Foundations of Natural Language Processing
Lecture 11
Morphology Parsing

Alex Lascarides
(slides based on those of Shay Cohen)

School of informatics

Last Time

- A range of ML methods that are useful for NLP
- Today: **Morphology**:
  Words aren't the smallest unit!

Morphology

This lecture:

- The problem (some examples)
- Finite State Transducers (FSTs)
- Using FSTs to tackle morphology parsing and generation

Morphology

- Morphology is the study of the *structure of words*
  - English has relatively impoverished morphology.
  - Languages with rich morphology: Turkish, Arabic, Hungarian, Korean . . .

- For example, Turkish is an *agglutinative* language: tends to express concepts in complex words constructed by concatenating *morphemes*, each expressing some simple (component) concept:

evlerinizden: "from your houses" morphemes:	ev-	ler-	iniz-	den
	house	plural	your	from

- This lecture will mostly discuss how to build an English morphological analyser.

Morphology

'Whole' words constructed by combining:

1. Stems (*house, combine, eat, walk,. . .*)
   The 'dictionary' bit

2. Affixes (prefixes, suffixes, infixes and circumfixes)
   grammar parts.
   The type of affix is determined by where it goes with respect to the stem.

Prefix	before the stem
Suffix	after the stem
Infix	middle of the stem
Circumfix	'reduction' of the stem

Four methods to combine them

Inflection (stem + grammar affix): no change to grammatical category
(*walk* $\rightarrow$ *walking*)

Derivation (stem + grammar affix): change to grammatical category
(*combine* $\rightarrow$ *combination*)

Compounding (stems together): *doghouse*

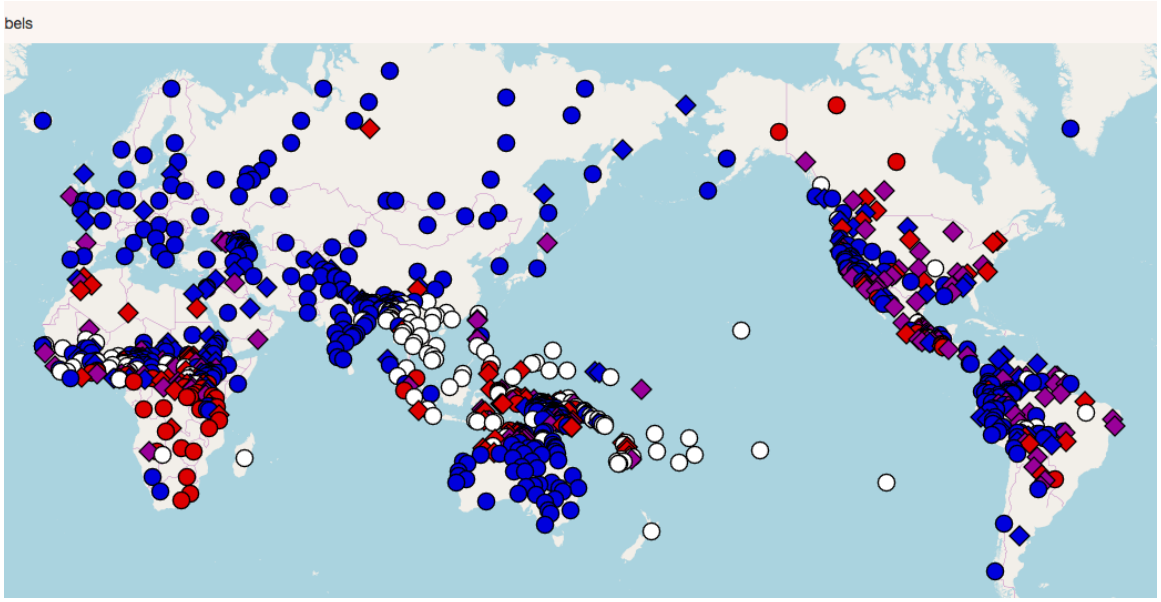
Cliticization: *I've, we're, he's*

Morphology can be concatenative
or non-concatenative (e.g. templatic morphology as in Arabic)

