Regular Expressions and Automata in Natural Language Analysis

CS 4705

Some slides adapted from Hirschberg, Dorr/Monz, Jurafsky

Rule-based vs. Statistical Approaches

- Rule-based = linguistic
- For what problems is rule-based better suited and when is statistics better
    - Identifying proper names
    - Distinguishing a biography from a dictionary entry
    - Answering questions
- How far can a simple method take us?
    - *How much is Google worth?*
    - *How much is Microsoft worth?*
- How much knowledge of language do our algorithms need to do useful NLP?
    - 80/20 Rule:
        - Claim: 80% of NLP can be done with simple methods
        - When should we worry about the other 20%?

Rule-based vs. Statistical Approaches

- Rule-based = linguistic
- For what problems is rule-based better suited and when is statistics better
  - Identifying proper names
  - Distinguishing a biography from a dictionary entry
  - Answering questions
- How far can a simple method take us?
  - *How much is **Google** worth?*
  - *How much is **Microsoft** worth?*
  - *How much is **IBM** worth?*
- How much knowledge of language do our algorithms need to do useful NLP?
  - 80/20 Rule:
    - Claim: 80% of NLP can be done with simple methods
    - When should we worry about the other 20%?

Rule-based vs. Statistical Approaches

- Rule-based = linguistic
- For what problems is rule-based better suited and when is statistics better
  - Identifying proper names
  - Distinguishing a biography from a dictionary entry
  - Answering questions
- How far can a simple method take us?
  - *How much is **Google** worth?*
  - *How much is **Microsoft** worth?*
  - *How much is **IBM** worth?*
  - *How much is **Walmart** worth?*
- How much knowledge of language do our algorithms need to do useful NLP?
  - 80/20 Rule:
    - Claim: 80% of NLP can be done with simple methods
    - When should we worry about the other 20%?

Rule-based vs. Statistical Approaches

- Rule-based = linguistic
- For what problems is rule-based better suited and when is statistics better
  - Identifying proper names
  - Distinguishing a biography from a dictionary entry
  - Answering questions
- How far can a simple method take us?
  - *How much is **Google** worth?*
  - *How much is **Microsoft** worth?*
  - How much is a **Columbia University education** worth?
  - How much is the **Statue of Liberty** worth?
  - How much **is your life** worth?
- How much knowledge of language do our algorithms need to do useful NLP?
  - 80/20 Rule:
    - Claim: 80% of NLP can be done with simple methods
    - When should we worry about the other 20%?

Today

- Review some simple representations of language and see how far they will take us
  - Regular Expressions
  - Finite State Automata
- Think about the limits of these simple approaches
  - When are simple methods good enough?
  - When do we need something more?

Regular Expression/Pattern Matching in NLP

- Simple but powerful tools for 'shallow' processing of a document or "corpus"
  - What word begins a sentence?
  - What words begin a question?
  - Identify all noun phrases
- Allow us to
  - Build simple interactive applications (e.g. Eliza)
  - Morphological analysis
  - Recognize Named Entities (NE): people names, company names

Review

RE	Matches	Uses
/./	Any character	A non-blank line
/\./, /\?/	A '.', a '?'	A statement, a question
/[bckmsr]/	Any char in set	Rhyme:/[bckmrs]ite/
/[a-z]/	Any l.c. letter	Rhyme: /[a-z]ite/
/ [A-Z]/	Capitalized letter	Possible NE
/ [^A-Z]/	Lower case letter	Not an NE

RE	Description	Uses?
/a*/	Zero or more a’s	/(very[ ])*/
/a+/	One or more a’s	/(very[ ])+/
/a?/	Optional single a	/(very[ ])?/
/cat|dog/	‘cat’ or ‘dog’	/[A-Z,a-z]* (cat|dog)/
/^[Nn]o$/		A line with only ‘No’ or ‘no’ in it
/\bun\B/	Prefixes	Words prefixed by ‘un’ (nb. *union*)

RE plus	E.G.
/kitt(y|ies|en|ens)/	Morphological variants of 'kitty'
/ (.+ier) and \1 /	Patterns: happier and happier, fuzzier and fuzzier, *classifier and classifier*

Question Answering

- Eliza

Eliza (Weizenbaum)

- Men are all alike.
- IN WHAT WAY?
- They're always bugging us about something or other.
- CAN YOU THINK OF A SPECIFIC EXAMPLE?
- Well, my boyfriend made me come here.
- YOUR BOYFRIEND MADE YOU COME HERE
- He says I'm depressed much of the time.
- I AM SORRY TO HEAR YOU ARE DEPRESSED
- It's true. I am unhappy
- DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
- I need some help, that much seems certain.
- WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
- Perhaps I could learn to get along with my mother.
- TELL ME MORE ABOUT YOUR FAMILY
- My mother takes care of me.
- WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU
- My father.
- YOUR FATHER
- You are like my father in some ways.

Eliza-style regular expressions

Step 1: replace first person with second person references

```
s/\bI(’m| am)\b /YOU ARE/g
s/\bmy\b /YOUR/g
S/\bmine\b /YOURS/g
```

Step 2: use additional regular expressions to generate replies

```
s/.* YOU ARE (depressed|sad) .*/I AM SORRY TO HEAR YOU ARE \1/
s/.* YOU ARE (depressed|sad) .*/WHY DO YOU THINK YOU ARE \1/
s/.* all .*/IN WHAT WAY/
s/.* always .*/CAN YOU THINK OF A SPECIFIC EXAMPLE/
```

Step 3: use scores to rank possible transformations

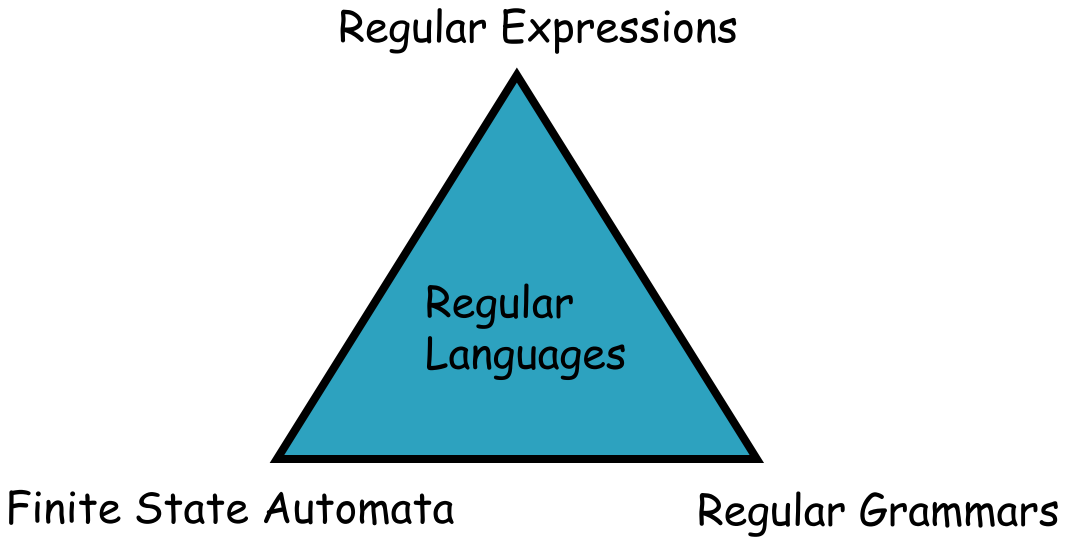
How far does this allow you to go? How much of a question answering system?

Advantages?

