



VES INSTITUTE OF TECHNOLOGY  
DEPARTMENT OF COMPUTER ENGINEERING

# Morphology

By Vidya Zope, VESIT

# What is Morphology?

- Study of Words
  - Their **internal structure**

washing → wash + -ing

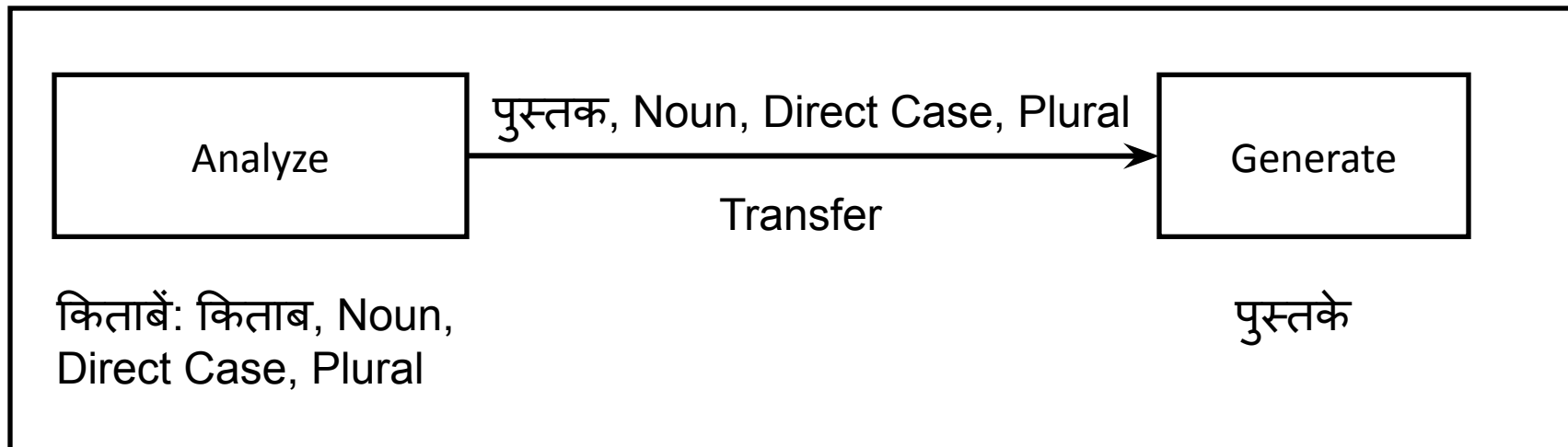
- How they are **formed**?

bat	→	bats	∴	rat	→	rats
write	→	writer	∴	browse	→	browser

- Morphology tries to formulate rules

# Morphology for NLP

- Machine Translation



- Information Retrieval

- goose and geese are two words referring to the same root goose

# Need of MA

## Efficiency

– Listing all of the plural forms of English nouns, all of the verb forms for a particular stem, etc...is a waste of space (and time if the entries are being made by hand).      Suffixes are productive

- Why not list all the forms of a word along with their features?
  - Drink:
    - drink, V, 1<sup>st</sup> person
    - drink, V, 2<sup>nd</sup> person
    - drink, V, 3<sup>rd</sup> person, plural
  - Drinks: drink, V, 3<sup>rd</sup> person, singular
  - Drank: ...
  - Drunk: ...
  - Drinking: ...

# Need of MA

- Reasons:
  - Productivity: going, drinking, running, playing
    - Storing every form leads to inefficiency
  - Addition of new words
    - Verb: To fax. Forms: fax, **faxes**, faxed, faxing
  - Morphological complex languages: Marathi
    - दारासमोरच्यांनी – दार(SG)+समोर+चा(PL)+नी  
Meaning: दरवाजे के सामने वालों ने
    - Polymorphemic
    - Possible to store all the forms?

