Ling/CSE 472: Introduction to Computational Linguistics

4/9
Morphology and FST

Overview

- Morphology primer
- Using FSAs to recognize morphologically complex words
- FSTs (definition, cascading, composition)
- FSTs for morphological parsing
- Reading questions

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]	[kæts]	[ninjo]	[ninjos]
‘cat’	‘cats’	Eng: ‘boy’	
[dɑg]	[dɑgz]		
‘dog’	‘dogs’		
[fɪnʧ]	[fɪnʧəz]	[karakol]	[karakoles]
‘finch’	‘finches’	Eng: ‘snail’	

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 far, far more. Sak et al (2011) estimate 4.1M distinct word types in a 4.5M word corpus and 52,000 distinct lexical endings (word forms minus the stem).

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?

Reading questions

- What do 1sg et al mean in Bender 2013 (29)? Is there a complete list of common labels for these?

- What is the syntax being used for the output morphological analyzer? Is the order of grammatical tags arbitrary?

Reading questions

- Considering that morphemes seem to be often dependent on phonological surroundings, when coding for foma is it more useful to set rules for sounds transcribed onto a "phonological alphabet" of symbols rather than directly from the orthography?

- It was noted that alphabets and writing systems don't tend to reflect all of the phonological processes. How much does this actually impact NLP? If we are analyzing written language (in this context of looking at writing), why does the phonology matter?

- What strategies are there for distinguishing between phonological context-driven form changes and cases of allomorphy? Are they always consistent in languages?

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, …

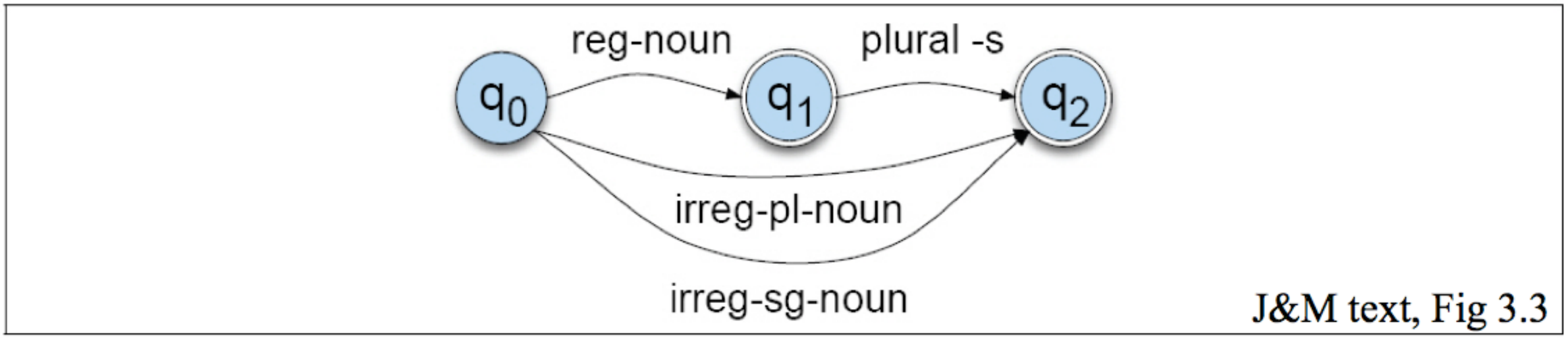
- Problem?

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.