Suffixing versus Prefixing

Values

○	Little affixation	141
●	Strongly suffixing	406
◆	Weakly suffixing	123
◆	Equal prefixing and suffixing	147
◆	Weakly prefixing	94
●	Strong prefixing	58



Different inflection in different languages

- English: nouns are inflected for number; verbs for person and tense
  - *book* (1)/ *books* (> 1)
  - *You read* (2nd pers. present or past)
    she reads (3rd pers. present)
    she read (3rd pers. past)

- German: nouns inflected for number and case:

	Singular	Plural
Nominative	das Haus	die Häuser
Genitive	des Hauses	der Häuser
Dative	dem Haus / dem Hause	den Häusern
Accusative	das Haus	die Häuser

- Spanish: inflection depends on gender (*el sol*/*la luna*)

- Luganda: nouns have ten genders!

Examples: Agglutination and compounding

- *ostoskeskuksessa*
  ostos#keskus+N+Sg+Loc:in
  shopping#center+N+Sg+Loc:in
  'in the shopping center' (Finnish)

- *qangatasuukkuvimmuuriaqalaaqtunga*
  "I'll have to go to the airport" (Inuktitut)

- *Avrupalilaştiramadiklarmizidanmisinizcasina*
  "as if you are reportedly of those of ours that we were unable to Europeanize" (Turkish)

In the most extreme examples, the meaning of the word is the meaning of a sentence!

Morphological parsing: the problem

- English has concatenative morphology. Words can be made up of a main stem plus one or more affixes carrying grammatical information.

Surface form:	cats	walking	smoothest
Lexical form:	cat+N+PL	walk+V+PresPart	smooth+Adj+Sup

- Morphological parsing is the problem of extracting the lexical form from the surface form. (For ASR, this includes identifying the word boundaries.)

- We should take account of:
  - Irregular forms (e.g. goose $\rightarrow$ geese)
  - Systematic rules (e.g. 'e' inserted before suffix 's' after s,x,z,ch,sh: fox $\rightarrow$ foxes, watch $\rightarrow$ watches)
  - Things that look like affixes but aren't (*proactive* vs. *protect*)
  - Blocking: **semi**-productivity of morphological rules: N+*ful* $\mapsto$ Adj (*graceful*, *pityful* . . .), but **intelligenceful* (intelligent), **gloryful* (glorious). . .

Why bother?

- Any NLP tasks involving grammatical parsing will typically involve morphology parsing as a prerequisite.

- Search engines: e.g. a search for 'fox' should return documents containing 'foxes', and vice versa.

- Even a humble task like spell checking can benefit
  e.g. *sleeped* $\rightarrow$ *slept*

Why an exhaustive word list isn't adequate

But why not just list all derived forms separately in our wordlist (e.g. *walk, walks, walked, walking*)?

- Might be OK for English, but not for a morphologically rich language — e.g. in Turkish, can pile up to 10 suffixes on a verb stem, leading to 40,000 possible forms for some verbs!

- Similarly for noun compounding in German, which is highly productive.

- Even for English, morphological parsing makes adding/learning new words easier (and it makes POS tagging easier).

- In speech processing, word breaks aren't known in advance.

How expressive is morphology?

- Morphemes are tacked together in a rather 'regular' way.
- But (in contrast to sentential syntax)
  there are no long range dependencies.
- So finite-state machines are a good way to model morphology.
  - No need for "unbounded memory".

Parsing and generation

Parsing here means going from the surface to the lexical form.

Eg. foxes $\rightarrow$ fox +N +PL.

Generation is the opposite process: fox +N +PL $\rightarrow$ foxes.
It's helpful to consider these two processes together.

Either way, it's often useful to proceed via an intermediate form, corresponding to an analysis in terms of morphemes (= minimal meaningful units) before orthographic rules are applied.

