

Lecture 3: Morphology and Finite-State Methods

USC VSoE CSCI 544: Applied Natural Language Processing

Jonathan May -- 梅約納

August 30, 2017

Typical Pipeline for nlp Tasks

1. Find the “units of meaning”
2. Do “shallow” analysis (POS tagging)
3. Do sentence-level analysis (parsing, SRL)
4. Do document-level analysis (topic models, classification)
5. Extrinsic task (question answering)

Typical Pipeline for nlp Tasks

1. Find the “units of meaning”
2. Do “shallow” analysis (POS tagging)
3. Do sentence-level analysis (parsing, SRL)
4. Do document-level analysis (topic models, classification)
5. Extrinsic task (question answering)

This class is (mostly) about English ...

But if we were in Turkey, Finland, or Egypt, this part of the class would take weeks or months. An important step that is really easy in English.

Why morphology

Morpheme

Smallest unit of language that carries meaning

- “books”: two morphemes (“book” and “s”), one syllable
- “unladylike”: three morphemes, four syllables
- To do an analysis of language, we must do an analysis of the most fundamental unit of language!
- This subfield of linguistics is called morphology

Definitions

Derivational

You have a **new** word **derived** from an existing word that alters the **meaning**

- Nominalization: computerization, appointee, killer
- Adjectivization: computational, clueless, embraceable

Inflectional

You have a **variation** of a word that expresses **grammatical** contrast

- tense, number, person
- word class doesn't change
- “The pizza guy comes at noon” (from “come”)

Definitions

- **Root:** common to a set of derived or inflected forms
- **Stem:** root or roots of a word together with derivational affixes
- **Affix:** bound morpheme that comes after or within a root or stem
- **Clitic:** a morpheme that functions like a word but doesn't appear on its own (e.g., the 've in "I've")

Examples

- Rechts+schutz+ver+sicher+ungs+gesell+schaft+en: Legal protection insurance policy (German)
- uygar+laş+tır+ama+dık+larımız+dan+mış+sınız+casına: Behaving as if you are among those whom we could not cause to become civilized (Turkish)
- “tú amaste” “ellos aman” “yo amaría” (Spanish)
- “I eat”, “he eats”, “they’re eating”, “I ate” (English)
- “wo ai”, “ni ai”, “ni.men ai” (Chinese)

Non-Concatenative Morphology

- Infixes (e.g., Tagalog)
 - hingi (borrow)
 - humingi (borrower)
- Circumfixes (e.g., German)
 - sagen (say)
 - gesagt (said)
- Reduplication (e.g., Motu, spoken in Papua New Guinea)
 - mahuta (to sleep)
 - mahutamahuta (to sleep constantly)
 - mamahuta (to sleep, plural)

Comparative Morphology

- Chinese is very easy
- English is fairly simple and regular
 - Few irregular verbs, but they're frequent
 - Derivational morphology is very productive (e.g., “faxed”, “Skyped”, “Brittaed”)

We'd like to analyze words to understand them

- word->analysis (parsing)
- analysis->word (generation)

- Tools we can use to do this:

- Lexicon of stems (cat=noun, city=noun, walk=verb, red=adj) and affixes (s=plural noun, s=3p sing, ing=prespart)
- morphotactics (plural follows noun)
- orthographic rules (y->ie when pluralizing nouns)

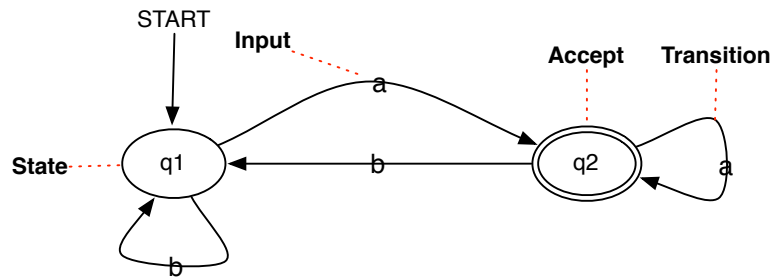
WORD	STEM (+FEATURES)*
cats	cat +N +PL
cat	cat +N +SG
cities	city +N +PL
geese	goose +N +PL
ducks	(duck +N +PL) or (duck +V +3SG)
merging	merge +V +PRES-PART
caught	(catch +V +PAST-PART) or (catch +V +PAST)

Finite-State Automata

A Simple Problem

- We want to know whether a word is in a language or not
 - We'll get to transforming string to morpheme in a bit
- For English, it's possible to get by just with making a list
- Much harder for other languages
- Even for English, you miss out on derivations and inflections
- Turn to a tool called Finite State Automaton (FSA)

Defining FSAs

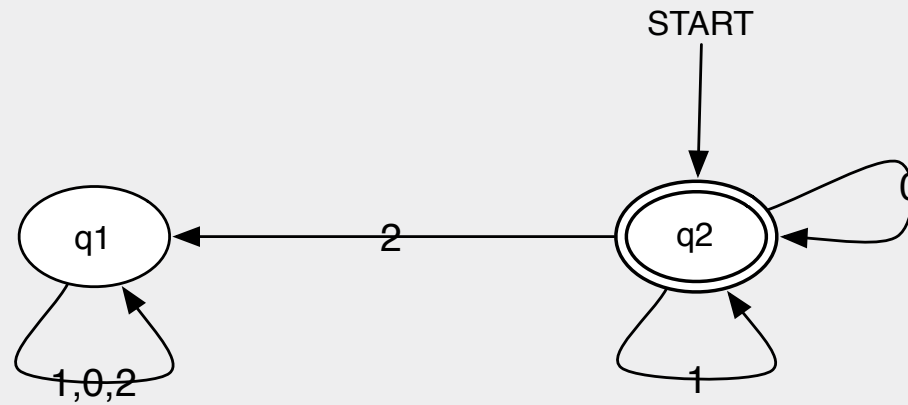


FSA over alphabet $\{a, b\}$

- We define a language to be a set of strings over some alphabet Σ
- A set of states Q
- a designated start state q_0
- a set of accepting final states $F \subset Q$
- edges: given current state q_i and input $x \in \Sigma$, gives new state q_j

