

Finite State Transducers

Morphological Parsing & Tokenization

Mausam

(Based on slides by Jurafsky & Martin,
Julia Hockenmaier)

Morphology: What is a word?

Basic word classes (parts of speech)

Content words (open-class):

- Nouns: student, university, knowledge,...
- Verbs: write, learn, teach,...
- Adjectives: difficult, boring, hard,
- Adverbs: easily, repeatedly,...

Function words (closed-class):

- Prepositions: in, with, under,...
- Conjunctions: and, or,...
- Determiners: a, the, every,...

How many words are there?

The Unix command "`wc -w`" counts the words in a file.

```
> cat example.txt
```

```
This company isn't New York-based anymore  
We moved to Chicago
```

```
> wc -w example.txt  
10 example.txt
```

"`wc -w`" uses blanks to identify words:

```
This1 company2 isn't3 New4 York-based5 anymore6  
We7 moved8 to9 Chicago10
```

Words aren't just defined by blanks

Problem 1: Compounding

“ice cream”, “website”, “web site”, “New York-based”

Problem 2: Other writing systems have no blanks

Chinese: 我开始写小说 = 我 开始 写 小说
I start(ed) writing novel(s)

Problem 3: Clitics

English: “doesn't”, “I'm”,

Italian: “dirglielo” = dir + gli(e) + lo
tell + him + it

How many words are there?

Of course he wants to take the advanced course too.
He already took two beginners' courses.

This is a bad question. Did I mean:

How many **word tokens** are there?

(16 to 19, depending on how we count punctuation)

How many **word types** are there?

(i.e. How many different words are there?)

Again, this depends on how you count, but it's usually much less than the number of tokens)

How many words are there?

Of **course** he wants to **take** the advanced **course** **too**.
He already **took** **two** beginners' **courses**.

The same (underlying) word can take different forms:
course/courses, take/took

We distinguish concrete **word forms** (take, taking)
from abstract **lemmas** or dictionary forms (take)

Different words may be spelled/pronounced the same:
of course vs. advanced course
two vs. too

How many different words are there?

Inflection creates different forms of the same word:

Verbs: to be, being, I am, you are, he is, I was,

Nouns: one book, two books

Derivation creates different words from the same lemma:

grace \Rightarrow disgrace \Rightarrow disgraceful \Rightarrow disgracefully

Compounding combines two words into a new word:

cream \Rightarrow ice cream \Rightarrow ice cream cone \Rightarrow ice cream cone bakery

Word formation is productive:

New words are subject to all of these processes:

Google \Rightarrow Googler, to google, to ungoogle, to misgoogle, googlification, ungooglification, googlified, Google Maps, Google Maps service,...

Lexeme, Lemma, Stem, Root

- A **lexeme** is a unit of lexical meaning underlying a set of words that are related through inflection
- A **lemma** is a word that stands at the head of a definition in a dictionary.
- A **root** is the central (free) morpheme to which other bound morphemes are added to form a word.
- A **stem** is the portion of strings that are common in all the inflections of a word.
 - ◆ *reproduce, reproduces* and *reproducing* are forms of the same lexeme, for which
 - *reproduce* is the lemma.
 - *reproduc* is the stem.
 - *duce(?)* is the root.

Inflectional morphology in English

Verbs:

- Infinitive/present tense: walk, go
- 3rd person singular present tense (s-form): walks, goes
- Simple past: walked, went
- Past participle (ed-form): walked, gone
- Present participle (ing-form): walking, going

Nouns:

- Number: singular (book) vs. plural (books)
- Plural: books
- Possessive (~ genitive case): book's, books
- Personal pronouns inflect for person, number, gender, case:
I saw him; he saw me; you saw her; we saw them; they saw us.

Derivational morphology

Nominalization:

V + -ation: computerization

V+ -er: killer

Adj + -ness: fuzziness

Negation:

un-: undo, unseen, ...

mis-: mistake,...