Surface form:	foxes
Intermediate form:	fox ^ s #
Lexical form:	fox +N +PL

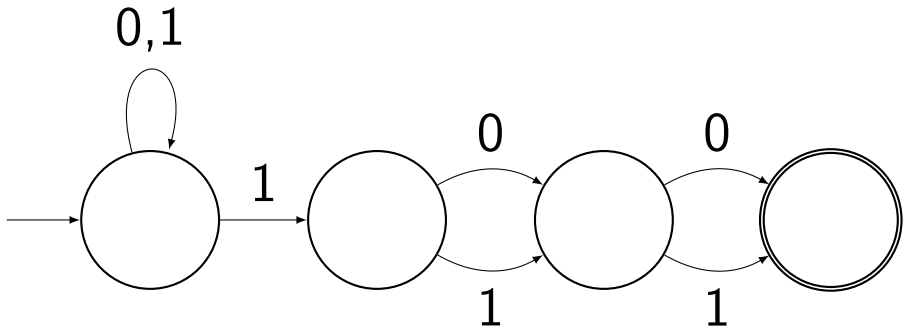
(^ means morpheme boundary, # means word boundary.)

NB. The translation between surface and intermediate form is exactly the same if 'foxes' is a 3rd person singular verb!

Nondeterministic Finite State Automatons (NFAs)

- A nondeterministic finite state automaton (NFA) is a finite state machine where a state can have more than one outgoing arc.

E.g., $(0|1)^*1(0|1)^2$ is captured by the following:

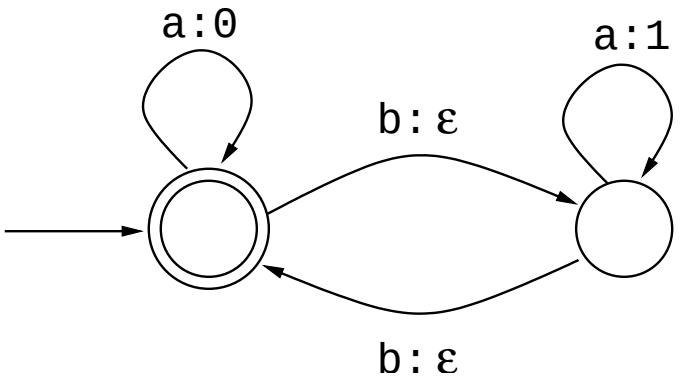


- An ϵ-NFA allows an input (in parsing) or output (in generation) defined by an arc to be the empty string.

Finite-state Transducers

- ϵ-NFAs over an input alphabet Σ can capture transitions that (optionally) produce *output* symbols (over a possibly different alphabet Π).

input alphabet $\Sigma = \{a, b\}$
output alphabet $\Pi = \{0, 1\}$:



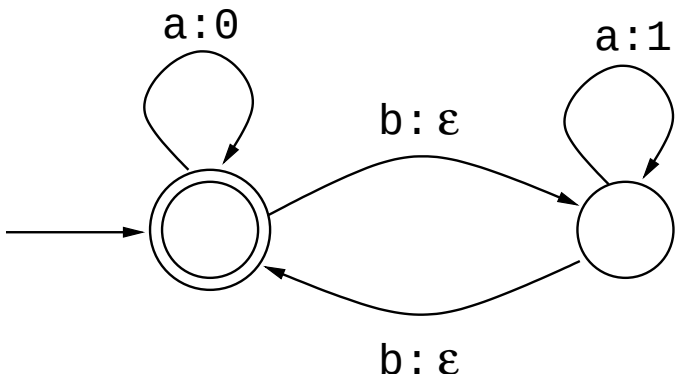
- Such a thing is called a finite state transducer.

- In effect, it specifies a (possibly multi-valued) translation from one regular language to another:

  – $abba \mapsto 00$, $aababa \mapsto 0010$ . . .

- More than one output can be possible (e.g., an arc labelled $a : 0, 1$)

Quick exercise



What output will this produce, given the input $aabaaabbbab$?

1. 001110
2. 001111
3. 0011101
4. More than one output is possible.

