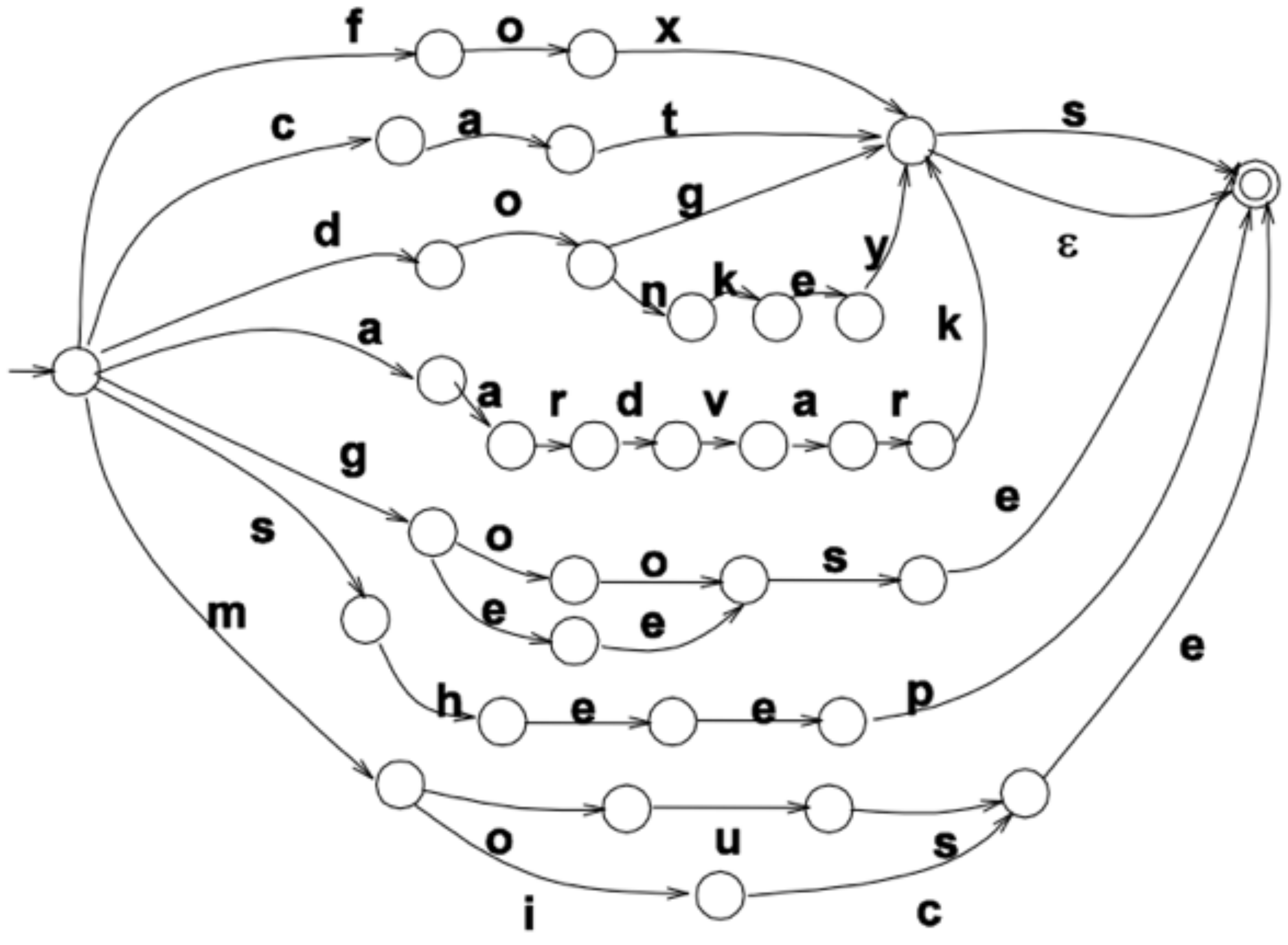
FSA Example using Word Classes

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



J&M text, Fig 3.3

A variation with some words

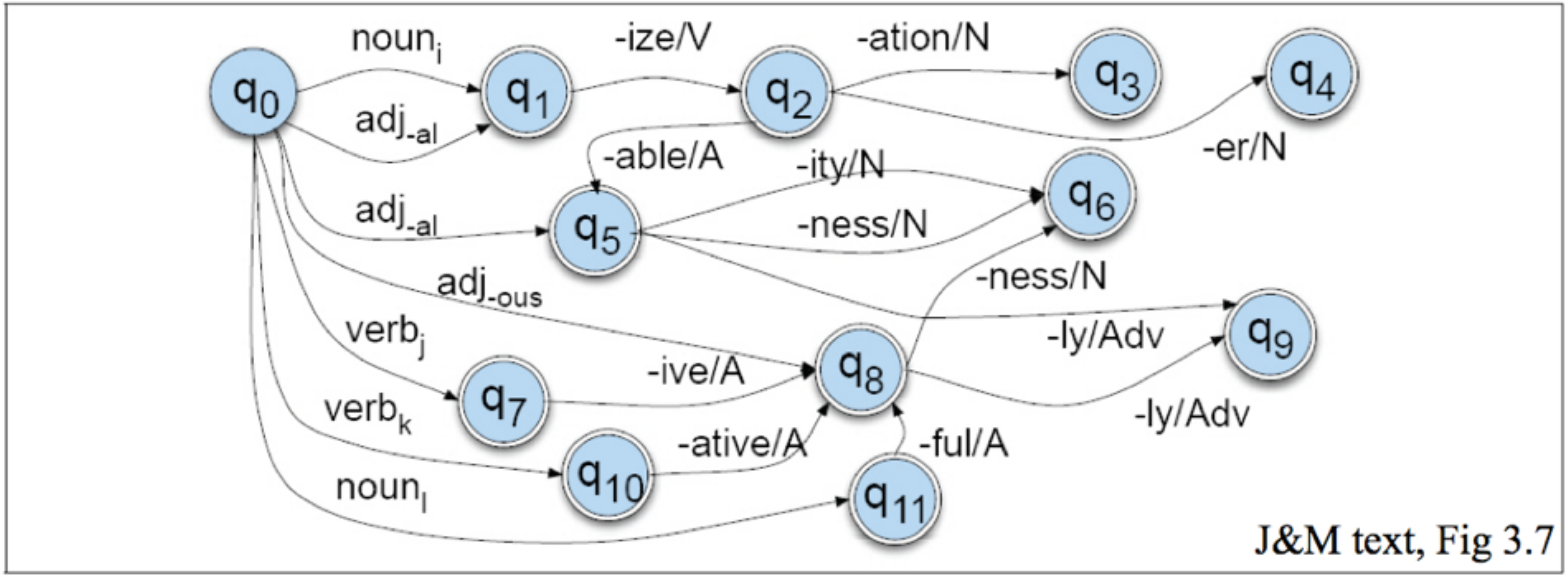


Note: Orthographic issues are not addressed.

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