# Need for morphological analysis

Information retrieval: search

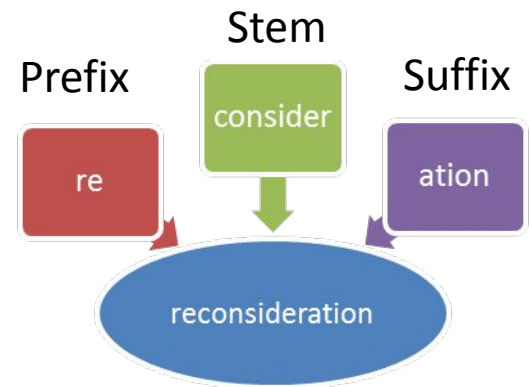
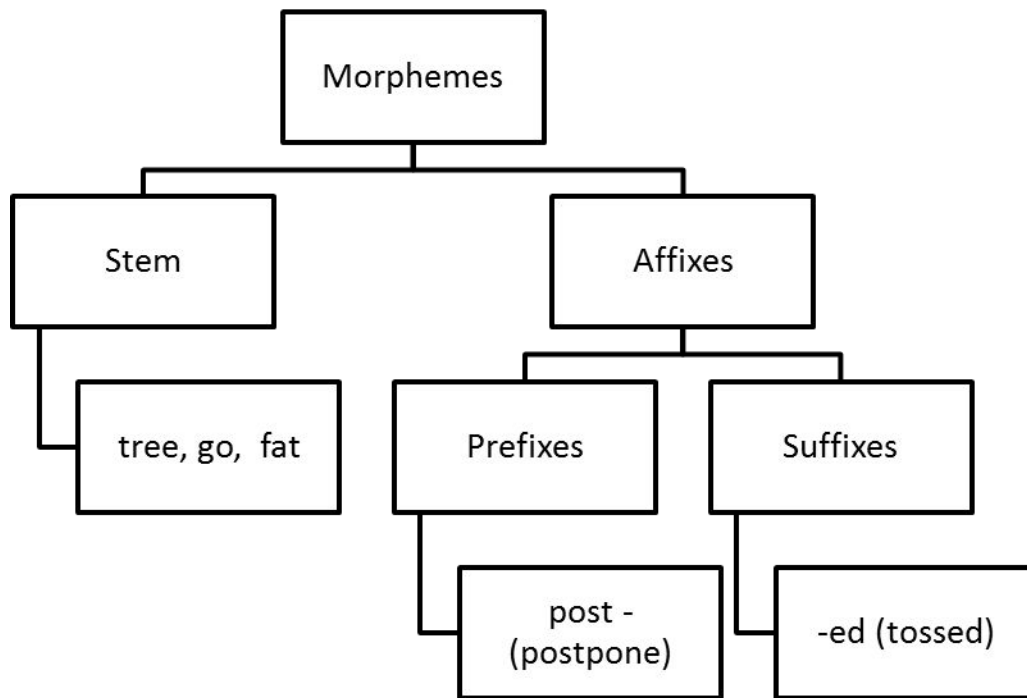
- Systems benefit from being able to search for singular and plural forms of search terms
- Generally fairly easy in English
- Complications
  - » Irregular plurals handled via morphological rules  
goose-> geese  
fish -> fish                      ox -> oxen
  - » Spelling rules needed  
fox +PL -> foxes      fly +PL -> flies

# Morphological analysis python softwares

- **ParaMorfo**: This is used to perform the morphological generation and analysis of Spanish and Guarani
- **HornMorpho**: This is used for the morphological generation and analysis of Oromo and Amharic nouns and verbs as well as Tigrinya verbs
- **AntiMorfo**: This is used for the morphological generation and analysis of Quechua adjectives, verbs, and nouns as well as Spanish verbs
- **MorfoMelayu**: This is used for the morphological analysis of Malay words

# Morphemes

- Smallest meaning bearing units constituting a word





# Turkish Inflectional Morphology

- Some of inflectional suffixes that Turkish nouns can have:
  - singular/plural : masa / masalar
  - possessive markers : masam / masan / masası / masamız / masanız / masaları
  - case markers :
    - ablative : masadan
    - accusative : masayı
    - dative : masaya
- Some of inflectional suffixes that Turkish verbs can have:
  - tense : gel / geldi / geliyor / gelmiş / gelecek
  - second tense : geliyordu / gelmişti / gelecekti
  - agreement marker : geldim / geldin / geldi / geldik / geldiniz / geldiler
- There are order among inflectional suffixes (morphotactics )
  - masalarımından -- masa +PLU +P1SG +ABL
  - geliyordum -- gel +PROG +PAST +1SG