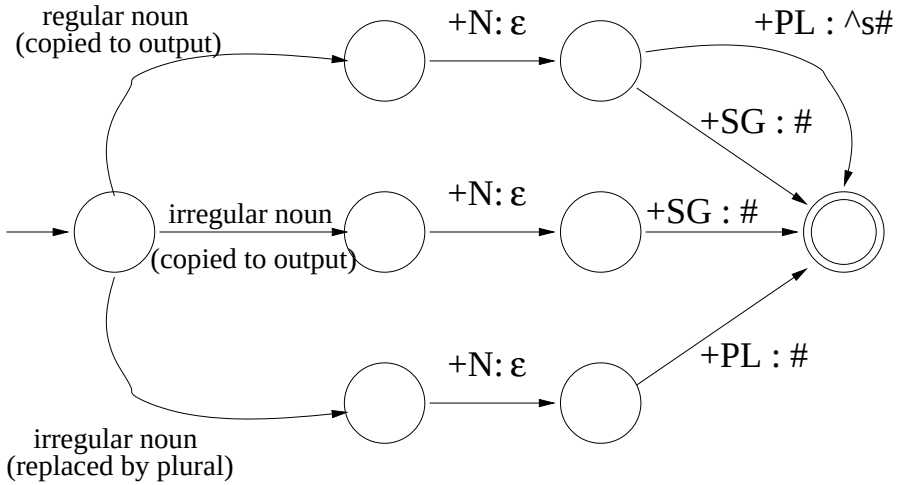
Formal definition

- A finite state transducer T with inputs from Σ and outputs from Π consists of:
  - states Q, S (for start), F (for 'finish')
  - a transition relation $\Delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times (\Pi \cup \{\epsilon\}) \times Q$
- This defines a many-step transition relation $\hat{\Delta} \subseteq Q \times \Sigma^* \times \Pi^* \times Q$
  - $(q, x, y, q') \in \hat{\Delta}$ means "starting from state q, the input string x can be translated into the output string y, ending up in state q'." (Details omitted.)
- A finite state transducer can be run in either direction! From T you can obtain another transducer $\overline{T}$ by swapping inputs and outputs.

Stage 1: From lexical to intermediate form

- Convert 'fox+N+PL' to 'fox ^ s # '
  . . . while taking account of irregular forms like goose/geese.

- We can do this with a transducer of the following schematic form:

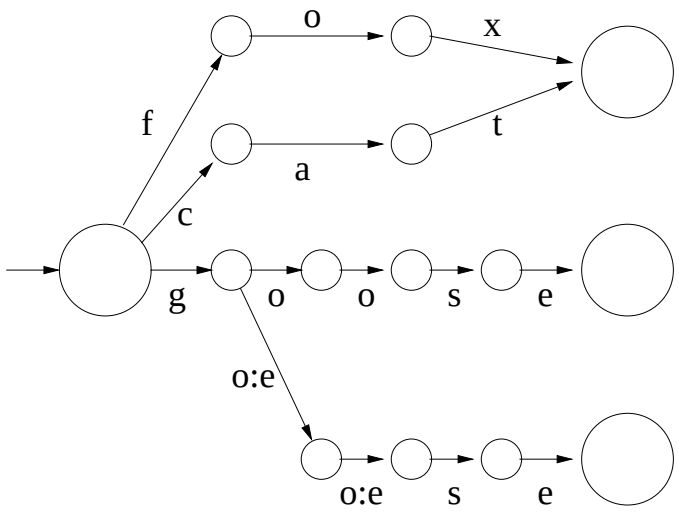


- We treat each of +N, +SG, +PL as a single symbol.

- The 'transition' labelled $+PL : \hat{}s\#$ abbreviates three transitions: $+PL : \hat{}$, $\epsilon : s$, $\epsilon : \#$.