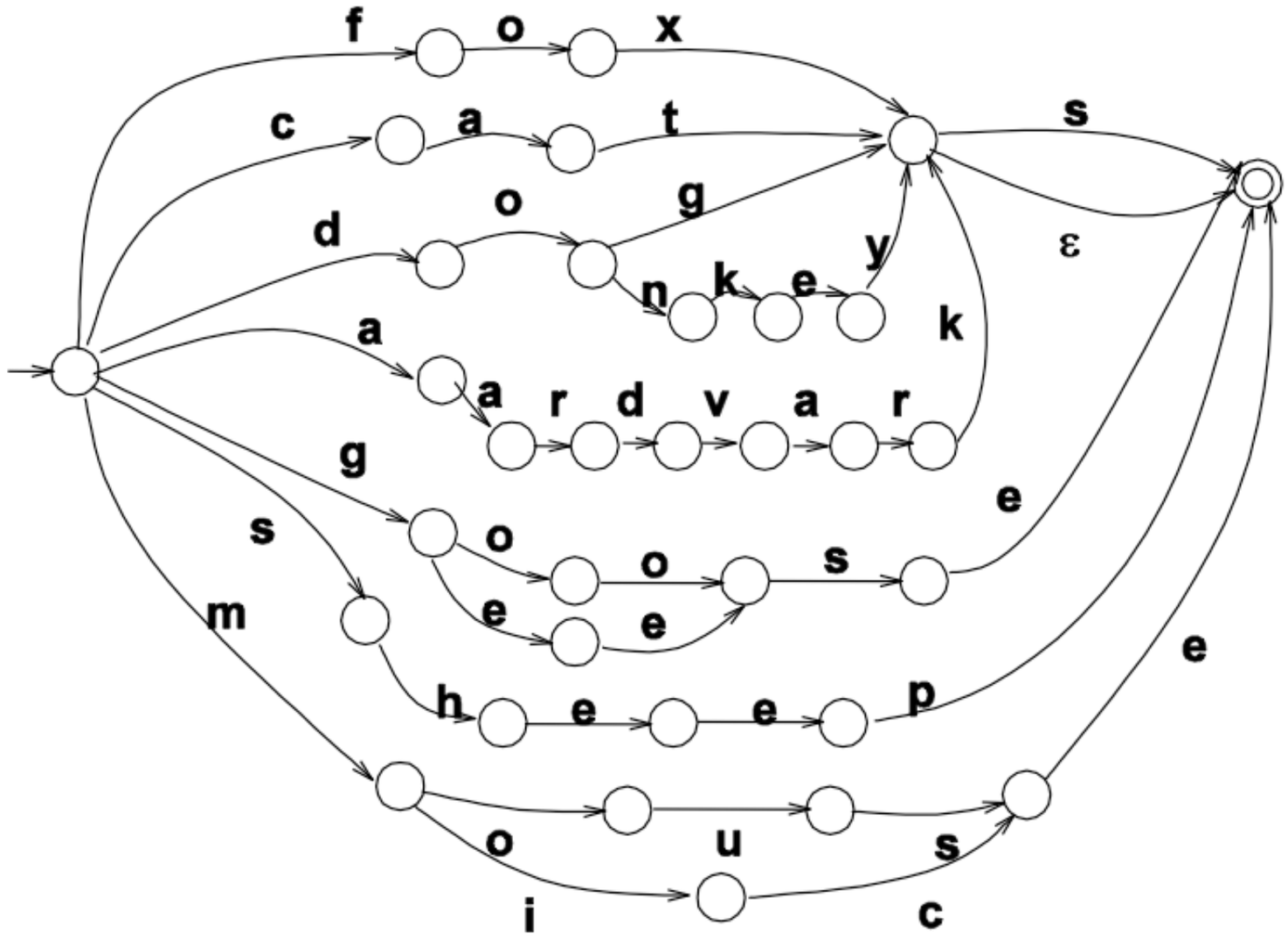
FSA Example using Word Classes

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



J&M text, Fig 3.3

A variation with some words

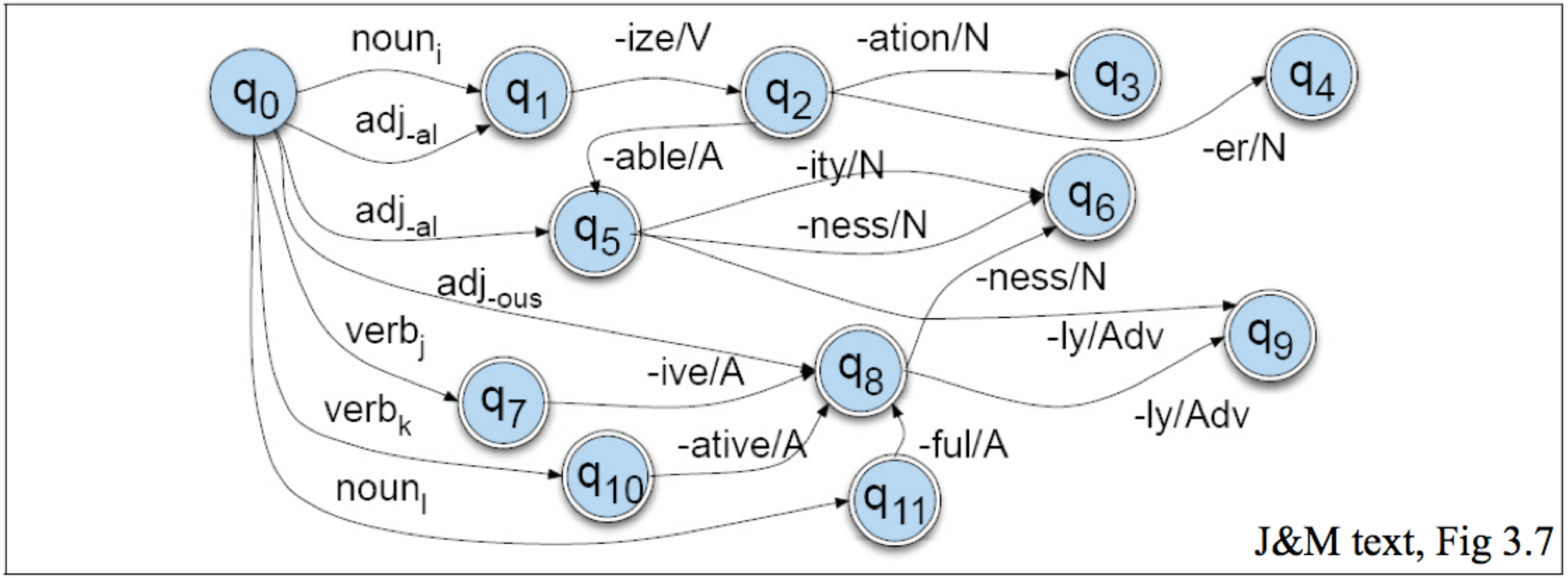


Note: Orthographic issues are not addressed.

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

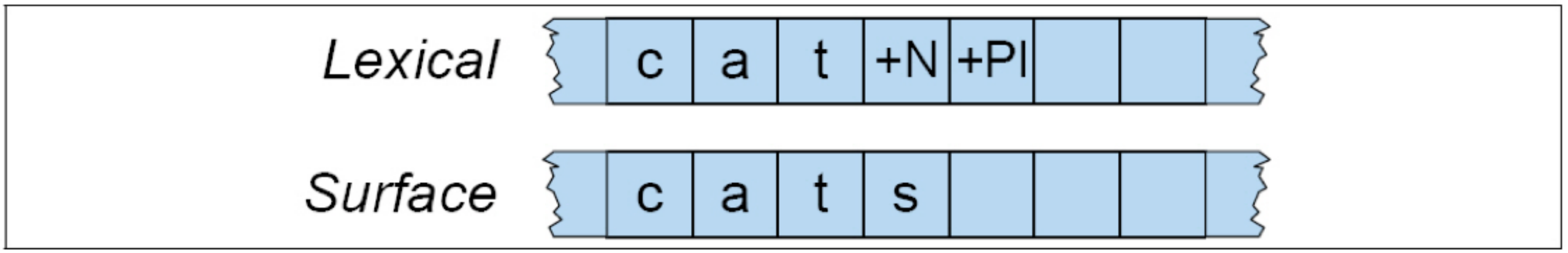
- A parsing task:
  - – Recognize a string
  - – Output information about the stem and affixes of the string
- Something like this:
  - – Input: cats
  - – Output: cat+N+PL
- We will use Finite-State Transducers to accomplish this.

Finite-State Transducer (FST)

- An FST:
    - 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
    - scan 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).