# Full morph. parsing

English		Spanish		
Input	Morphologically Parsed Output	Input	Morphologically Parsed Output	Gloss
cats	cat +N +PL	pavos	pavo +N +Masc +Pl	'ducks'
cat	cat +N +SG	pavo	pavo +N +Masc +Sg	'duck'
cities	city +N +Pl	bebo	beber +V +PInd +1P +Sg	'I drink'
geese	goose +N +Pl	canto	cantar +V +PInd +1P +Sg	'I sing'
goose	goose +N +Sg	canto	canto +N +Masc +Sg	'song'
goose	goose +V	puse	poner +V +Perf +1P +Sg	'I was able'
gooses	goose +V +1P +Sg	vino	venir +V +Perf +3P +Sg	'he/she came'
merging	merge +V +PresPart	vino	vino +N +Masc +Sg	'wine'
caught	catch +V +PastPart	lugar	lugar +N +Masc +Sg	'place'
caught	catch +V +Past			

**Figure 3.2** Output of a morphological parse for some English and Spanish words. Spanish output modified from the Xerox XRCE finite-state language tools.

- Need
  - 1. Lexicon of stems and affixes (maybe guess stems...)
  - 2. Morphotactics: morpheme ordering model
  - 3. Orthographic rules: spelling changes upon combination (city + -s --> citys -> cities)

# Classes of Morphology

- Inflection
- Derivation

- Compounding
  - *baseball desktop*
- Cliticization

Full Form	Clitic	Full Form	Clitic
am	'm	have	've
are	're	has	's
is	's	had	'd
will	'll	would	'd

# Inflection

- Indicates some grammatical function like

Case	लड़का (D)	लड़के (O)
Number	लड़का (Sg)	लड़के (Pl)
Person	जाऊँगा (1st)	जाओगे (2nd)
Gender	जाऊँगा(Masc)	जाऊँगी (Fem)
Tense	गया(Pas)	जाऊँगी (Fem)

Results in a word of the same class

- Productivity

# Characteristics of Inflectional Morphemes

1. **Do not change basic meaning** for part of speech, e.g. big, bigger, biggest
2. Express **grammatically** required features or indicate relations between different words in the sentence. Thus, in Lee loves Kim '-s' marks 3rd person singular.
3. The plural morpheme can be **combined with nearly any nouns**, usually in the same form, and usually with the same effect on meaning. For example the suffix –able is generally **added** to verb, to form adjectives as in changeable.
4. Occur outside any derivational morphemes.  
ration-al-iz-ations the **final 's' is inflectional** and appears at the very end of the word, outside the derivational morphemes aliz-ation.

# Derivation

- Usually, results in a word of a different class
  - -able when attached to a verb gives an adjective
  - read (V) + -able = readable (Adj)

• happy – happiness  
(adjective) (noun)

• examine – examination  
(verb) (noun)

• beauty – beautiful – beautifully  
(noun) (adjective) (adverb)

• danger – dangerous  
(noun) (adjective)

**visible – invisible**  
(Adjective) (Adjective)

**create – recreate**  
(noun) (noun)

**market – supermarket**  
(noun) (noun)

**terminate – determinate**  
(verb) (verb)

# Characteristics of Derivational Morphemes

**1. Change the part of speech or the basic meaning of a word.**

Thus, -ment added to a verb forms a noun judgment while re- added to a verb changes its meaning only as in re-activate which means activate again.

**2. Not required by syntactic relation outside the word.** Thus, unkind combines un- and kind into a single new word, but has no particular syntactic connections outside the word.

**3. Typically occur between the stem and any inflectional affixes.** Thus, in government, -ment; the derivational suffix precedes –s inflectional suffix.