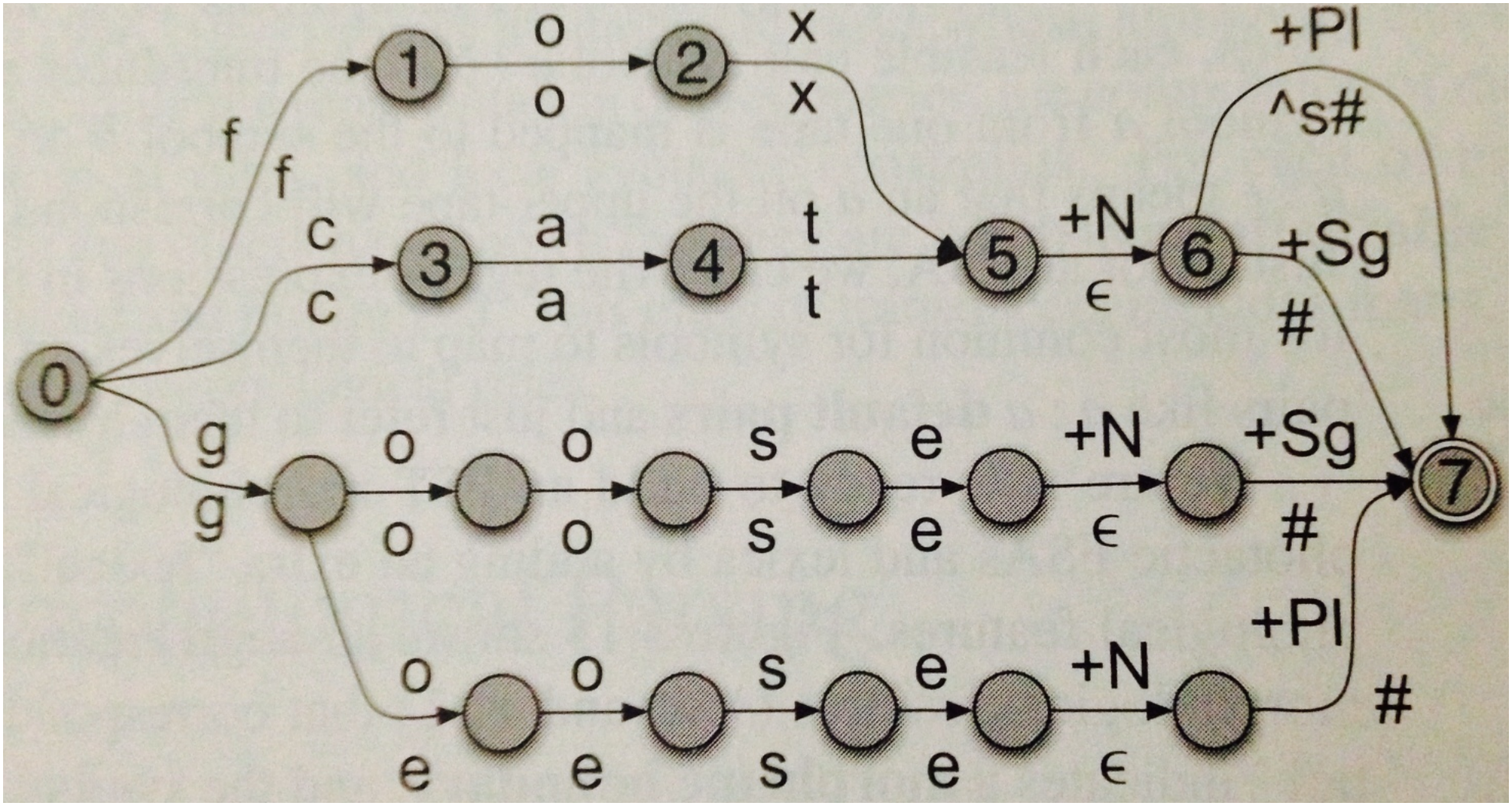
The Stage 1 transducer fleshed out

- The left hand part of the preceding diagram is an abbreviation for something like this (only a small sample shown):



- Here, for simplicity, a single label u abbreviates $u : u$.