Adjectivization:

V+ -able: doable

N + -al: national

Morphemes: stems, affixes

dis-grace-ful-ly
prefix-stem-suffix-suffix

Many word forms consist of a **stem** plus a number of **affixes** (*prefixes* or *suffixes*)

Infixes are inserted inside the stem.

Circumfixes (German gesehen) surround the stem

Morphemes: the smallest (meaningful/grammatical) parts of words.

Stems (grace) are often **free morphemes**.

Free morphemes can occur by themselves as words.

Affixes (dis-, -ful, -ly) are usually **bound morphemes**.

Bound morphemes have to combine with others to form words.

Morphemes and morphs

There are many *irregular word forms*:

- Plural nouns add -s to singular: book-books,
but: box-boxes, fly-flies, child-children
- Past tense verbs add -ed to infinitive: walk-walked,
but: like-liked, leap-leapt

Morphemes are abstract categories

Examples: plural morpheme, past tense morpheme

The same morpheme (e.g. for plural nouns) can be realized as *different surface forms (morphs)*:

-s/-es/-ren

Allomorphs: two different realizations (-s/-es/-ren)
of the same underlying morpheme (plural)

Morphological parsing

disgracefully
dis *grace* *ful* *ly*
prefix *stem* *suffix* *suffix*
NEG *grace+N* *+ADJ* *+ADV*

Morphological generation

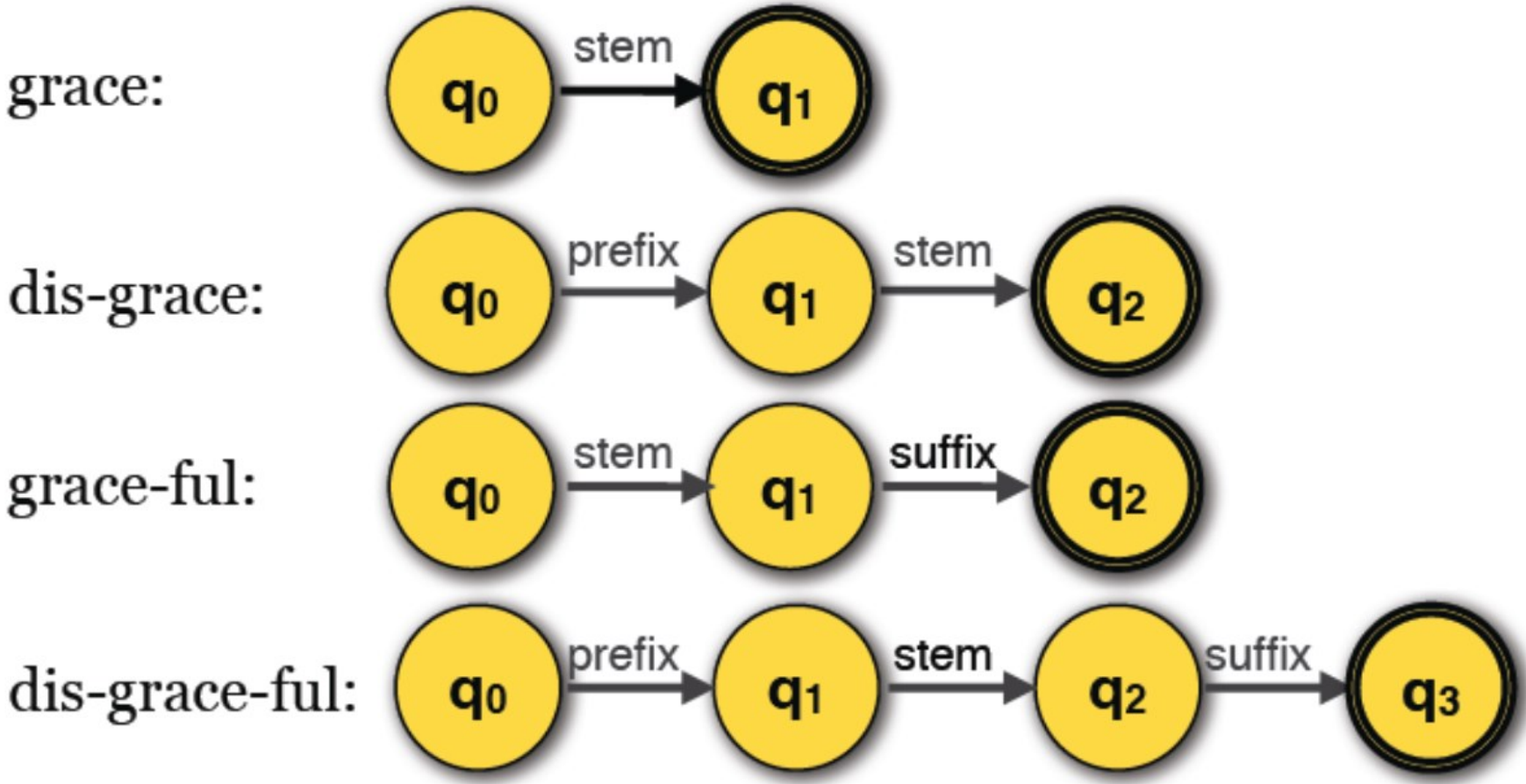
Generate possible English words:

grace, graceful, gracefully
disgrace, disgraceful, disgracefully,
ungraceful, ungracefully,
undisgraceful, undisgracefully,...

Don't generate impossible English words:

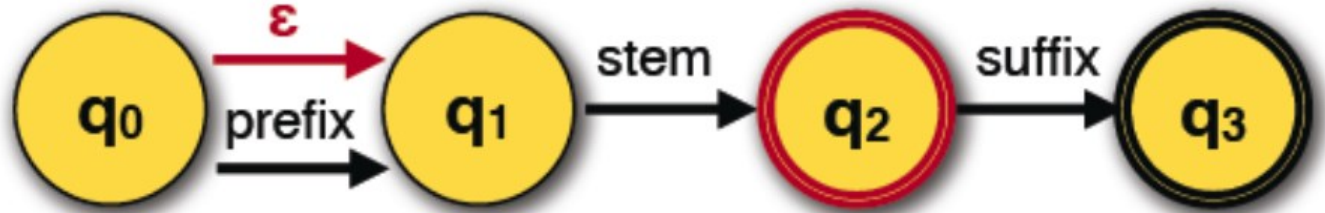
*gracelyful, *gracefuly, *disungracefully,...

Finite state automata for morphology



Union: merging automata

grace,
dis-grace,
grace-ful,
dis-grace-ful



Stem changes

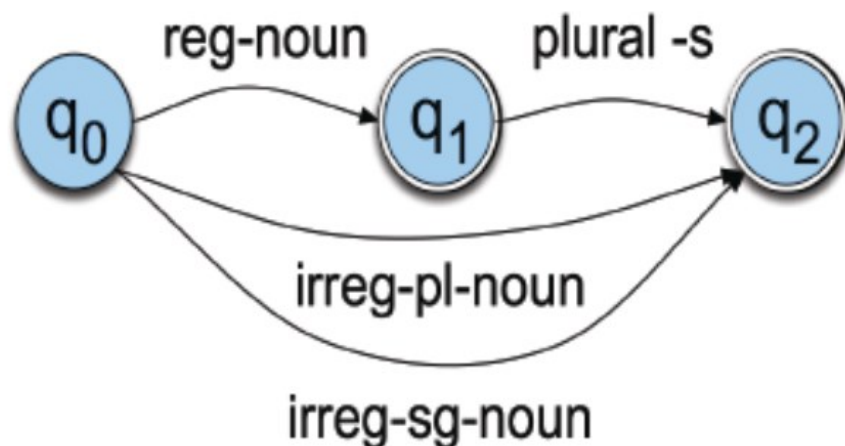
Some irregular words require stem changes:

Past tense verbs:

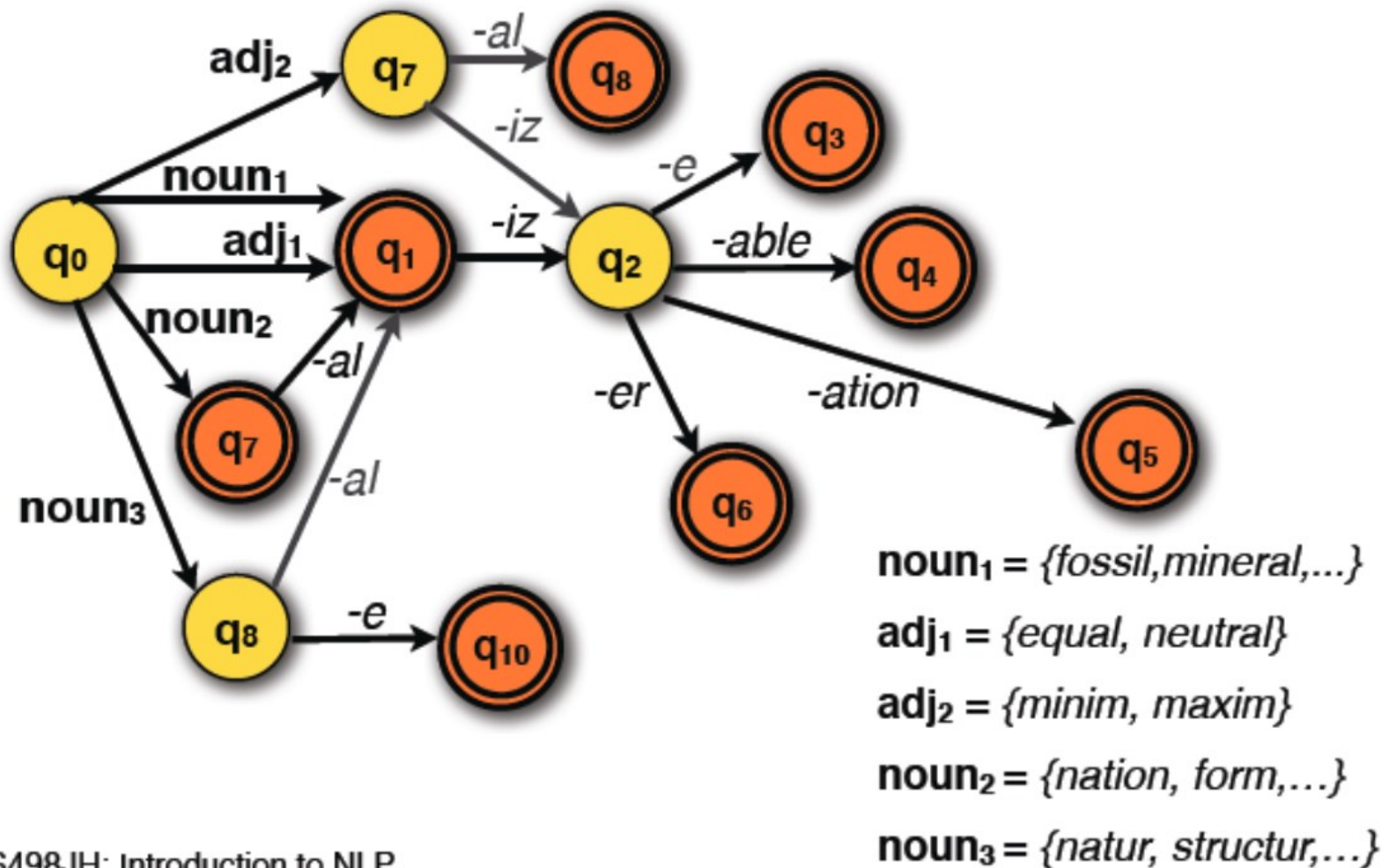
teach-taught, go-went, write-wrote

Plural nouns:

mouse-mice, foot-feet, wife-wives



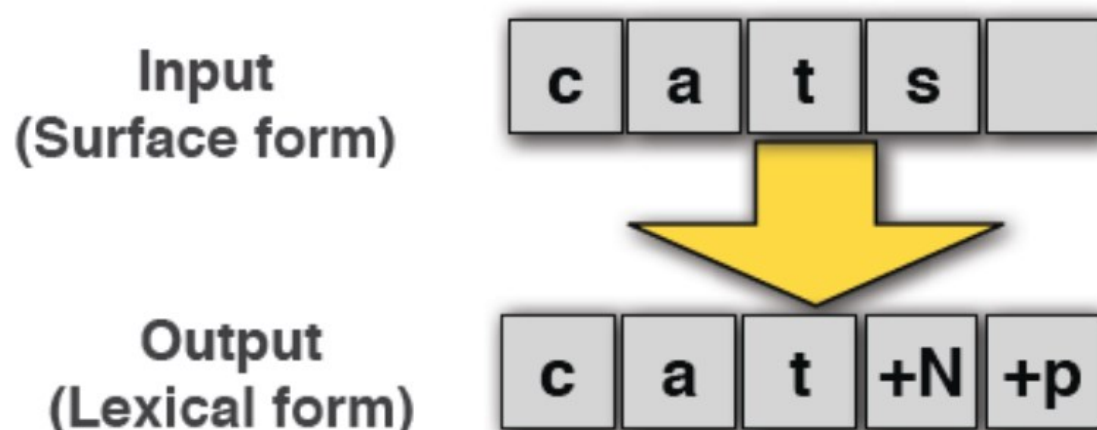
FSAs for derivational morphology



Recognition vs. Analysis

FSAs can recognize (**accept**) a string, but they don't tell us its internal structure.

We need is a machine that maps (**transduces**) the input string into an output string that encodes its structure:



Finite-state transducers

A **finite-state transducer** $T = \langle Q, \Sigma, \Delta, q_0, F, \delta, \sigma \rangle$ consists of:

- A finite set of states $Q = \{q_0, q_1, \dots, q_n\}$
- A finite alphabet Σ of input symbols (e.g. $\Sigma = \{a, b, c, \dots\}$)
- A finite alphabet Δ of output symbols (e.g. $\Delta = \{+N, +pl, \dots\}$)
- A designated start state $q_0 \in Q$
- A set of final states $F \subseteq Q$
- A transition function $\delta: Q \times \Sigma \rightarrow 2^Q$
 $\delta(q, w) = Q'$ for $q \in Q, Q' \subseteq Q, w \in \Sigma$
- An output function $\sigma: Q \times \Sigma \rightarrow \Delta^*$
 $\sigma(q, w) = \omega$ for $q \in Q, w \in \Sigma, \omega \in \Delta^*$

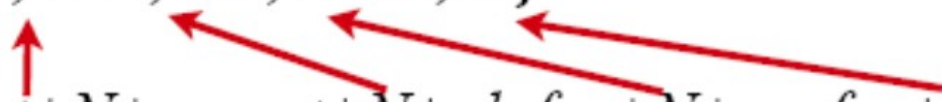
If the current state is q and the current input is w , write ω .

Finite-state transducers

An FST $T = L_{in} \times L_{out}$ defines a relation between two regular languages L_{in} and L_{out} :

$L_{in} = \{\mathbf{cat}, \mathbf{cats}, \mathbf{fox}, \mathbf{foxes}, \dots\}$

$L_{out} = \{cat+N+sg, cat+N+pl, fox+N+sg, fox+N+PL \dots\}$



$T = \{ \langle \mathbf{cat}, cat+N+sg \rangle, \langle \mathbf{cats}, cat+N+pl \rangle, \langle \mathbf{fox}, fox+N+sg \rangle, \langle \mathbf{foxes}, fox+N+pl \rangle \}$

Some FST operations

Inversion T^{-1} :

The inversion (T^{-1}) of a transducer switches input and output labels.