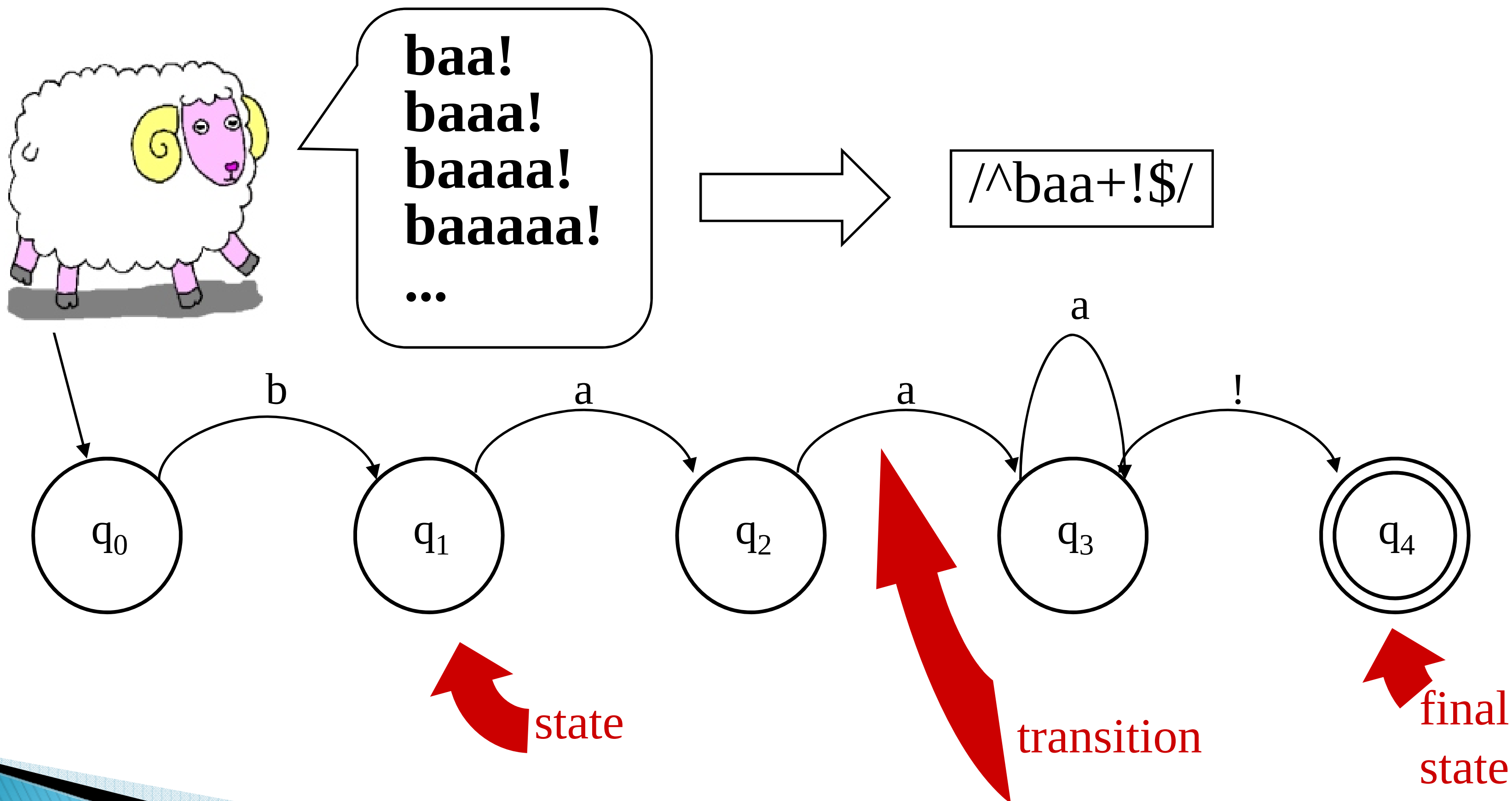
Disadvatages?

Three Views

- Three equivalent formal ways to look at what we're up to



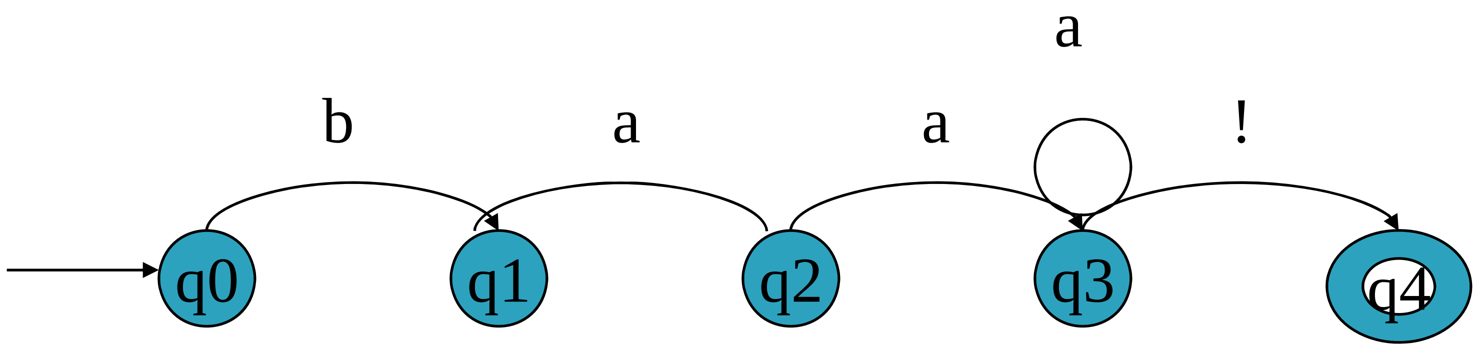
Finite-state Automata (Machines)



Slide from Do

Formally

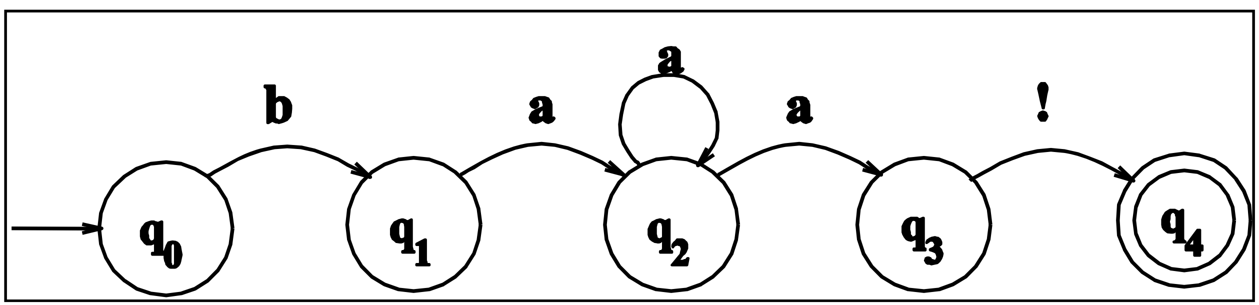
- FSA is a 5-tuple consisting of
  - Q: set of states {q0,q1,q2,q3,q4}
  - Σ: an alphabet of symbols {a,b,!}
  - q0: a start state in Q
  - F: a set of final states in Q {q4}
  - δ(q,i): a transition function mapping Q x Σ to Q