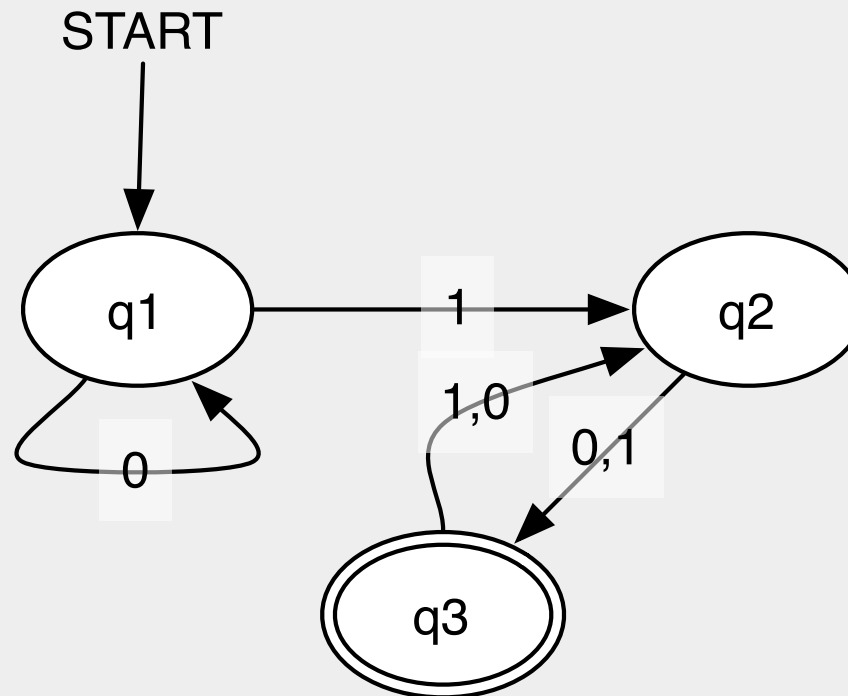
Examples

All binary strings



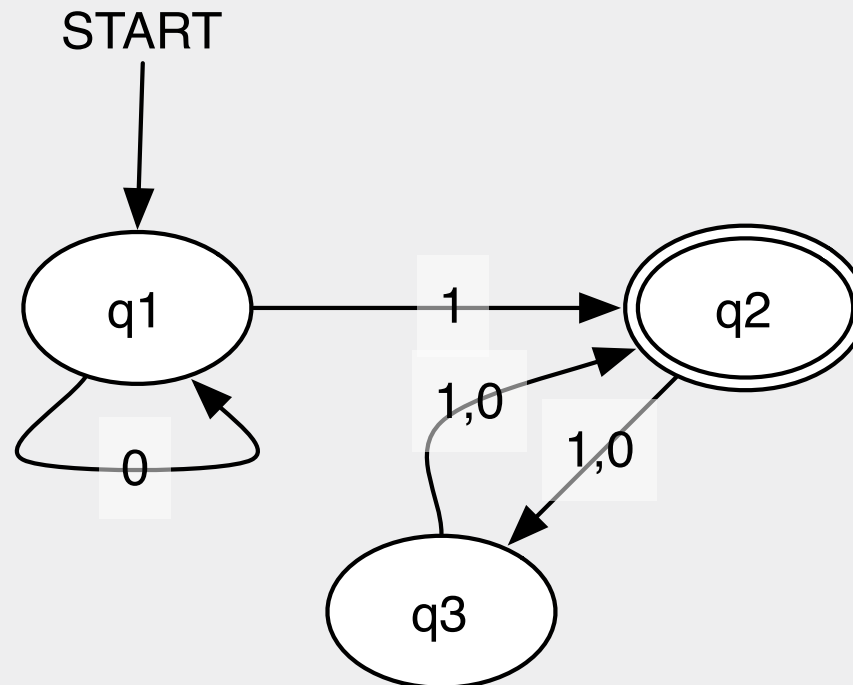
Examples

All non-zero binary strings of even length



Examples

All non-zero binary strings of odd length

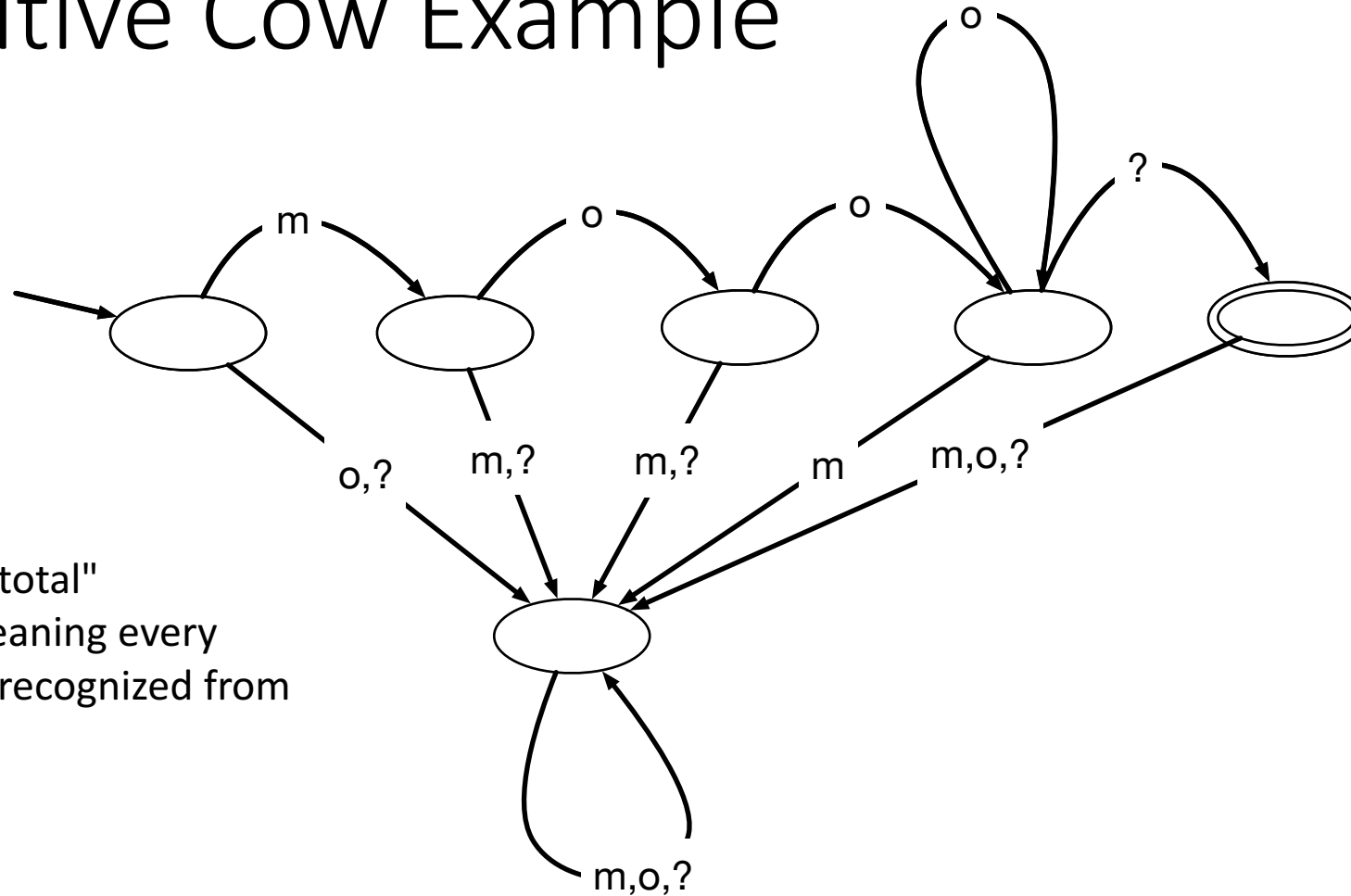


Examples

Suppose we wanted to accept the language of questioning cows

- every string must begin with a “m”
- every string must end with a question mark “?”
- there can be 2+ “o” in between

Inquisitive Cow Example



note: this is a "total"
automaton, meaning every
symbol can be recognized from