- What strings would this recognize? Is that really what we want?

Morphological Parsing

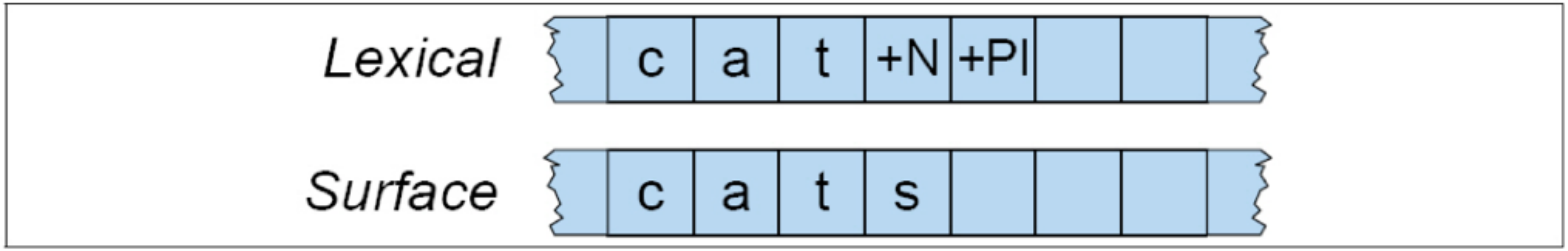
- A parsing task:
    - – Recognize a string
    - – Output information about the stem and affixes of the string
- Something like this:
    - – Input: cats
    - – Output: cat+N+PL
- We will use Finite-State Transducers to accomplish this.