# English Morphemes

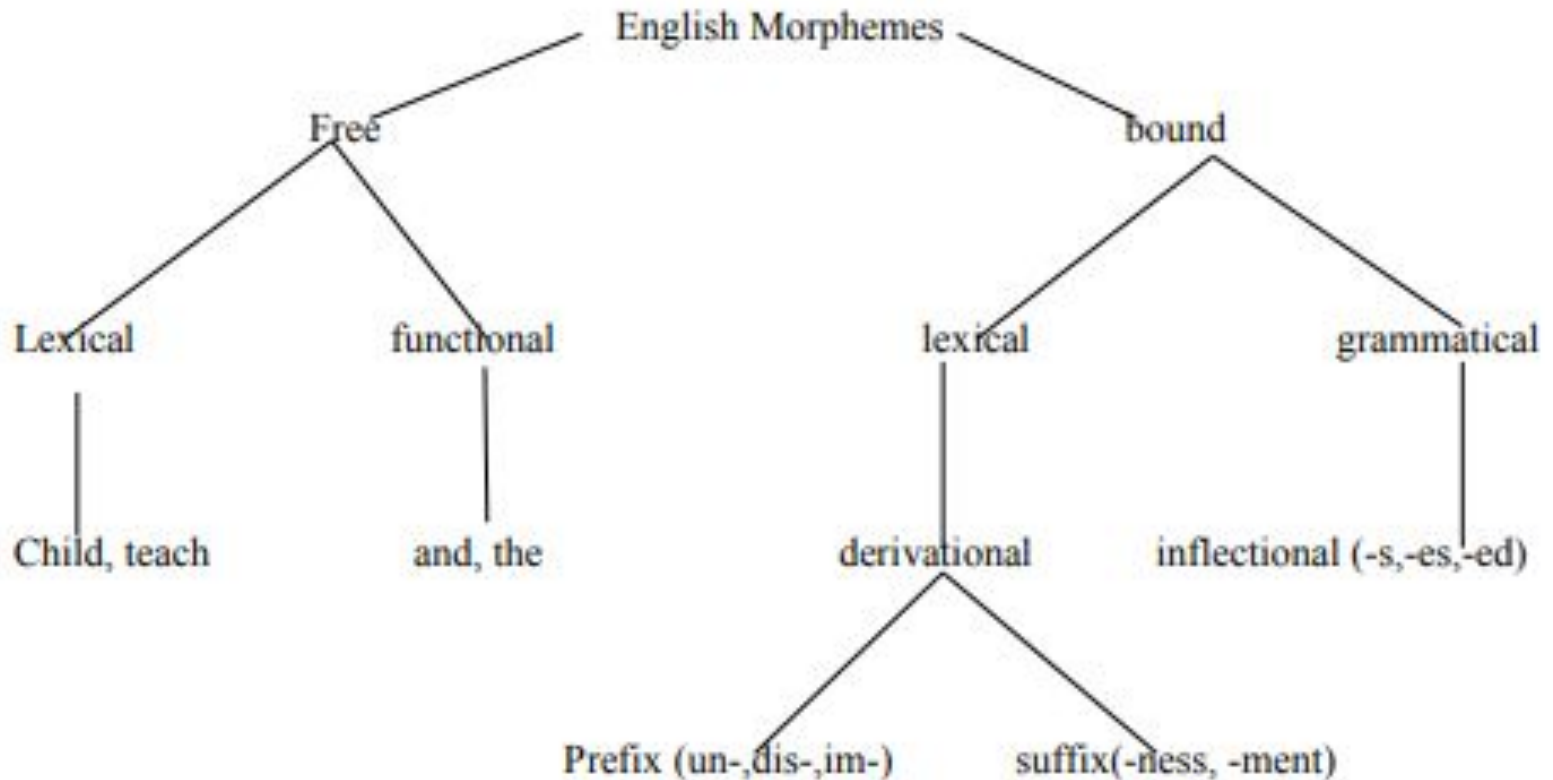


Figure 2.1 English Morpheme

Source: Yule,(1996:78).



# Word Formation Process

a word is a linguistic unit deserves some attention, because it is not as straight forward as one might expect"

Yule, (1996:64-69) summarizes the ways how words are formed as follows:

1\ Derivation: adding a derivational affix, to change the syntactic category, orient > *orientation, boring, quickly.*

2\ Compound: combining two words to make a new one for example: *black-board, notebook, blueberry, and workroom.*

3\ Acronym: a word formed by forming the initial letters of a name, such as *FAQ, NATO, and NASA*. It is pronounced as a word if the consonant and vowel line up in such way as to make this possible. For example:

NATO: North Atlantic Treaty Organization.

NASA: National Aeronautics and Space Administration.

FAQ: Frequently Asked Question.