Yet Another View

- State-transition table

	Input		
State	b	a	!
0	1	∅	∅
1	∅	2	∅
2	∅	3	∅
3	∅	3	4
4:	∅	∅	∅

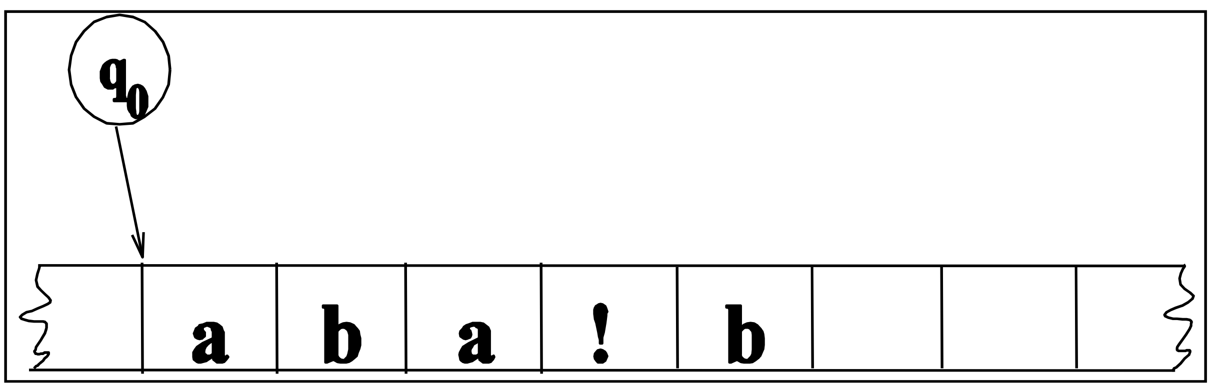


Recognition

- Recognition is the process of determining if a string should be accepted by a machine
- Or… it's the process of determining if a string is in the language we're defining with the machine
- Or… it's the process of determining if a regular expression matches a string

Recognition

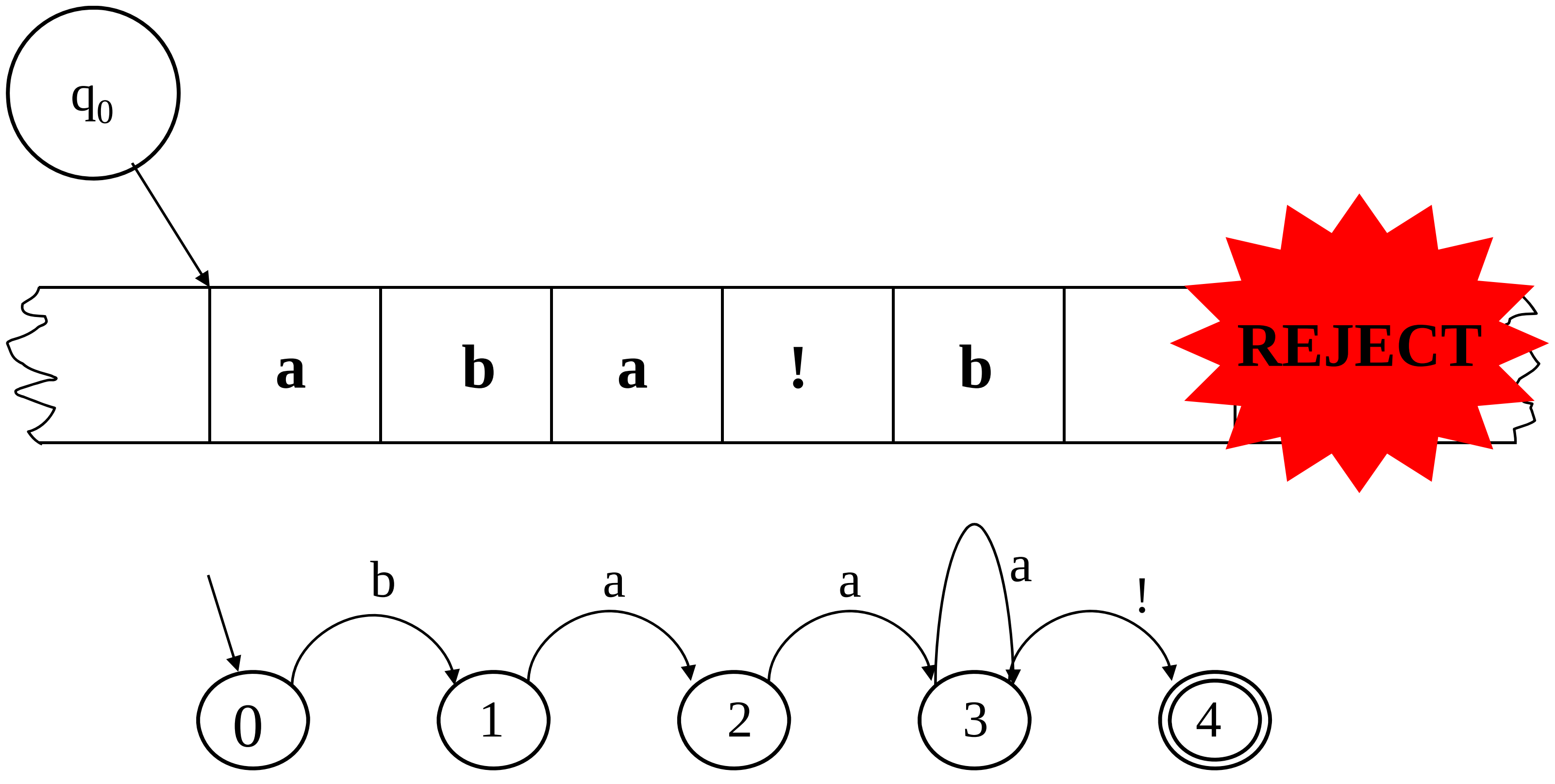
- Traditionally, (Turing's idea) this process is depicted with a tape.