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 FSTs

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

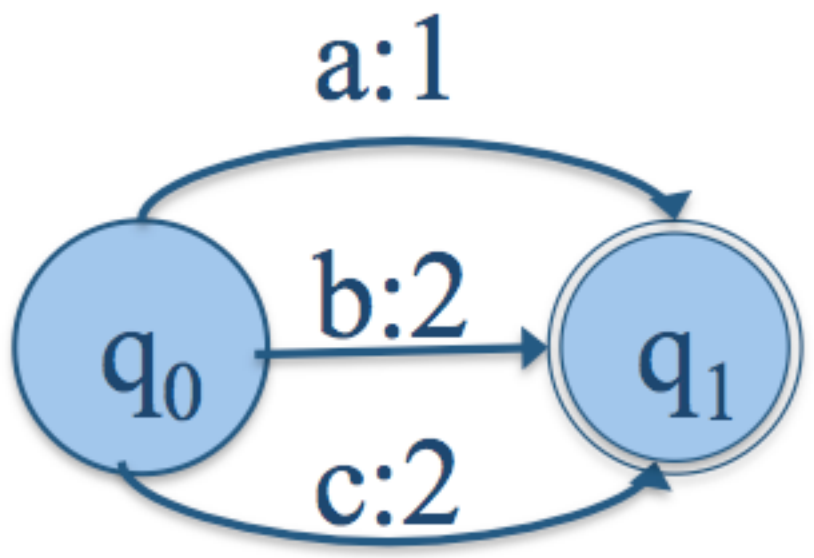


J&M text, Fig 3.12

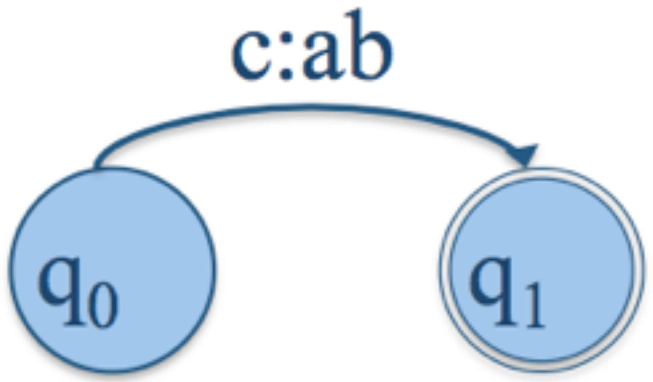
Regular Relations

- 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:

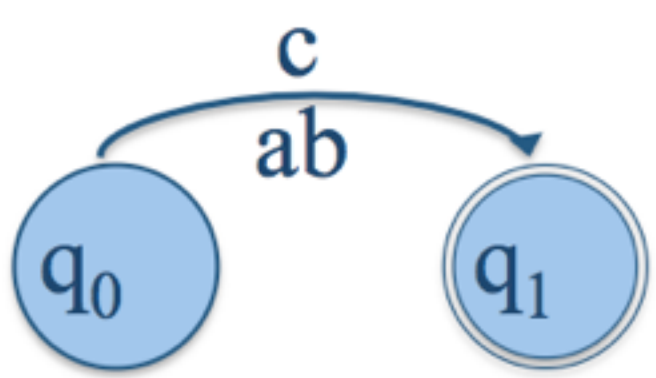


FST conventions

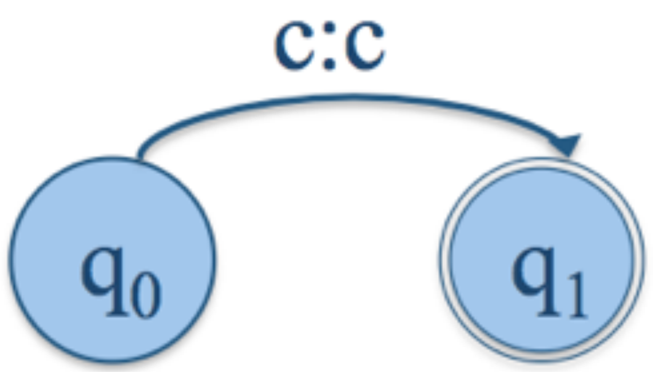


Complex input element

=

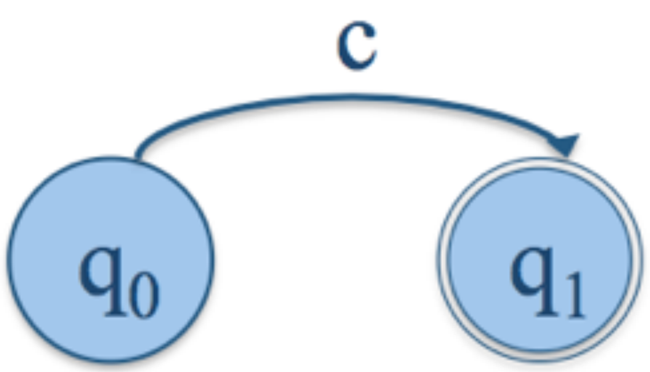


Divided into upper and lower

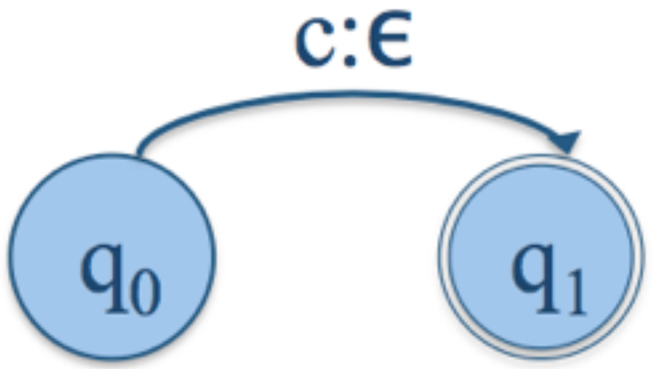


Default pair

=



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 T1 maps ‘a’ to ‘1’, then T1-1 maps ‘1’ to ‘a’
- Consequently, an FST designed as a parser can easily be changed into a generator.

Composition

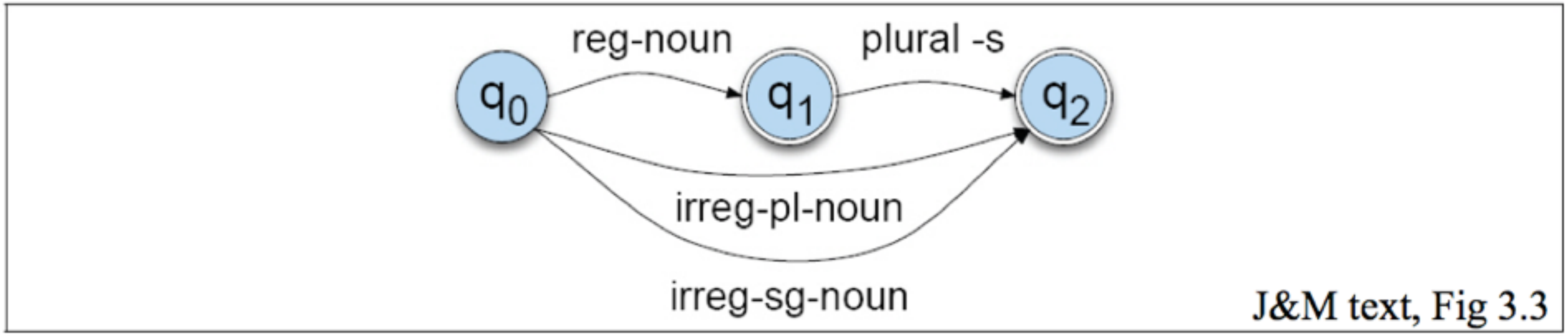
- It is possible to run input through multiple FSTs by using the output of one FST as the input of the next. This is called Cascading.
- Composing is equivalent in effect to Cascading but combines two FSTs and creates a new, more complex FST.
- $T1 \circ T2 = T2\ (T1(s))$
    - where s is the input string

Composition Example

- Very simple example:
    - T1 = {a:1}
    - T2 = {1:one}
    - T1 ◦ T2 = {a:one}
    - T2(T1(a)) =one
- Note that order matters: T1(T2(a)) ≠ one
- Composing will be useful for adding orthographic rules.