4\ Blending: Two words joined together e.g. *smoke + fog > smog, motor + hotel > motel, spoon + fork > spork, breakfast + lunch > brunch.*

5\ Clipping: shortening a word: *brother > bro, examination > exam, gasoline > gas.*

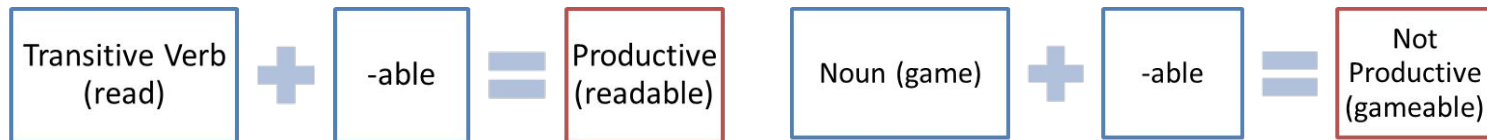
6\ Abbreviation: a little like clipping: *television > TV, April > Apr, Doctor > Dr, Bachelor of Arts > BA.*

# Problems in MA

- Productivity
- False Analysis
- Bound Base Morphemes

# Productivity

- Property of a morphological process to give rise to new formations on a systematic basis
- morpheme un- is unlike -able does not change the grammatical category of the stem it attaches to.... unbelievable.



## Exceptions

Peaceable	Actionable	Companionable
Saleable	Marriageable	Reasonable
Impressionable	Fashionable	knowledgeable

- Less Productive: -ity (e.g. brevity, specificity)
- Non Productive: -ceive, -mit, -cede suffix for words

# False analysis

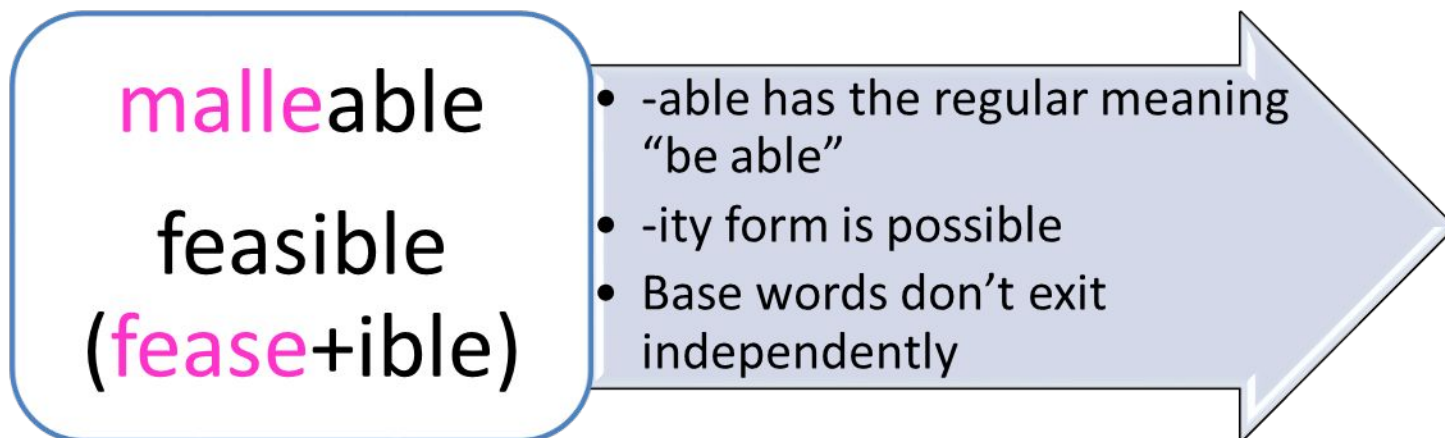
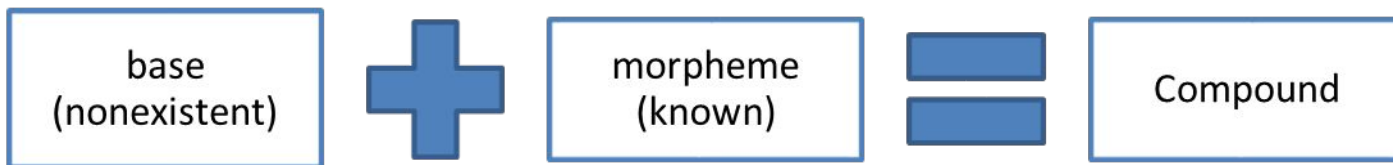
hospit<sup>able</sup>, size<sup>able</sup>