Recognition

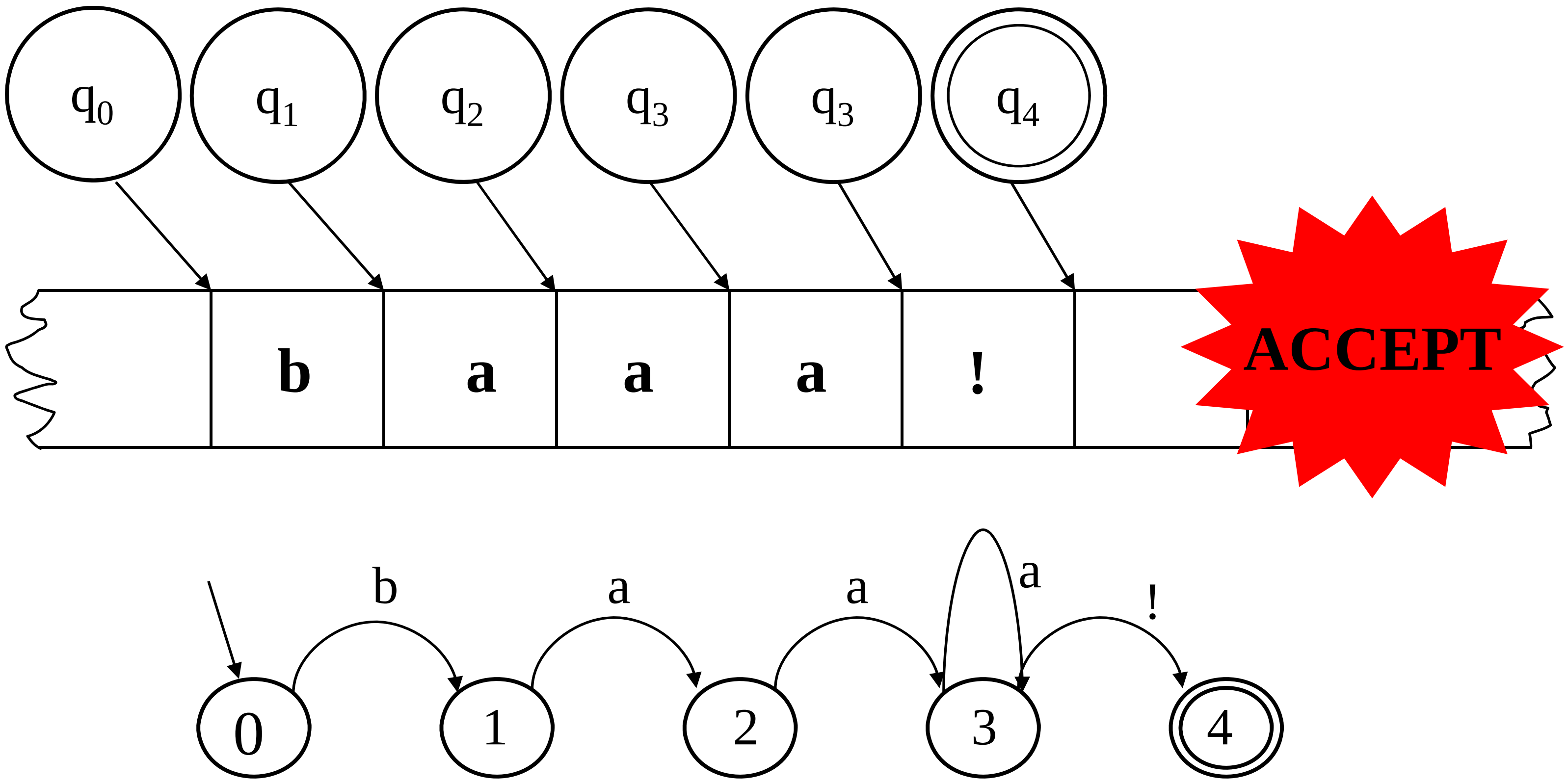
- Start in the start state
- Examine the current input
- Consult the table
- Go to a new state and update the tape pointer.
- Until you run out of tape.

Input Tape



Slide from Dorr/Monz

Input Tape

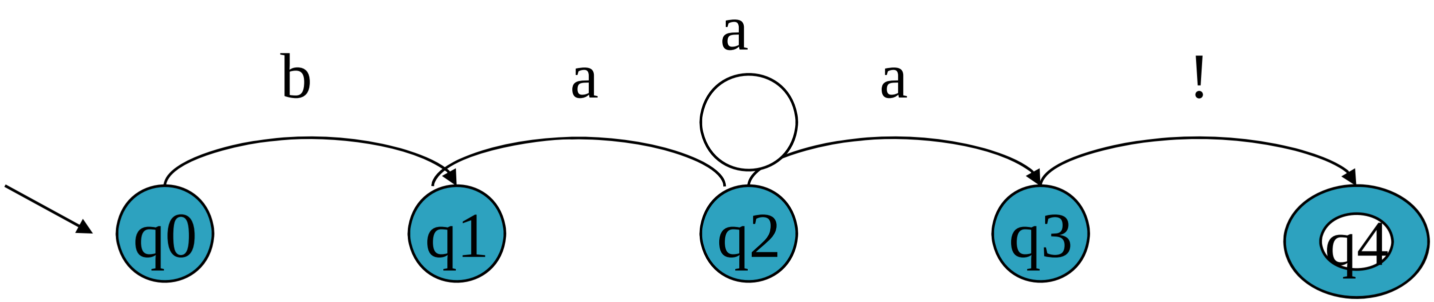


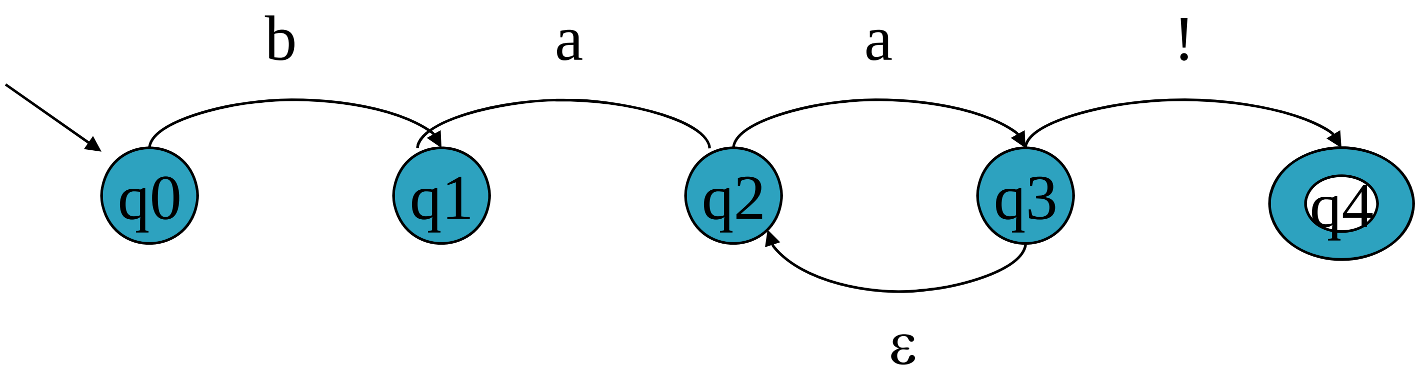
Slide from Dorr/Monz

Key Points

- Deterministic means that at each point in processing there is always one unique thing to do (no choices).
- D-recognize is a simple table-driven interpreter
- The algorithm is universal for all unambiguous languages.
  - To change the machine, you change the table.