every state

often we just leave out
the 'dead' transitions

Connection to Regular Expressions

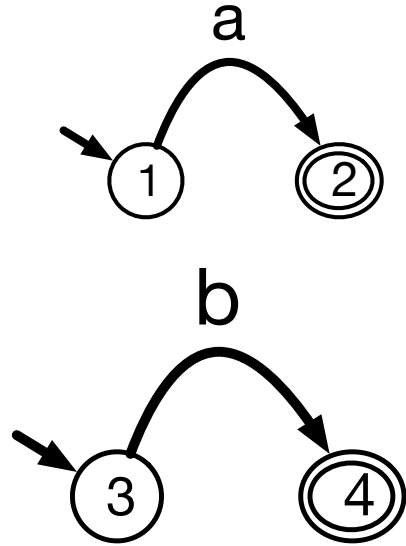
- Regular Expressions and FSAs both recognize exactly the regular languages:
 - For vocabulary Σ :
 $\{\epsilon\}, \emptyset, \{w\}$ for all $w \in \Sigma$ are regular
 - For regular languages A and B :
 $A \cup B, A \circ B, A^*$ are regular
- $/^{\wedge}\$/ = \{\epsilon\}$
- $/^{\wedge}abc\$/ = \{a\} \circ \{b\} \circ \{c\}$
- $/^{\wedge}[ac]\$/ = \{a\} \cup \{c\}; /dog|cat/ = \{dog\} \cup \{cat\}$
- $/^{\wedge}a^*\$/ = \{a\}^*$