They don't have the meaning "to be able"  
They can not take the suffix -ity to form a noun

Analyzing them as the words containing suffix  
-able leads to false analysis

# Bound Base Morphemes

- Occur only in a particular complex word
- Do not have independent existence



# More on Inflection

Noun inflectional suffixes	<ul style="list-style-type: none"><li>•Plural marker -s</li><li>•Possessive marker 's</li></ul>
Verb inflectional suffixes	<ul style="list-style-type: none"><li>•Third person present singular marker -s</li><li>•Past tense marker -ed</li><li>•Progressive marker -ing</li><li>•Past participle markers -en or -ed</li></ul>
Adjective inflectional suffixes	<ul style="list-style-type: none"><li>•Comparative marker -er</li><li>•Superlative marker -est</li></ul>

Inflectional Suffixes in English

# Prefixes of Number

Prefix	Meaning(s)	Example(s)
bi-	Two	Bilingual
centi-	Hundred	Centimeter
deca-	Ten	Decade
deci-	One tenth part of	Decimeter
di-	Two	Digraph
hemi-	Half	Hemimorphic
mono-	One, single	Monoplane
multi-	Many	Multiracial
oct-	Eight	Octagon
penta-	Five	Pentagon
poly-	Many	Polygamy
semi-	Half	Semidetached
tri-	Three	Trimaran
uni-	One	Unilateral

Sources: Schutz, (2005:12).

**Pejorative Prefixes** describe things as "bad" or "not genuine"

Prefix	Meaning(s)	Example(s)
dys-	Bad, disordered, difficult	Dysentery
mal-	Badly	Malnutrition
mis-	Wrongly	Misfortune
Pseudo-	False, imitation	Pseudodemocracy

Sources: Quirk et al, (1989:432).

### Prefixes of Location and Distance

Prefix	Meaning(s)	Example(s)
ad-	To, toward	Adhoc
amphi-	Around, on both sides of	Amphibious
ana-	Up, back	Anatomy
circum-	Around	Circumference
extra-	Outside,, beyond	Extraordinary
fore-	Before, ahead	Foreshore
infra-	Below	Infrared
inter-	Between, among	Interaction
intra-	Within, into	Intravenous
super-	Over, above	Superhuman
trans-	Across, through	Transfer

Sources: Sinclair (2002:15)



# Infixation

A term used in morphology referring to an affix which is added within a stem.

For example, *cupful*, *spoonful*, and *passerby* can be pluralized as *cupsful*, *spoonsful*, and *passersby*, using "s" as an infix

# Zero Affixation

transposes a lexeme from one word class to another without infixation has been referred to as conversion or zero derivation.

New words may be formed of the input word that serves as the base. A derivational process that involves no affixation, i.e., many words in English can function as a noun and verb, or noun and verb, or noun and adjectives.

Examples: What's the answer? (noun)

Answer the question? (verb)

I must clean my room. (verb)

It's a clean room. (adjective)

# Spelling Rules

- Generally words are pluralized by adding –s to the end
- Words ending in –s, -z, -sh and sometimes –x require –es
  - buses, quizzes, dishes, boxes
- Nouns ending in –y preceded by a consonant change the –y to -i
  - babies, floppies

# Verbal Inflection

Morphological Form Classes	Regularly Inflected Verbs				Irregularly Inflected Verbs		
Stem	Jump	Parse	Fry	Sob	Eat	Bring	Cut
-s form	Jumps	Parses	Fries	Sobs	Eats	Brings	Cuts
-ing participle	Jumping	Parsing	Frying	Sobbing	Eating	Bringing	Cutting
Past form	Jumped	Parsed	Fried	Sobbed	Ate	Brought	Cut
-ed participle	Jumped	Parsed	Fried	Sobbed	Eaten	Brought	Cut

Forms governed by spelling rules

Idiosyncratic forms

# Morphological Parsing

- Finding
  - Constituent morphemes
  - Features

Input	Morphological Parsed Output
cats	cat +N +PL
geese	goose +N +PL
goose	(goose +N +SG) or (goose +V)
gooses	goose +V +3G
caught	(catch +V +PAST-PART) or (catch +V +PAST)