Non-Deterministic FSAs for SheepTalk

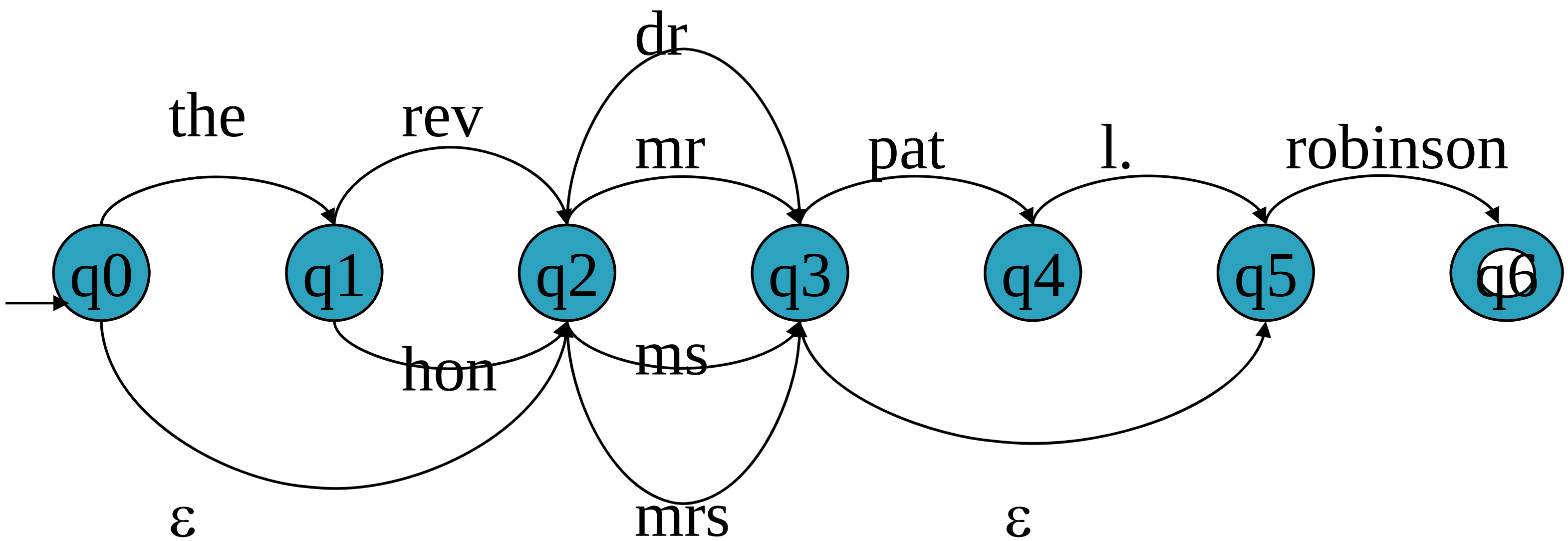




Problems of Non-Determinism

- At any choice point, we may follow the wrong arc
- Potential solutions:
  - Save backup states at each choice point
  - Look-ahead in the input before making choice
  - Pursue alternatives in parallel
  - Determinize our NFSAs (and then minimize)
- FSAs can be useful tools for recognizing – and generating – subsets of natural language
  - But they cannot represent all NL phenomena (e.g. center embedding: The mouse the cat chased died.)

FSAs as Grammars for Natural Language: Names



Recognizing Person Names

- If we want to extract all the proper names in the news, will this work?
  - What will it miss?
  - Will it accept something that is not a proper name?
  - How would you change it to accept all proper names without false positives?
  - Precision vs. recall….

English Morphology

- Morphology is the study of the ways that words are built up from smaller meaningful units called morphemes
- 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

Regular and Irregular Nouns and Verbs

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

What we want

- Something to automatically do the following kinds of mappings:
- Cats   cat +N +PL
- Cat   cat +N +SG
- Cities   city +N +PL
- Merging   merge +V +Present-participle
- Caught   catch +V +past-participle

Why care about morphology?

Spelling correction: referece

- Morphology in machine translation
  - Spanish words **quiero** and **quieres** are both related to **querer** '**want**'
- Hyphenation algorithms: refer-ence
- Part-of-speech analysis: google, googler
- Text-to-speech: grapheme-to-phoneme conversion
  - ho*th*ouse (/T/ or /D/)
- Allows us to guess at meaning
  - **'Twas brillig and the slithy toves…**
  - **Muggles moogled migwiches**

Morphology and FSAs

- We'd like to use the machinery provided by FSAs to capture facts about morphology
  - Ie. Accept strings that are in the language
  - And reject strings that are not
  - And do it in a way that doesn't require us to in effect list all the words in the language

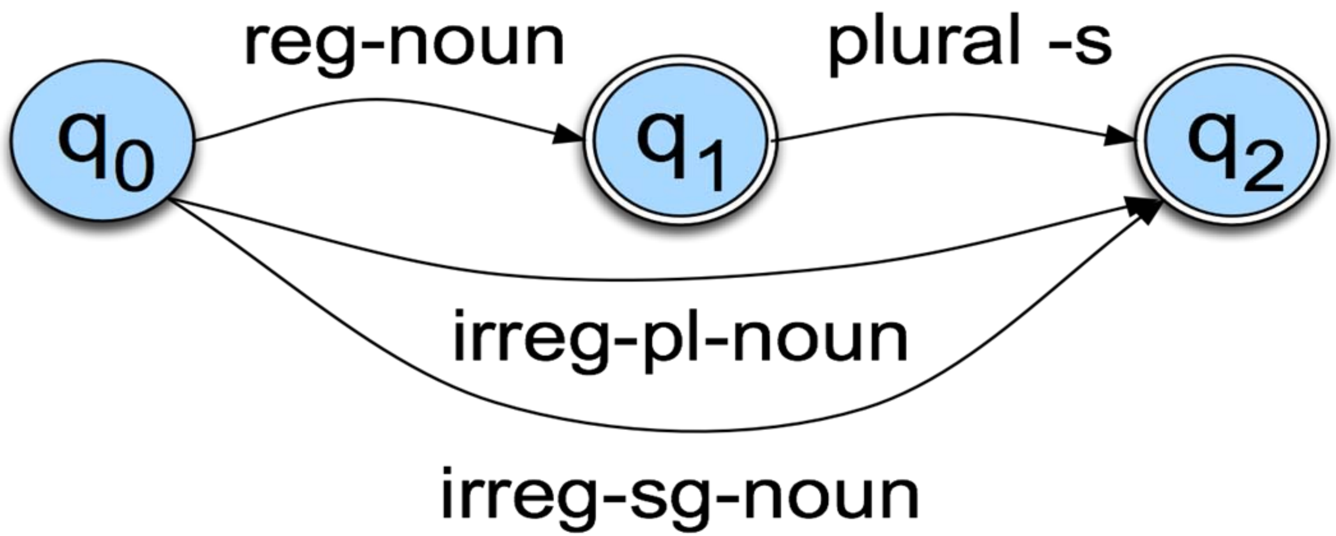
What do we need to build a morphological parser?

- Lexicon: list of stems and affixes (w/ corresponding part of speech (p.o.s.))
- Morphotactics of the language: model of how and which morphemes can be affixed to a stem
- Orthographic rules: spelling modifications that may occur when affixation occurs
  - in → il in context of l (in- + legal)
- Most morphological phenomena can be described with regular expressions – so finite state techniques often used to represent morphological processes

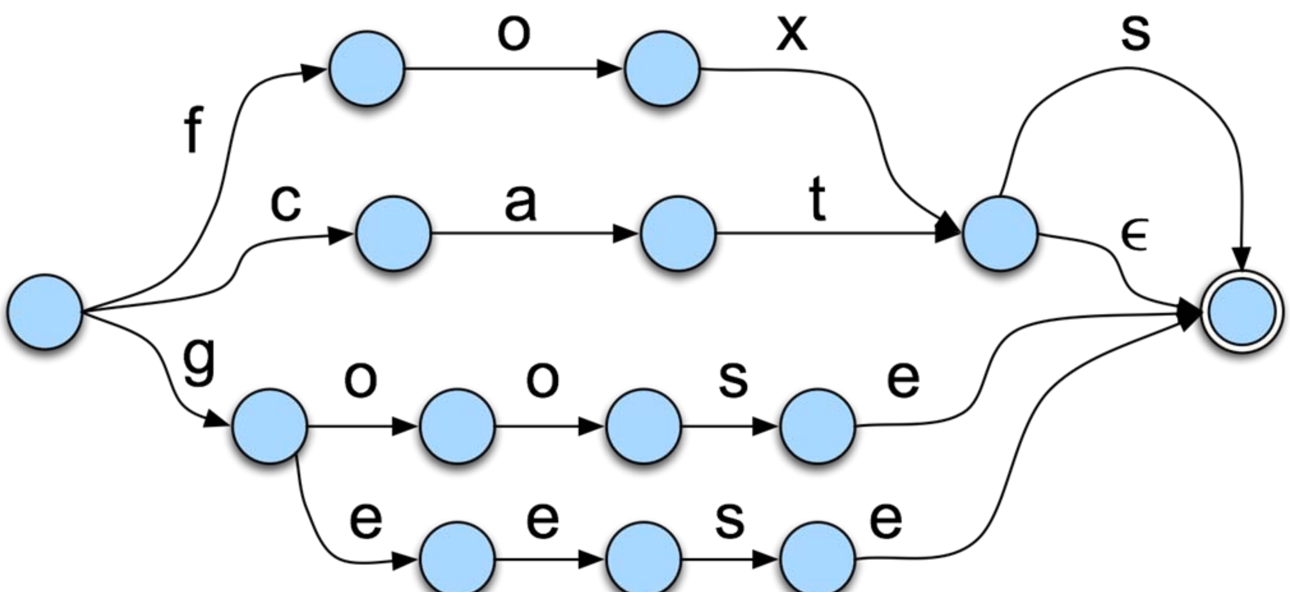
Start Simple

- Regular singular nouns are ok
- Regular plural nouns have an –s on the end
- Irregulars are ok as is