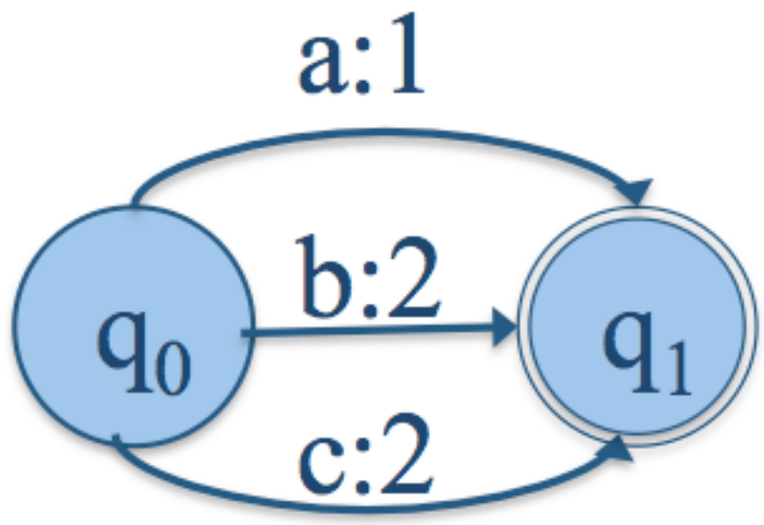


J&M text, Fig 3.12

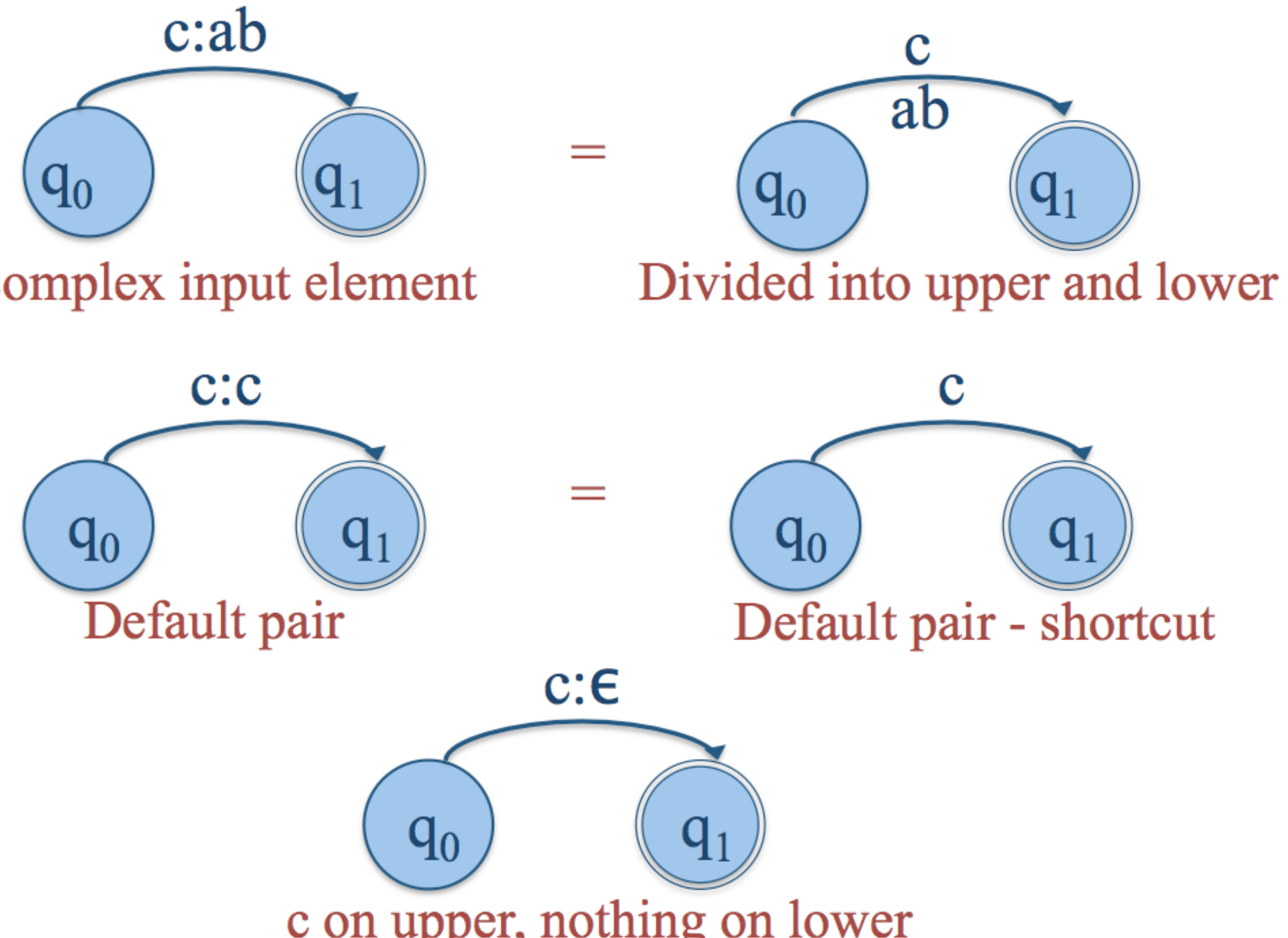
Regular Relations

- 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



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 ◦ T2 = T2 (T1(s))