Stage 1 in full



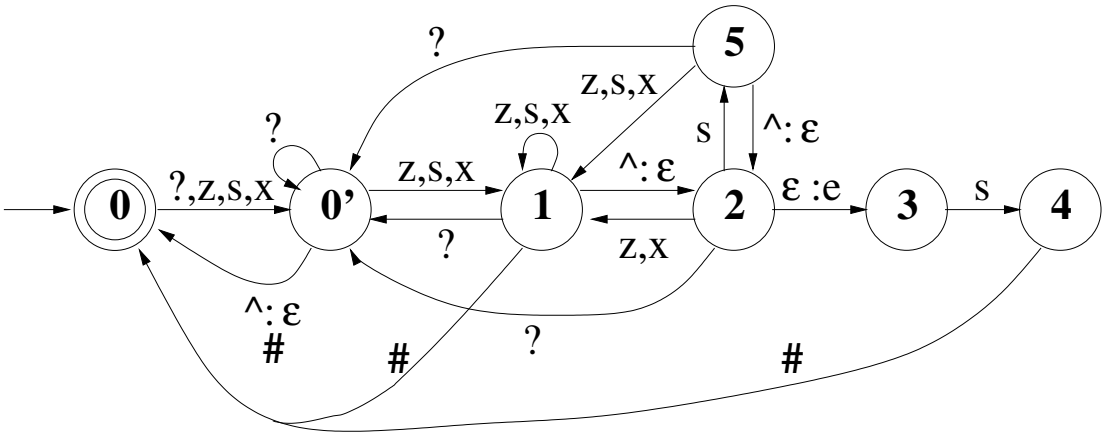
Stage 2: From intermediate to surface form

- To convert a sequence of morphemes to surface form, we apply a number of orthographic rules:
  - E-insertion: Insert e after s,z,x,ch,sh before a word-final morpheme -s.
    (fox $\rightarrow$ foxes)
  - E-deletion: Delete e before a suffix beginning with e,i.
    (love $\rightarrow$ loving)
  - Consonant doubling: Single consonants b,s,g,k,l,m,n,p,r,s,t,v are doubled before suffix -ed or -ing.
    (beg $\rightarrow$ begged)

- We shall consider a simplified form of E-insertion, ignoring ch,sh.

- This rule is oblivious to whether -s is a plural noun suffix or a 3rd person verb suffix!
  fox ^ s # $\rightarrow$ foxes

A transducer for E-insertion (adapted from J+M)



Here ? may stand for any symbol except z,s,x,ˆ,#.

(Treat # as a 'visible space character'.)

At a morpheme boundary following z,s,x, we arrive in State 2.

If the ensuing input sequence is s#, our only option is to go via states 3 and 4. Note that there's no #-transition out of State 5.

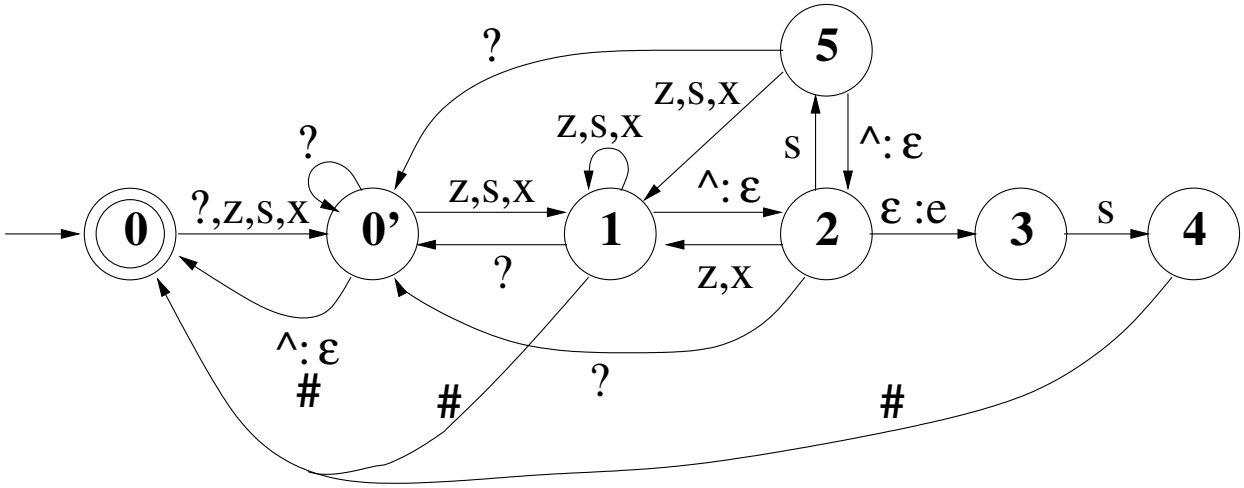
State 5 allows e.g. 'exˆserviceˆmen#' to be translated to 'exservicemen'.