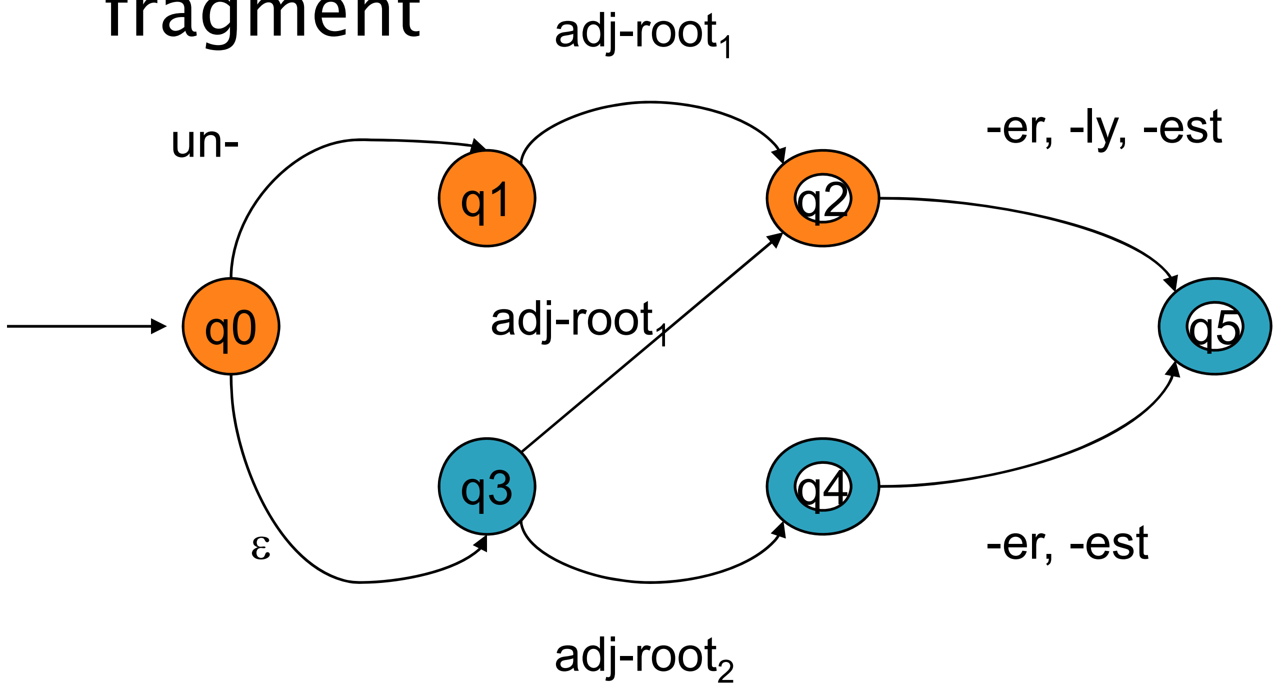
Simple Rules



Now Add in the Words



- Derivational morphology: adjective fragment



- Adj-root$_1$: clear, happi, real
- Adj-root$_2$: big, red (*bigly)

Parsing/Generation vs. Recognition

- We can now run strings through these machines to recognize strings in the language
  - Accept words that are ok
  - Reject words that are not
- But recognition is usually not quite what we need
  - Often if we find some string in the language we might like to find the structure in it (parsing)
  - Or we have some structure and we want to produce a surface form (production/generation)
- Example
  - From “cats” to “cat +N +PL”

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"

Applications

- The kind of parsing we're talking about is normally called morphological analysis
- It can either be
  - An important stand-alone component of an application (spelling correction, information retrieval)
  - Or simply a link in a chain of processing

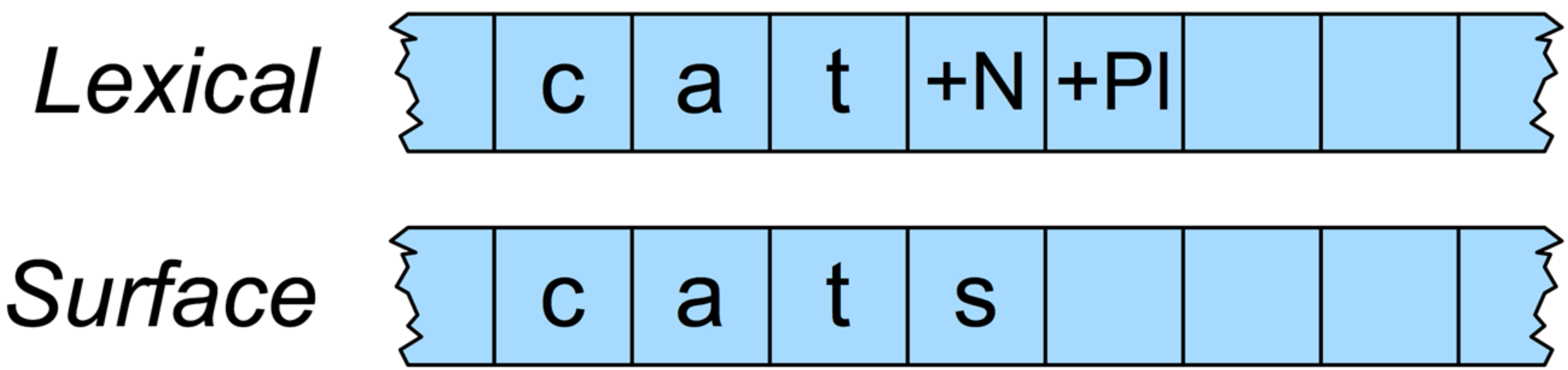
Generativity

- Nothing really privileged about the directions.
- We can write from one and read from the other or vice-versa.
- One way is generation, the other way is analysis

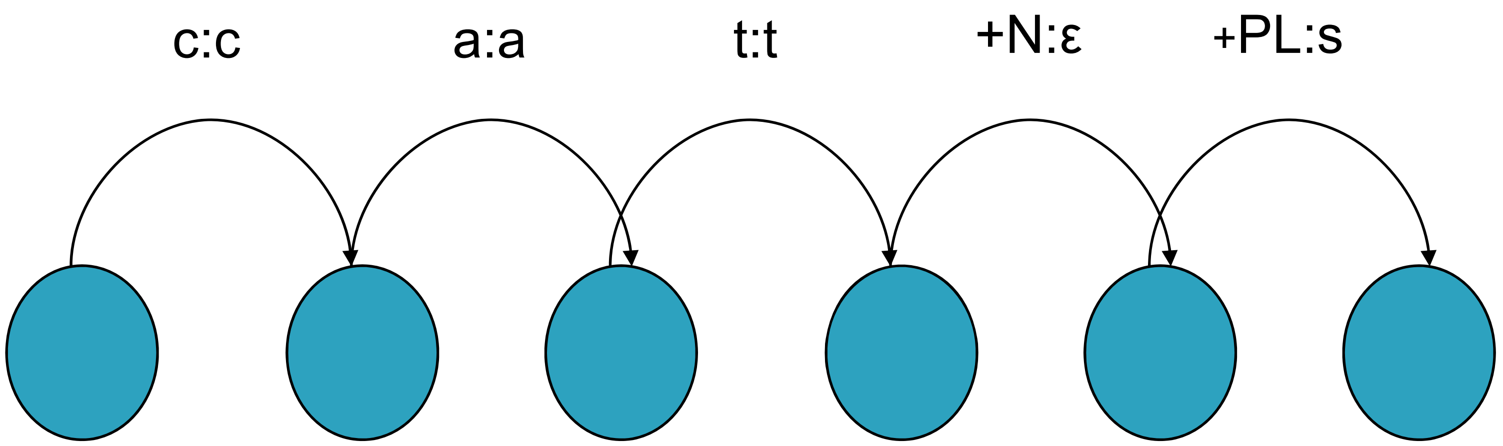
FSTs

Kimmo Koskenniemi's two-level morphology

Idea: word is a relationship between **lexical** level (its morphemes) and **surface** level (its orthography)