Connection to Regular Expressions

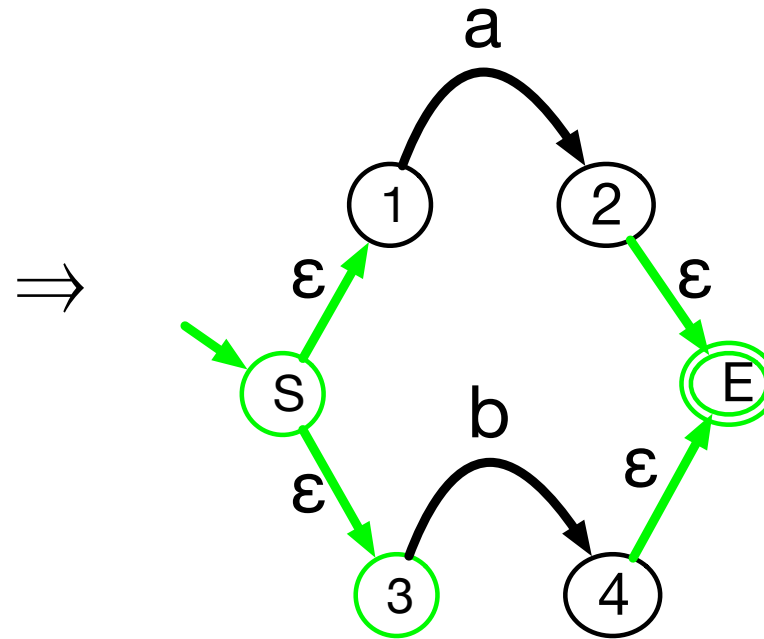
- The rest is syntactic sugar!
 - `./` = [<all of unicode>]
 - `/abc/` = `/^.*abc.*$/`
 - `/a+/` = `/aa*/`
 - `/[^a]/` = `/[ABCD...Zbcd...!@#$...<rest of unicode>]/`

Regex->FSA

- **Union** (\cup)



epsilon = 'change state without reading symbol'



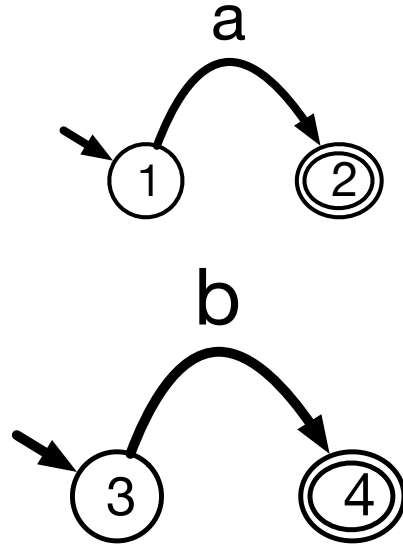
nondeterminism: more than one way to go
(any successful path means acceptance)

Non-deterministic FSA

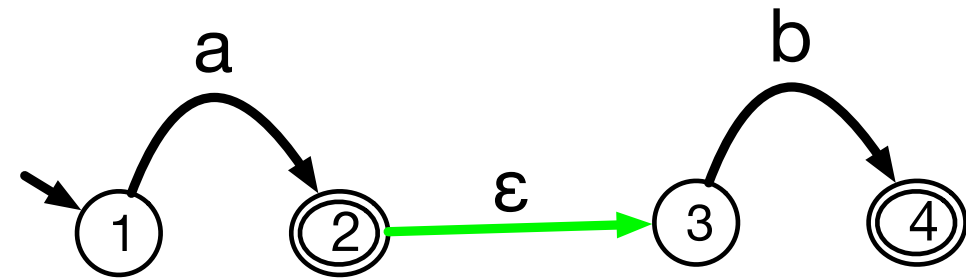
- Allow empty input
- Allows multiple “universes” for strings to follow
- If any accepts, then it is part of the language

Regex->FSA

- Concatenation (\circ)

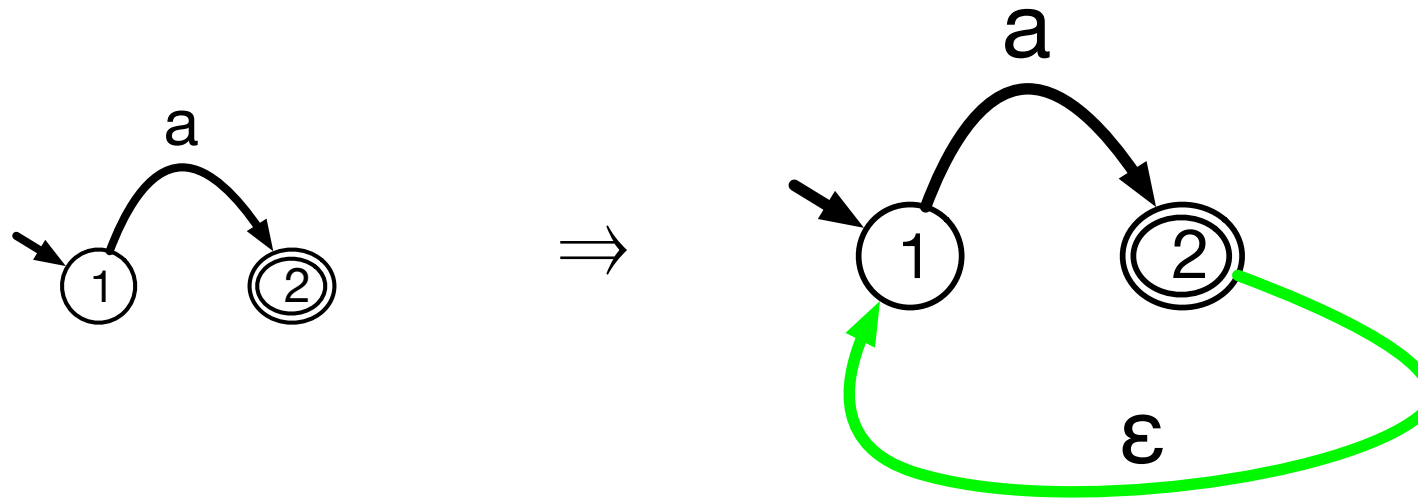


\Rightarrow



Regex->FSA

- Kleene* (*)