Putting it all together

- FSTs can be cascaded: output from one can be input to another.
- For generation:
  - Stage 1: lexical to intermediate form; then
    Stage 2 (orthographic rules): intermediate to surface form.

  This is typically deterministic (the lexical form yields unique surface form), even though the transducers make use of non-determinism along the way.
- Running the same cascade backwards yields parsing: surface to lexical form.
- Because of ambiguity, this process is frequently non-deterministic:
  e.g. foxes might be analysed as fox+N+PL or fox+V+Pres+3SG.
- Such ambiguities are not resolved by morphological parsing itself: left to a later processing stage.

Example: 10 ways to parse asses!



On the input string 'asses', 10 transition sequences are possible!

- $0 \xrightarrow{a} 0' \xrightarrow{s} 1 \xrightarrow{s} 1 \xrightarrow{\epsilon} 2 \xrightarrow{e} 3 \xrightarrow{s} 4$, output ass^s
- $0 \xrightarrow{a} 0' \xrightarrow{s} 1 \xrightarrow{s} 1 \xrightarrow{\epsilon} 2 \xrightarrow{e} 0' \xrightarrow{s} 1$, output ass^es
- $0 \xrightarrow{a} 0' \xrightarrow{s} 1 \xrightarrow{s} 1 \xrightarrow{e} 0' \xrightarrow{s} 1$, output asses
- $0 \xrightarrow{a} 0' \xrightarrow{s} 1 \xrightarrow{\epsilon} 2 \xrightarrow{s} 5 \xrightarrow{\epsilon} 2 \xrightarrow{e} 3 \xrightarrow{s} 4$, output as^s^s
- $0 \xrightarrow{a} 0' \xrightarrow{s} 1 \xrightarrow{\epsilon} 2 \xrightarrow{s} 5 \xrightarrow{\epsilon} 2 \xrightarrow{e} 0' \xrightarrow{s} 1$, output as^s^es
- $0 \xrightarrow{a} 0' \xrightarrow{s} 1 \xrightarrow{\epsilon} 2 \xrightarrow{s} 5 \xrightarrow{e} 0' \xrightarrow{s} 1$, output as^ses
- Four of these can also be followed by $1 \xrightarrow{\epsilon} 2$ (output ^).

The Porter Stemmer

- Lexicon can be quite large with finite state transducers
- Sometimes need to extract the stem in a very efficient fashion (such as in IR)
- The Porter stemmer: a lexicon-free method for getting the stem of a given word:
  - ATIONAL $\rightarrow$ ATE (e.g., relational $\rightarrow$ relate)
    ING $\rightarrow \epsilon$ if stem comtains a vowel (e.g. motoring $\rightarrow$ motor)
    SSES $\rightarrow$ SS (e.g., grasses $\rightarrow$ grass)
- Makes errors:
  - organization $\rightarrow$ organ (computerization $\rightarrow$ computer)
    policy $\rightarrow$ police (juicy $\rightarrow$ juice)

Learning Morphological Parsers

- A vibrant area of study
- Two main paradigms: unsupervised and supervised.
- Results make errors!
  - Very good for English
    For other languages a long way to go!

  Sometimes ambiguity can't be resolved.

Summary

- Words have structure: stem + affixes
- Affixes can carry meaning and affect grammatical category.
- Languages vary on how productive, and extensive, morphology is.
- Non-deterministic Finite State Transducers (NFTs) can parse and generate morphologically complex words.
  - In effect, a compact repesentation of millions of possible word forms!
- NFTs aren't designed to resolve morphological ambiguities (no representation of context).
- Whether foxes is fox+N+PL or fox+V+3SG depends on words in its neighbourhood. . .

Next class: Part-of-speech tagging