Transitions



- c:c means read a c on one tape and write a c on the other
- +N:ϵ means read a +N symbol on one tape and write nothing on the other
- +PL:s means read +PL and write an s

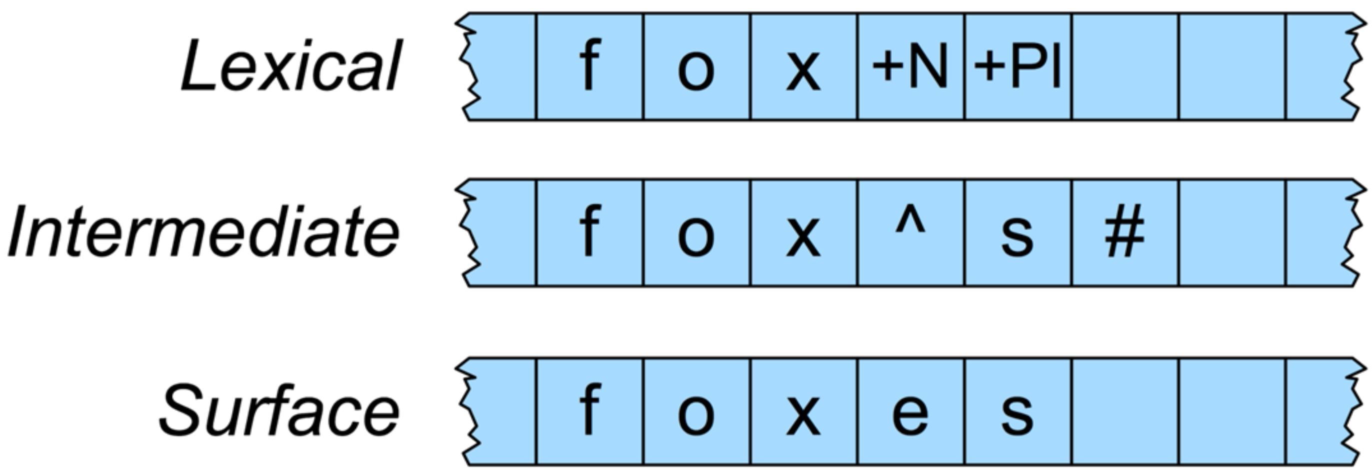
The Gory Details

- 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
  - Fox and Foxes

Multi-Tape Machines

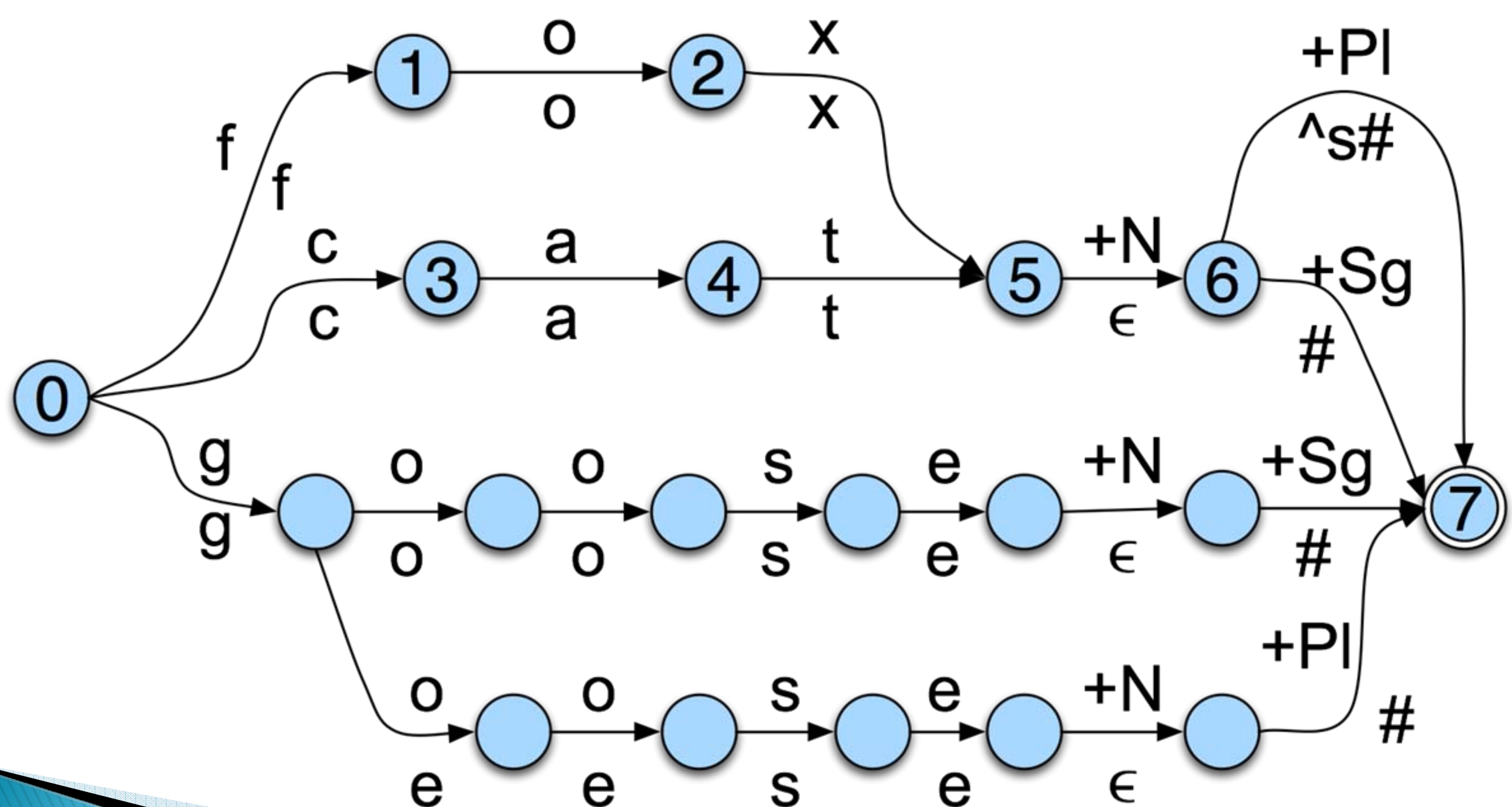
- To deal with this we can simply add more tapes and use the output of one tape machine as the input to the next
- So to handle irregular spelling changes we'll add intermediate tapes with intermediate symbols