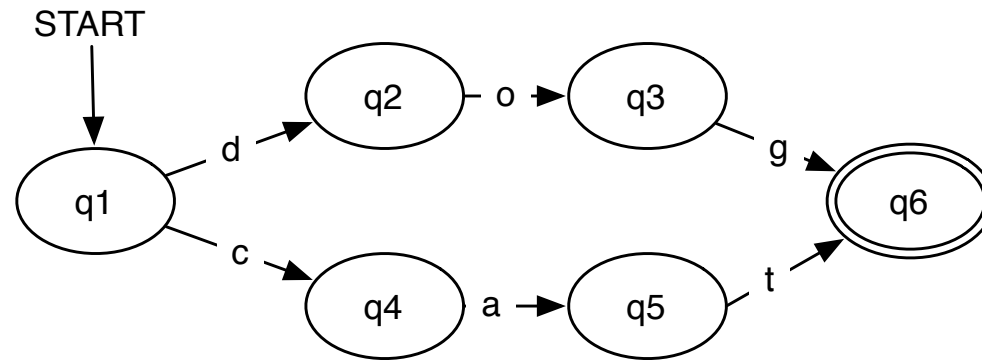
What can you do with FSAs

- Equivalence to regular expressions
- Intersection: given two languages (L_1, L_2) , give $L_1 \cap L_2$
- Difference: given two languages (L_1, L_2) , give $L_1 - L_2$
- Complementation: given a language L_1 , give $\Sigma^* - L_1$
- Reversal: given a language L_1 , give $\{x : x^R \in L_1\}$
- Concatenation: Given two languages (L_1, L_2) , give $\{x : x = y + z, y \in L_1, z \in L_2\}$
- Closure: infinite repetition

Uhh ... what about morphology?

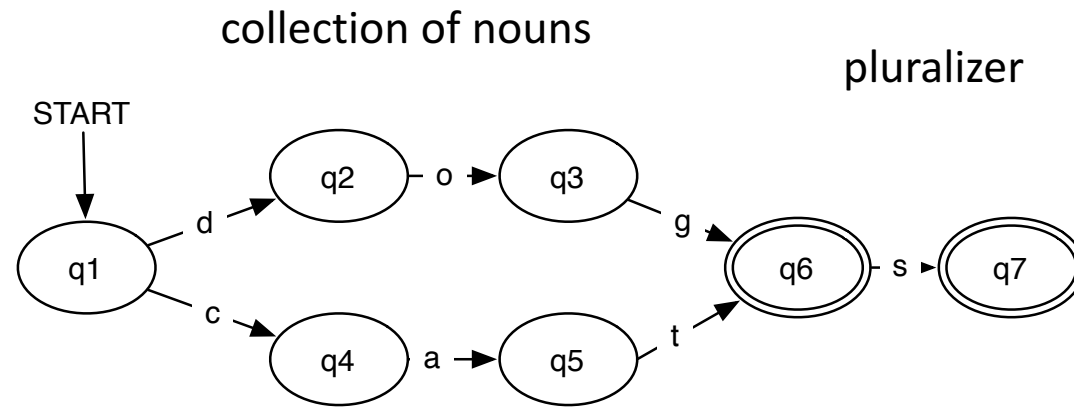
- We've been talking about toy languages, but it works for real languages too
- Why do you want to recognize languages?
 - Spell checkers
 - Language identification
 - Speech synthesis
- Suppose you have an FSA for English stems (one for nouns, verbs, adjectives, etc.)
- Now suppose that you have an FSA that can generate inflectional forms
- Combine them with union / concatenation!