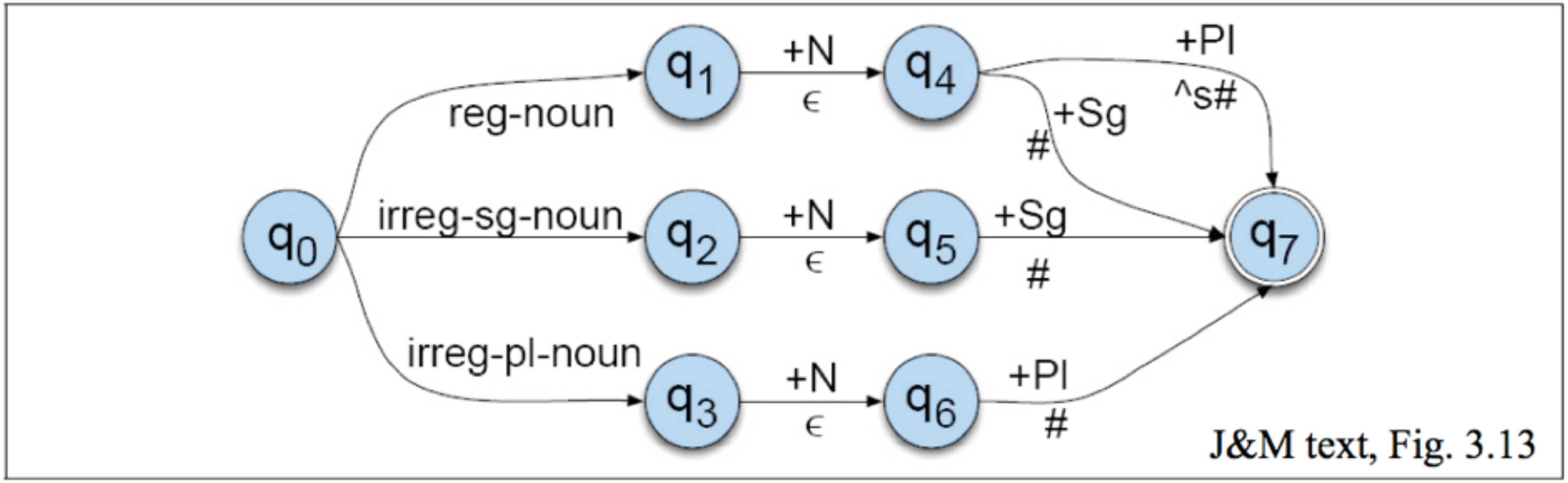
Comparing FSA Example with FST

- 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?

Overview

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

Reading Questions

- What is the exact relationship between set and FSA? How are the set operations (union, intersection, etc.) applicable to FSAs?

- The example of "work + s, ed, ing" for concatenation is helpful. But what are some usages of union, intersection and subtraction in the set theory? They are more or less just maths operations to me...

Reading Questions

- What are one-level language and two-level transducer? What does the level refer to?
- Projection: What is meant by the word "extract" one side from a given relation? Re-labeling them produces what exactly?

Reading Questions

- What are the drawbacks/limitations to finite-state networks?
- Can I understand finite state network as a special kind of directed graph?

Reading Questions

- on p34-35 what are the relations between "underlying, intermediate, surface strings" and FST? I am not sure I understand "to arrive at a single lexical transducer is to use the two-level rule formalism" at the bottom of p34.

- I understood how to do compositions of relations with a single tuple, but I thought it would be nice to clarify what it looks like for larger relations.

- {<a,b>,<a,c>,<b,c>} o {<b,e>,<c,e>,<e,f>} = ?

Reading Questions

- String and word are interchangeable in the reading. What about in our class? Do we use string as a single word, a sentence, or just a list of symbols?
- The reading mentions that German has infinitely many words. Is it correct to state that any language contain infinitely many words? Proof or disproof?

Reading Questions

- Other than keeping the memory cost small, does minimality have other significance?
- Why is each state in Fig 1.31 final? How does it relate the two strings?