Multi-Level Tape Machines



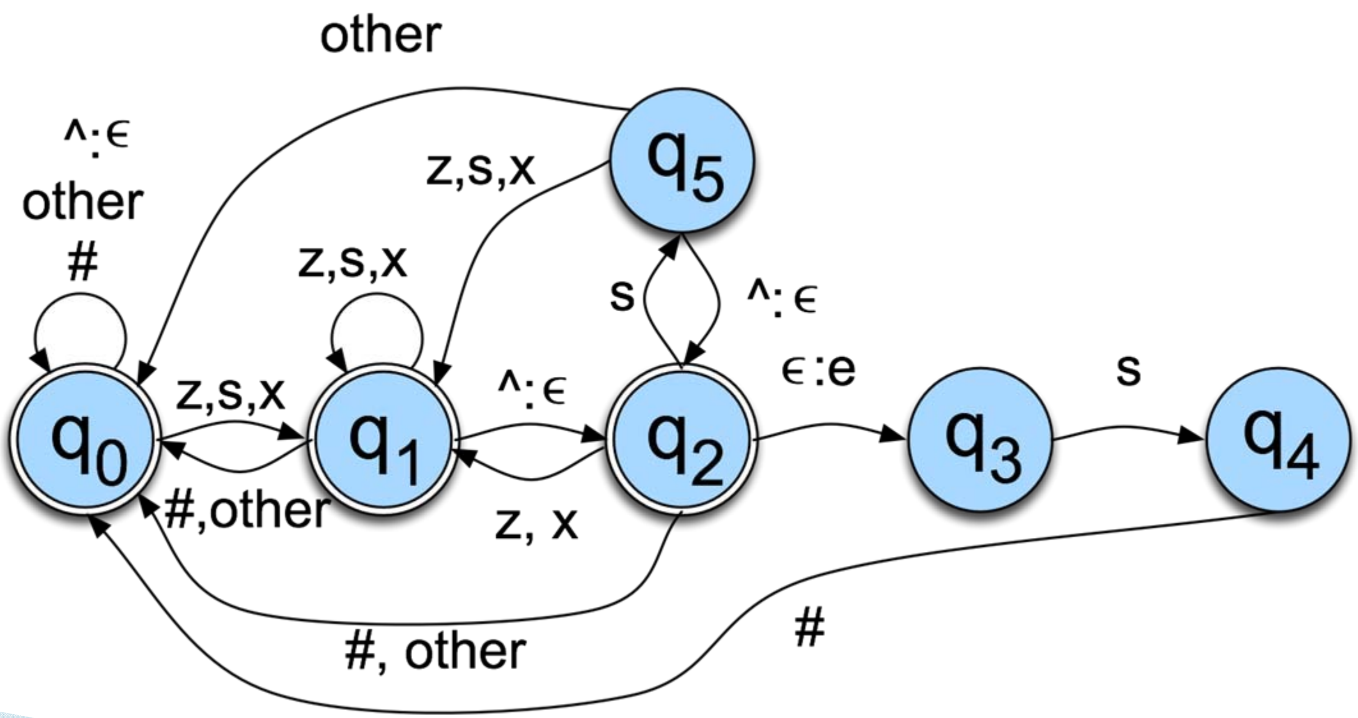
- We use one machine to transduce between the lexical and the intermediate level, and another to handle the spelling changes to the surface tape

Lexical to Intermediate Level

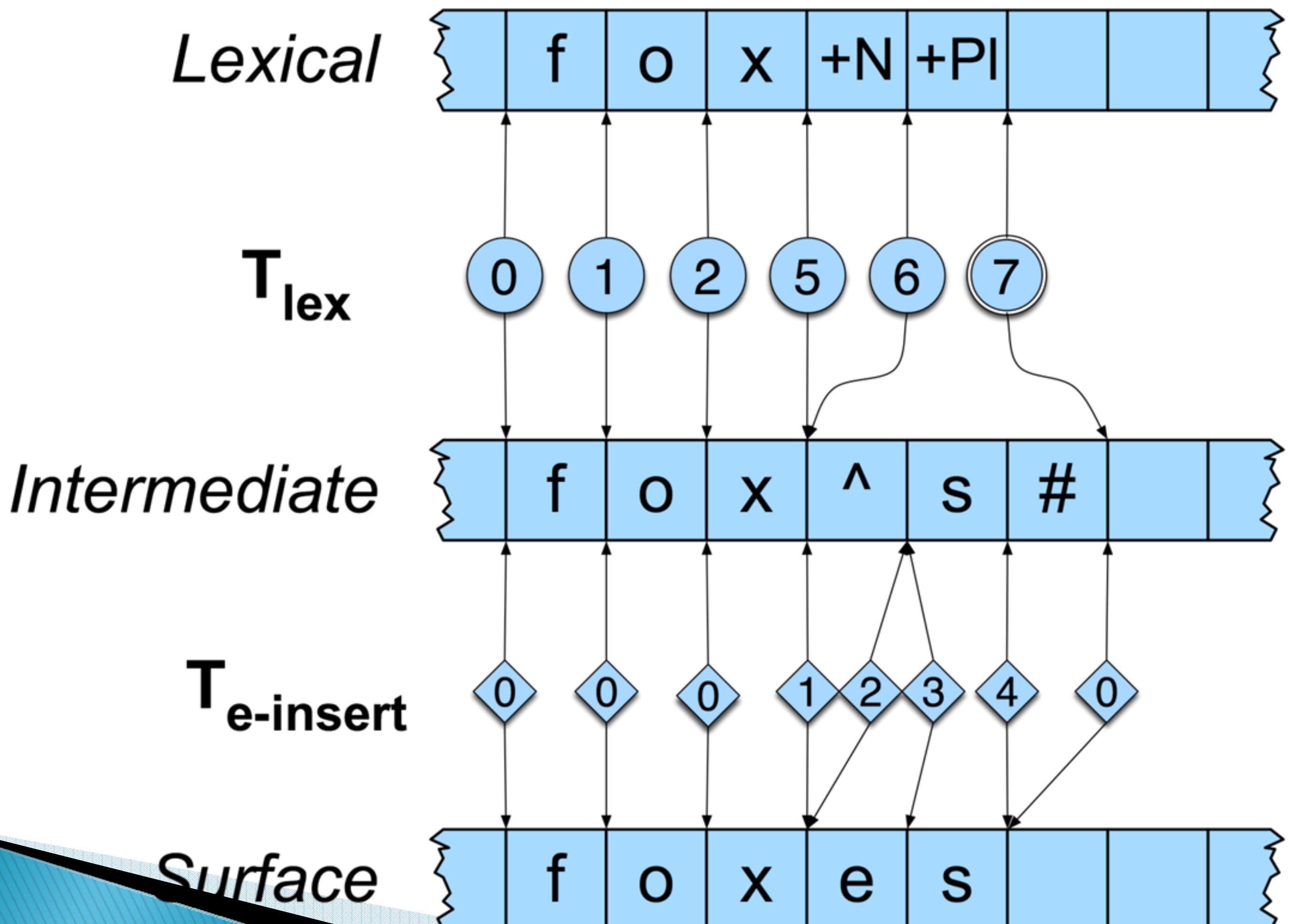


Intermediate to Surface

- The add an “e” 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

Summing Up

- Regular expressions and FSAs can represent subsets of natural language as well as regular languages
  - Both representations may be difficult for humans to understand for any real subset of a language
  - Can be hard to scale up: e.g., when many choices at any point (e.g. surnames)
  - But quick, powerful and easy to use for small problems
- Next class:
  - Read Ch 4