Nouns and their plurals



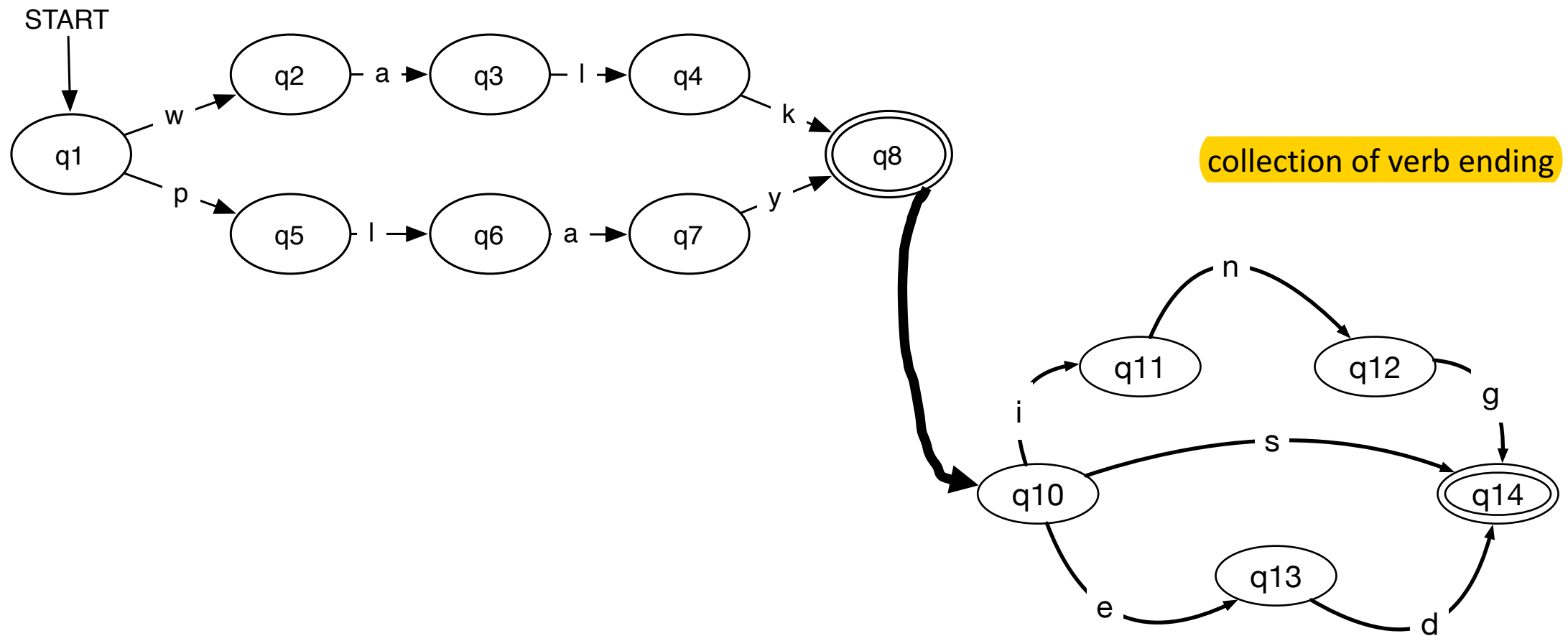
/dog|cat/

Nouns and their plurals

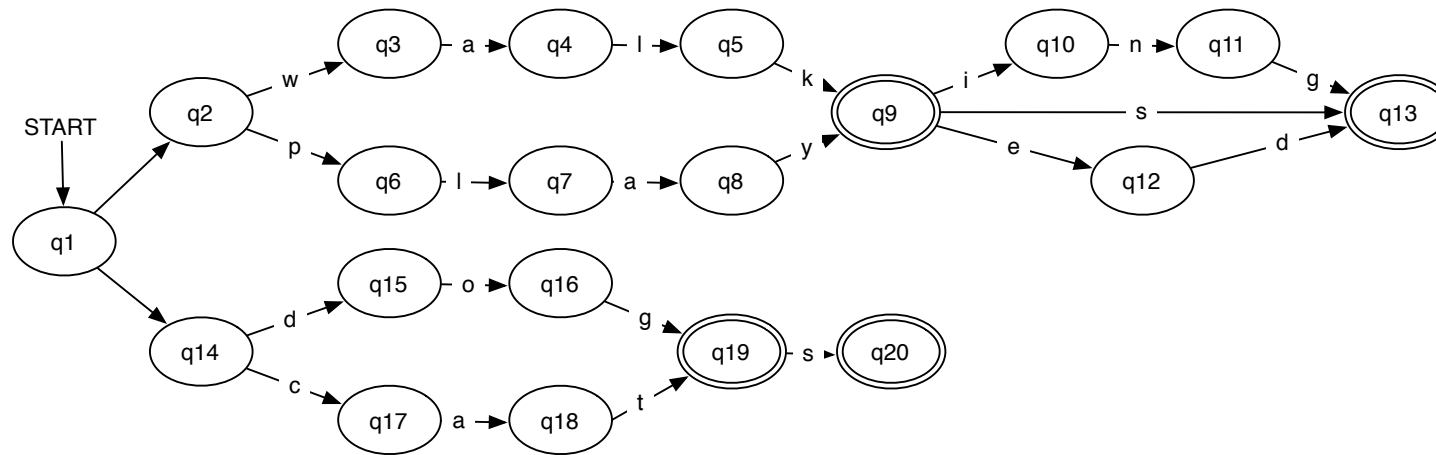


`/(dog|cat)s?/`

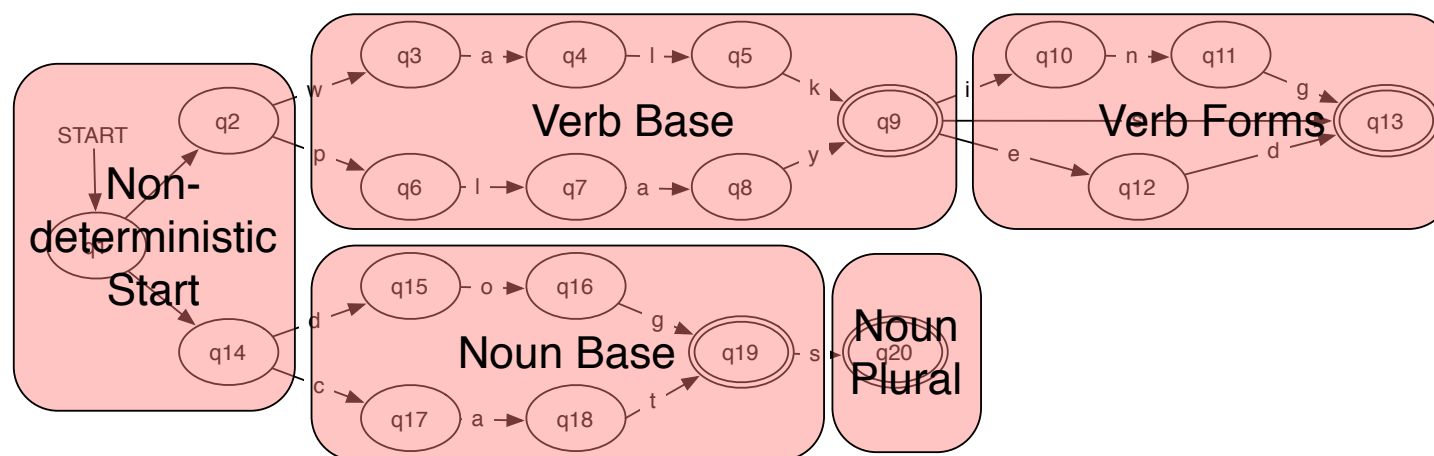
collection of verbs



Non-deterministic composition



Non-deterministic composition



Finite-State Transducers

FSA to FST

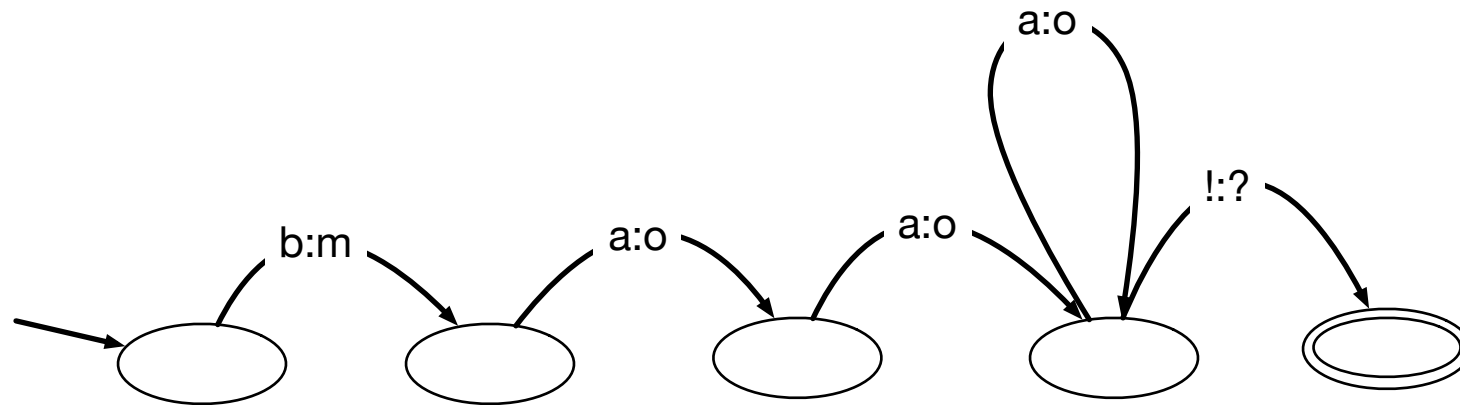
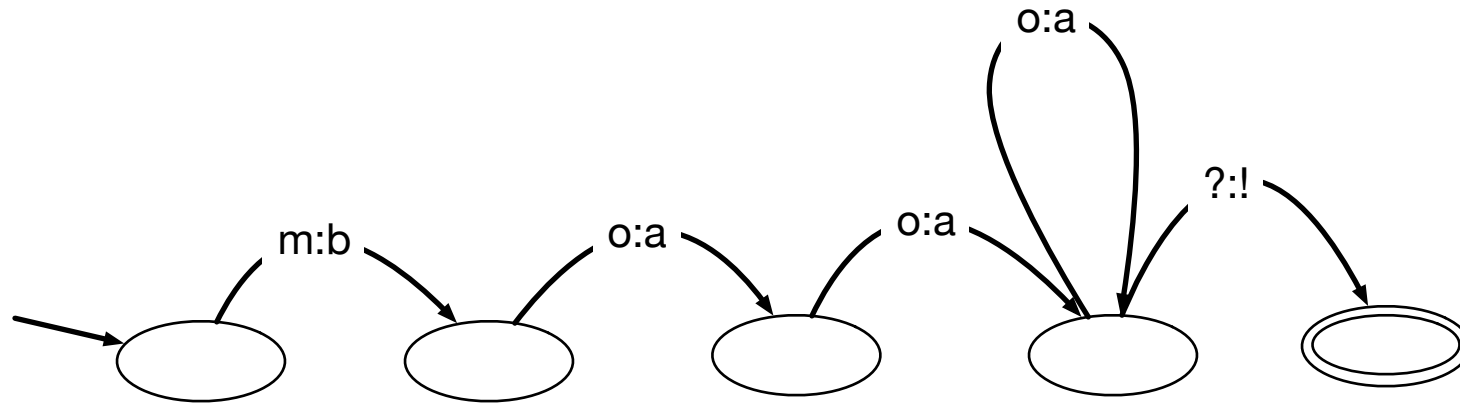
- FSA gives a binary output: is this a string or not