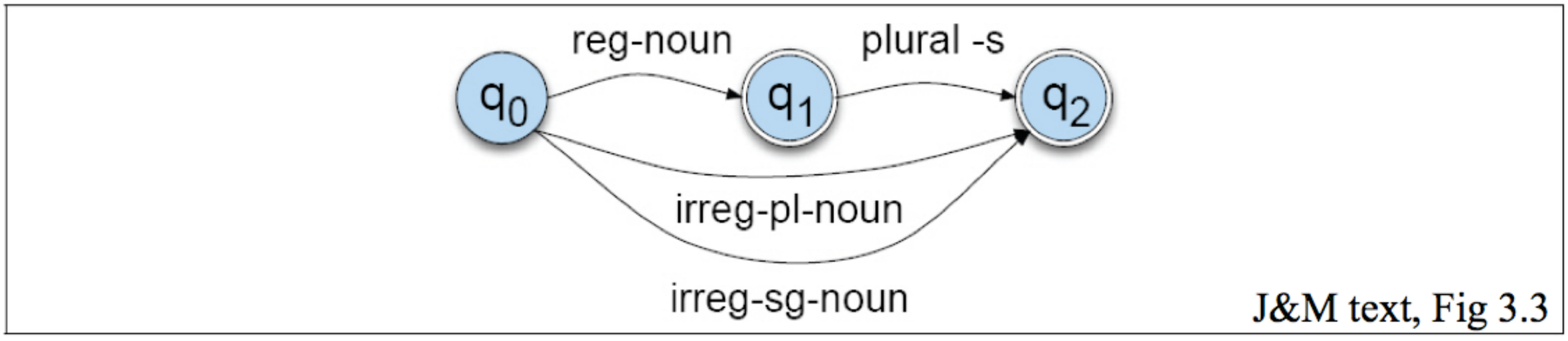
    - where s is the input string

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.

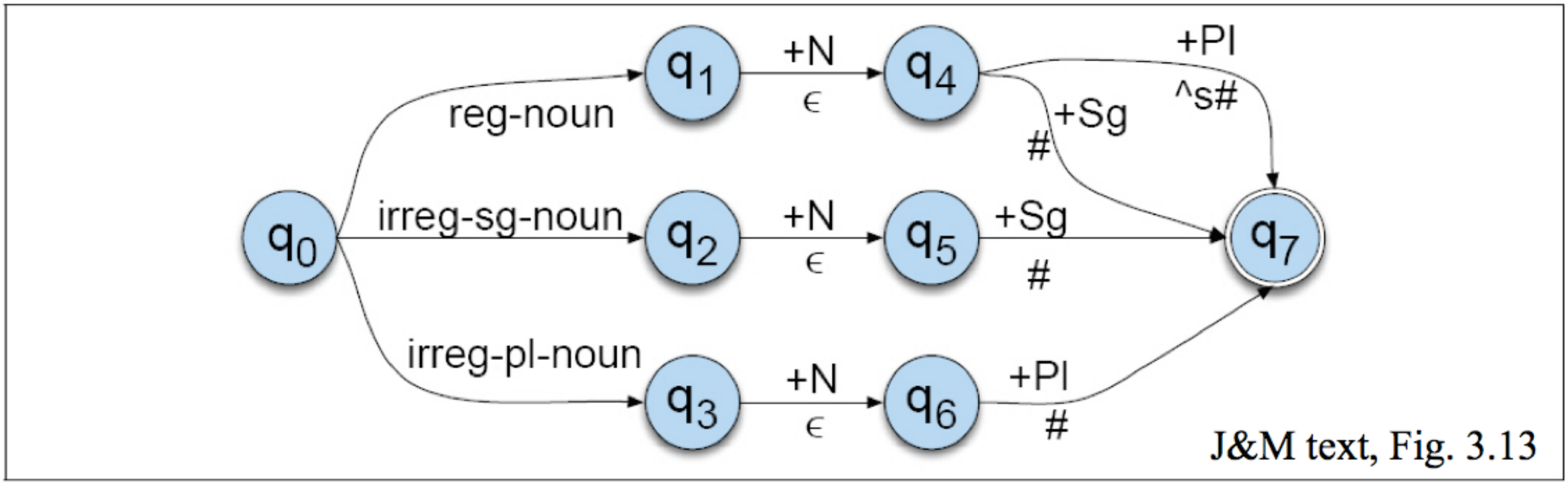
Comparing FSA Example with FST

- Recall this FSA singular and plural recognizer:



J&M text, Fig 3.3

An FST to parse English Noun Number Inflection



^ = 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?