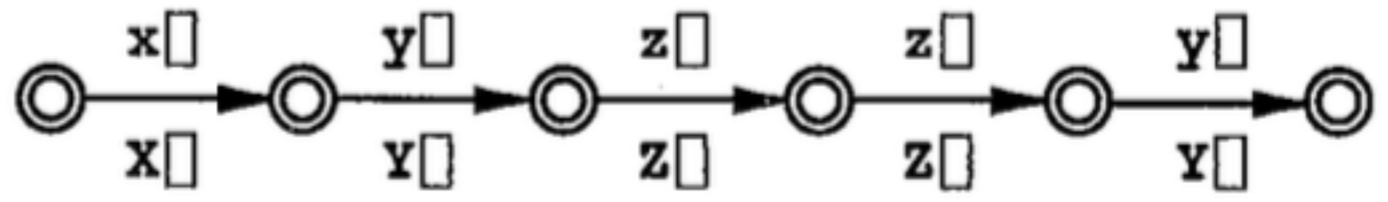


Figure 1.31: A Linearized Path in the Lowercase/Uppercase Transducer

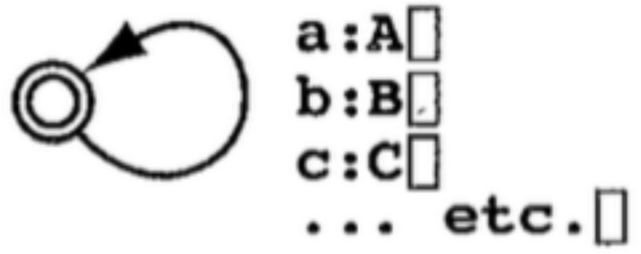


Figure 1.30: The Lowercase/Uppercase Transducer. Each lowercase:UPPERCASE pair of symbols in fact has its own arc.

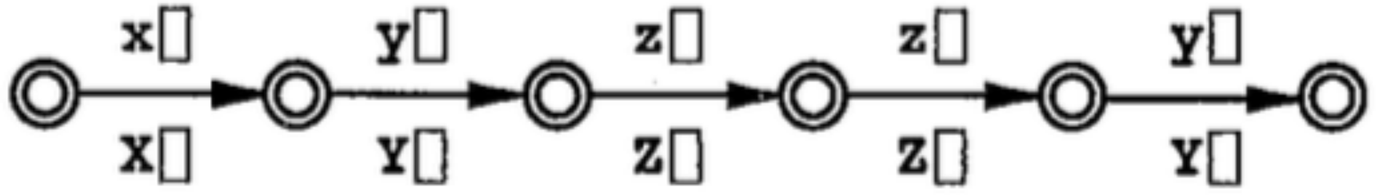


Figure 1.31: A Linearized Path in the Lowercase/Uppercase Transducer

Reading Questions

- on p12 and p13, for the word "canto", the output can be "cantar..." or "canto..." etc. Are there rules for adding epsilons in the analyzers? It seems that by adding different number of epsilons at different places of the input can result in different output?

- Figure 1.17 seems to indicate that epsilon can appear in both the input and output of the Spanish Morphological Analyzer; is there any reason you'd put epsilon in the output? It seems to me that this is just throwing away information, when they could have just assigned some arbitrary symbol to it instead. Is the "-amos" verb ending just so long that the information isn't necessary?

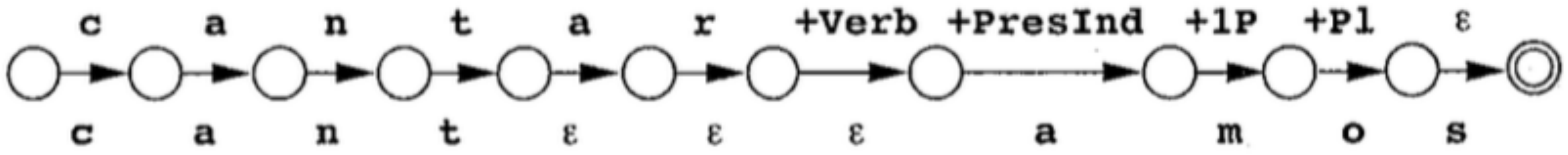


Figure 1.17: Another Verb Path