- What if we want to, for example, inflect words to reflect morphological variation? (Or vice-versa, given an inflected form, get back the stem.)
 - Useful for searching (“foxes” and “fox” are related)
 - Useful for generation: I want to say “go”, but what’s the third-person past tense?
- The answer is a finite state transducer

FST definition

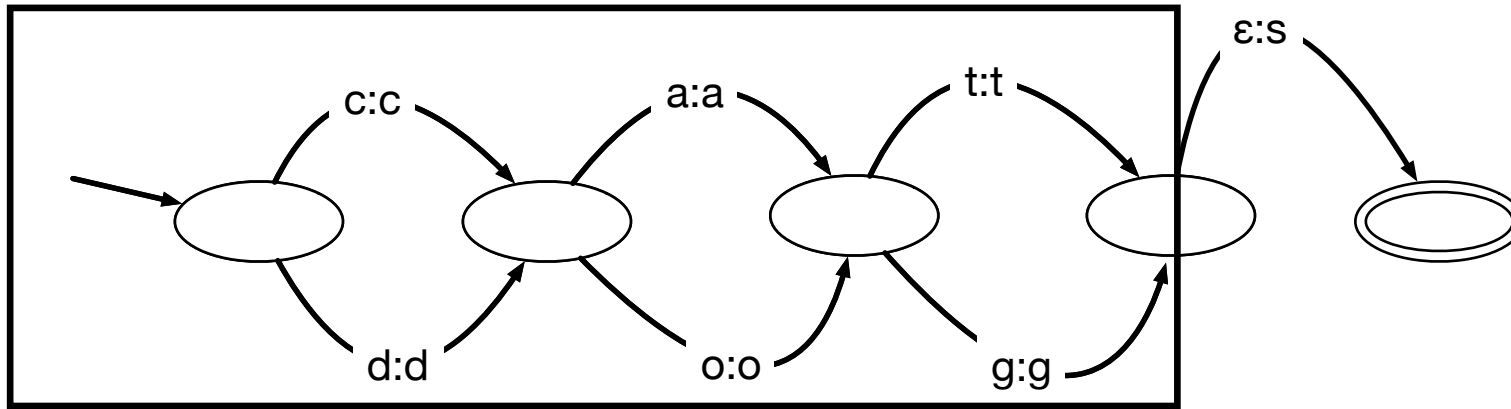
- In addition to everything that you had from an FSA, now each transition also has an output (possibly empty)
- Think of this as “translating” an input string to an output