*This can be used to switch from **parsing** words to **generating** words.*

Composition ($T \circ T'$): (Cascade)

Two transducers $T = L_1 \times L_2$ and $T' = L_2 \times L_3$ can be composed into a third transducer $T'' = L_1 \times L_3$.

*Sometimes **intermediate representations** are useful*

English spelling rules

English spelling (orthography) is funny:

The underlying morphemes (*plural-s*, etc.) can have different orthographic surface realizations (-s, -es)

Spelling changes at morpheme boundaries:

- E-insertion: fox +s = fox**e**s
- E-deletion: make**e** +ing = making

Intermediate representations

English plural -s: **cat** \Rightarrow **cats** **dog** \Rightarrow **dogs**
but: **fox** \Rightarrow **foxes**, **bus** \Rightarrow **buses** **buzz** \Rightarrow **buzzes**

We define an **intermediate representation** which captures morpheme boundaries (^) and word boundaries (#):

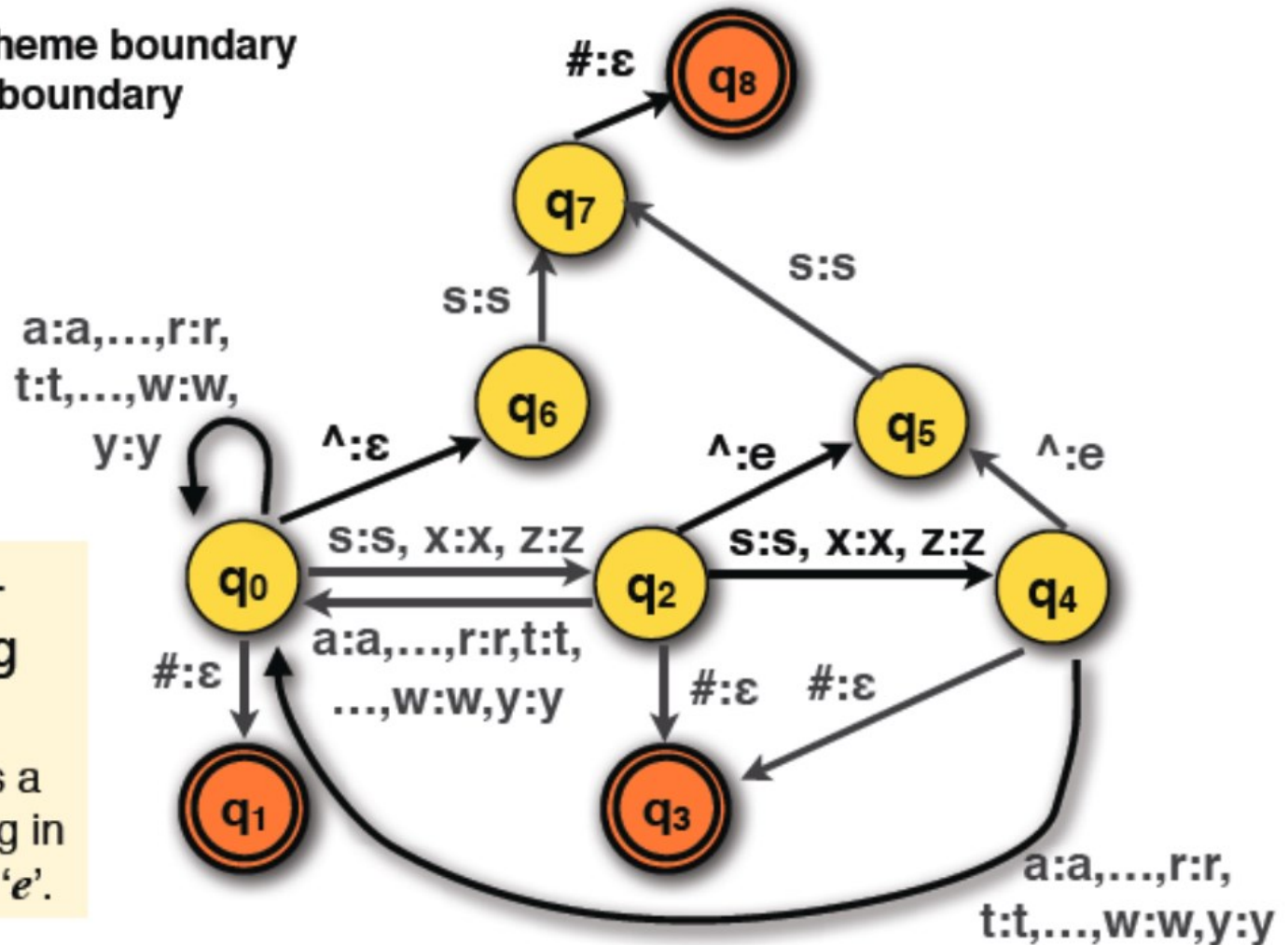
<i>Lexicon:</i>	cat+N+PL	fox+N+PL
\Rightarrow <i>Intermediate representation:</i>	cat^s#	fox^s#
\Rightarrow <i>Surface string:</i>	cats	foxes