Review: FSAs and FSTs

- FSAs define sets of strings (regular languages).
- FSTs define sets of ordered pairs of strings (regular relations).
- Formally interesting because not all languages/relations can be defined by FSAs/FSTs.
- Are all finite languages and relations regular?
- Linguistically interesting because:
  - FSAs have enough power for morphotactics.
  - FSTs have (almost) enough power for morphophonology.
  - Both are very efficient.

FSTs: “Quiz”

- Why do FSTs have complex symbols labeling the arcs?
- What happens if you give an FST an input on only one “tape”?
- What happens if the input has symbols outside the FST’s alphabet?
- Do the upper and lower tape strings always have the same length?

xfst regex syntax

- Why is it so different from what we see in J&M and elsewhere?
- Why are there so many operators?

Overview

- Morphology primer
- Using FSAs to recognize morphologically complex words
- FSTs (definition, cascading, composition)
- FSTs for morphological parsing
- Reading questions

Reading questions

- FOMA: What's the difference between parallel forms and exceptional forms, and when would you use each?

- I was impressed by all the different kinds of exceptions that could be handled by foma. Even without considering exceptions, though, this seems like a lot of work that has to be done by hand just to describe regular morphological forms. How complex does this get, and does the design all have to be done by hand?

- I didn't quite understand how lexical entries for exceptions such as make/made are incorporated into the FSA. Also, how big would this dictionary be? I wouldn't have excepted most cases to be covered in a regular language-way but that seems to be the case.

FOMA: Parallel forms and exceptions

```
define ParallelForms [c a c t u s %+N %+Pl .x. c a c t i ];
define Exceptions [[{make} "+V" "+Past" .x. {made}] | [{make} "+V" "+PastPart" .x. {made}]] ;
```

Now, we can combine the two with the main grammar as follows:

```
regex Exceptions .P. [Grammar | ParallelForms] ;
```

Reading questions

- What do they mean with "FSTs" when they are talking about alternation rules? Additionally, what do they mean when they say that word-formation rules can be described in the direction of generation?

- Does FSTs for morphological analysis mean that it's reversible, like you can give it either input or output and then it will give you the opposite?

- How does foma deal with inserting ^ into surface forms like "king" which might be accidentally analyzed as "k^ing"? Does foma produce multiple possible strings with ^ inserted and run them all through the transducer?

Reading questions

- I'm not understanding what paths and arcs refer to in the .lexc script output mentioned in the morphological analysis tutorial and how the flag diacritics changes the number of paths. Does this refer to the FST and the number of 'paths' to each different state?

- I was wondering If you would be able to give a more concrete example of how and the when the flags in foma should be used. I was just confused on that section of the tutorial.

Reading questions

- I do wonder in which situations/within which NLP subtasks it is necessary to treat suppletive forms as part of the same word paradigm. Like, in what circumstances is it even necessary to categorically mark go and went as related beyond semantic similarity?

- "All of the kinds of processes described in this chapter are reflected in the orthography of at least some language, at least some of the time." (pp. 33). This give me the impression that the problem of morphophonology is an uncommon problem. How often do these problems come up? Can it be ignored for some languages?

Reading questions

- The book completely convinced me that morphophonological subtleties of different languages can make NLP difficult in real world. Given that many different languages have different morphophonological rules that are distinct from each other, I wonder if there could ever be a universal NLP application that can handle multiple languages. Was there any attempt for universal NLP application that can take more than one languages?

- Near the end of section 24 in Bender Ch 3, there is a mention of NLP needing to be prepared to handle variations in morphemes due to phonological feature influence. Seeing as it would be impossible to ignore morphemes entirely, would it be possible to write a set of multi-language rules that could cover the most common phonological changes? For example, the assimilation that occurs in voicing (e.g. English plural markers).

Reading questions

- For a language with as many irregularities as English, is it better to "brute-force" morphological analysis and write individual rules and exceptions for everything, or would it be better to give the computer a corpus to look over and see if it can recognize the patterns that arise, bypassing having to tediously write so many rules? Or better yet, find some way to combine them?

- How do NLP systems that output speech based on some text usually learn the correct pronunciation for word forms that are spelled much differently than they're pronounced? Is it possible for them to fully "learn" the rules, or would all of these words and their pronunciations basically have to be hard-coded in?