Symmetric: reverse inputs and outputs to go the other direction.

Example: Inquisitive Cow \iff Emphatic Sheep



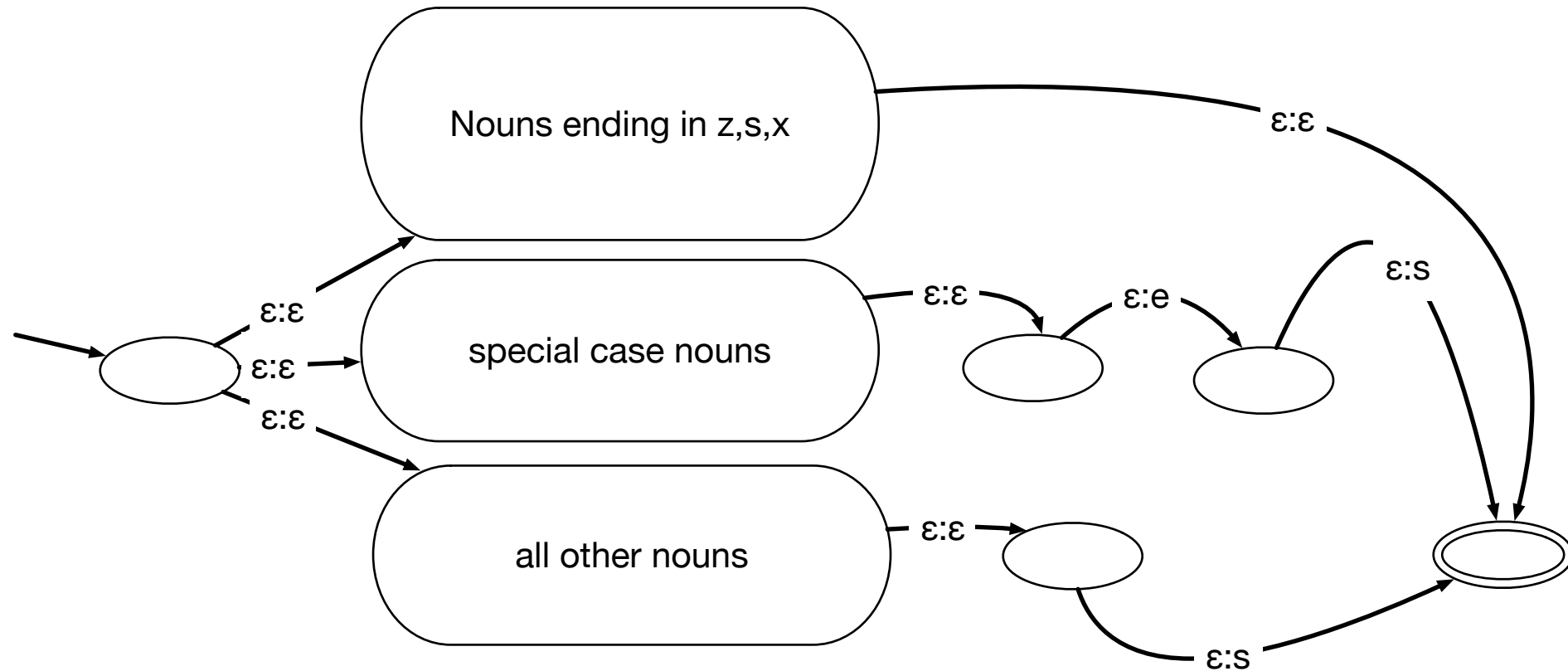
Pluralizer/depluralizer



can add more nouns (horse, president, car)

what about fox, glass, goose, sheep?

Pluralizer/depluralizer



Substitution in Regular Expressions

- FSA:Regex match :: FST:Regex substitute
 - Actually not true: regex substitute is more powerful
- syntax: s/<regex>/<replacement>/<modifiers>
 - modifiers: g (match everywhere), i (case insensitive) (these are most frequent)
- We've already seen one a lot: s/ /\n/g = "replace all whitespace with newline"
- s/a/d/ transforms "abacus" into "dbacus"
- s/a/d/g transforms "abacus" into "dbdcus"

Substitution with grouping

- Text in parentheses in the LHS of a substitution can be referred to in the RHS
- depluralizer: `s/(dog|cat)s/\1/`
 - dogs -> dog; cats -> cat
- multiple groups: `s/(dog|cat)s (eat|drink|love) (dog|cat)s/\1 \2s \3/`
 - "It's a dogs eat cats world" -> "It's a dog eats cat world"
- You'll see this in hw2!
 - python formulation: `re.sub(LHS, RHS, WORD)`