Intermediate-to-Surface Spelling Rule:

If plural 's' follows a morpheme ending in 'x', 'z' or 's', insert 'e'.

FST: intermediate to surface level

\wedge = morpheme boundary
= word boundary



Intermediate-to-Surface Spelling Rule:

If plural 's' follows a morpheme ending in 'x', 'z' or 's', insert 'e'.

Dealing with ambiguity

book: book +N +sg or book +V?

Generating words is generally unambiguous, but **analyzing** words often requires disambiguation.

Efficiency problem:

Not every nondeterministic FST can be translated into a deterministic one!

Practical Uses

- This kind of parsing is normally called **morphological analysis**
- Can be
 - An important stand-alone component of an application (**spelling correction, information retrieval, part-of-speech tagging,...**)
 - Or simply a link in a chain of processing (**machine translation, parsing,...**)

FST-based Tokenization

```
#!/usr/bin/perl

$letternumber = "[A-Za-z0-9]";
$notletter = "[^A-Za-z0-9]";
$alwayssep = "[\\?!()\\\";\\/\\|' ]";
$clitic = "(('|:|-|'|S|'D|'M|'LL|'RE|'VE|N'T|'s|'d|'m|'ll|'re|'ve|n't)";

$abbr{"Co."} = 1; $abbr{"Dr."} = 1; $abbr{"Jan."} = 1; $abbr{"Feb."} = 1;

while ($line = <>){ # read the next line from standard input

    # put whitespace around unambiguous separators
    $line =~ s/$alwayssep/ $& /g;

    # put whitespace around commas that aren't inside numbers
    $line =~ s/([0-9]),/$1 , /g;
    $line =~ s/,([0-9])/ , $1/g;

    # distinguish singlequotes from apostrophes by
    # segmenting off single quotes not preceded by letter
    $line =~ s/'/'$& /g;
    $line =~ s/($notletter)'/ $1 ' /g;

    # segment off unambiguous word-final clitics and punctuation
    $line =~ s/$clitic$/ $&/g;
    $line =~ s/$clitic($notletter)/ $1 $2/g;

    # now deal with periods. For each possible word
    @possiblewords=split(/s+/, $line);
    foreach $word (@possiblewords) {
        # if it ends in a period,
        if (($word =~ /$letternumber\./))
            && !($abbr{$word}) # and isn't on the abbreviation list
            # and isn't a sequence of letters and periods (U.S.)
            # and doesn't resemble an abbreviation (no vowels: Inc.)
            && !($word =~
                /^[A-Za-z]\.([A-Za-z]\.)+|[A-Z][bcdfghj-nptvxz]+\.\.$/) {
                # then segment off the period
                $word =~ s/\.$/ \./;
            }
        # expand clitics
        $word =~ s/'ve/have/;
        $word =~ s/'m/am/;
        print $word, " ";
    }
    print "\n";
}
```

Porter Stemmer (1980)

- Common algorithm for stemming English
- Conventions + 5 phases of reductions
 - ◆ phases applied sequentially
 - ◆ each phase consists of a set of commands
 - ◆ sample convention: *Of the rules in a compound command, select the one that applies to the longest suffix.*

Porter Stemmer (1980)

- Standard, very popular and usable **stemmer** (IR, IE) – identify a word's **stem**
- Sequence of cascaded rewrite rules, e.g.
 - ◆ **IZE** → ϵ (e.g. **unionize** → **union**)
 - ◆ **CY** → **T** (e.g. **frequency** → **frequent**)
 - ◆ **ING** → ϵ , if stem contains vowel (**motoring** → **motor**)
- Can be implemented as a lexicon-free FST (many implementations available on the web)
- <http://text-processing.com/demo/stem/>

Eliza

User₁: Men are all alike.

ELIZA₁: IN WHAT WAY

User₂: They're always bugging us about something or other.

ELIZA₂: CAN YOU THINK OF A SPECIFIC EXAMPLE

User₃: Well, my boyfriend made me come here.

ELIZA₃: YOUR BOYFRIEND MADE YOU COME HERE

User₄: He says I'm depressed much of the time.

ELIZA₄: I AM SORRY TO HEAR YOU ARE DEPRESSED

Eliza FST

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

RelNoun: Nominal Open IE

Constructions	Phrase	Extraction
Verb1	Francis Collins is the director of NIH	(Francis Collins; is the director of; NIH)
Verb2	the director of NIH is Francis Collins	(Francis Collins; is the director of; NIH)
Appositive1	Francis Collins, the director of NIH	(Francis Collins; [is] the director of; NIH)
Appositive2	the director of NIH, Francis Collins,	(Francis Collins; [is] the director of; NIH)
Appositive3	Francis Collins, the NIH director	(Francis Collins; [is] the director [of]; NIH)
AppositiveTitle	Francis Collins, the director,	(Francis Collins; [is]; the director)
<i>CompoundNoun</i>	<i>NIH director Francis Collins</i>	<i>(Francis Collins; [is] director [of]; NIH)</i>
Possessive	NIH's director Francis Collins	(Francis Collins; [is] director [of]; NIH)
PossessiveAppositive	NIH's director, Francis Collins	(Francis Collins; [is] director [of]; NIH)
AppositivePossessive	Francis Collins, NIH's director	(Francis Collins; [is] director [of]; NIH)
PossessiveVerb	NIH's director is Francis Collins	(Francis Collins; is director [of]; NIH)
VerbPossessive	Francis Collins is NIH's director	(Francis Collins; is director [of]; NIH)

Compound Noun Extraction Baseline

- NIH Director Francis Collins

(Francis Collins, is the Director of, NIH)

- Challenges

- ◆ New York Banker Association

ORG NAMES

- ◆ German Chancellor Angela Merkel

DEMONYMS

- ◆ Prime Minister Modi

COMPOUND
RELATIONAL NOUNS

- ◆ GM Vice Chairman Bob Lutz

Rule-Based System

- Classifies and filters orgs
- List of demonyms
 - ◆ appropriate location conversion
- Bootstrap a list of relational noun *prefixes*
 - ◆ vice, ex, health, ...

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 use for any real subset of a language
 - ◆ But quick, powerful and easy to use for small problems
- Finite state transducers and rules are common ways to incorporate linguistic ideas in NLP for small applications
- Particularly useful for no data setting