# Resources

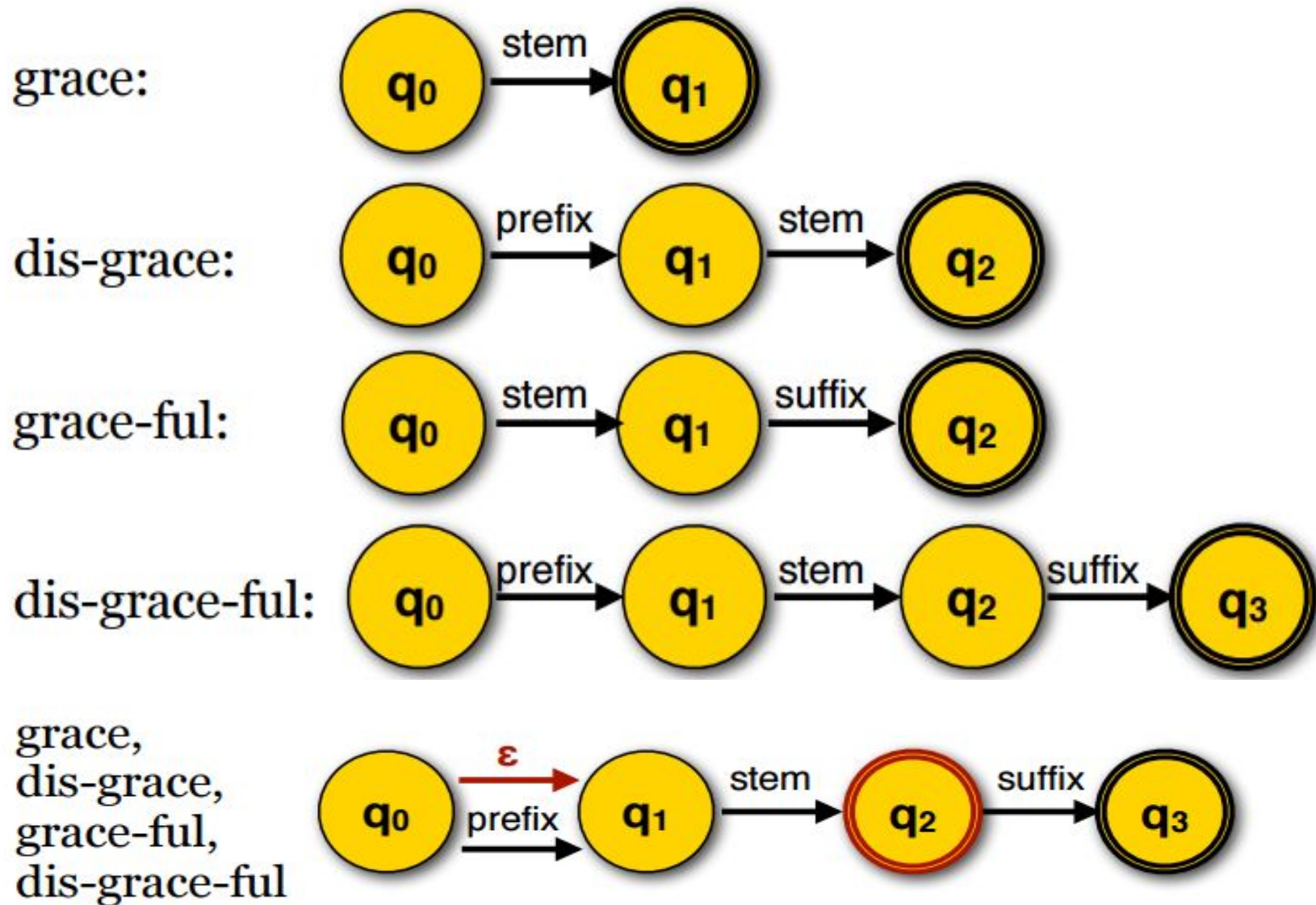
Lexicon	List of stems and suffixes along with basic information about them
Morphotactics	A model of morpheme ordering that explains which classes of morphemes can follow other classes of morphemes
Orthographic Rules	Spelling rules used to model the changes that occur in the work usually when two morphemes combine

# Morphological Recognition

## Lexicon

reg-noun	irregular-sg-noun	irregular-pl-noun	plural
flower	goose	geese	-s
cat	sheep	sheep	
dog	mouse	mice	

# Finite state automata for morphology



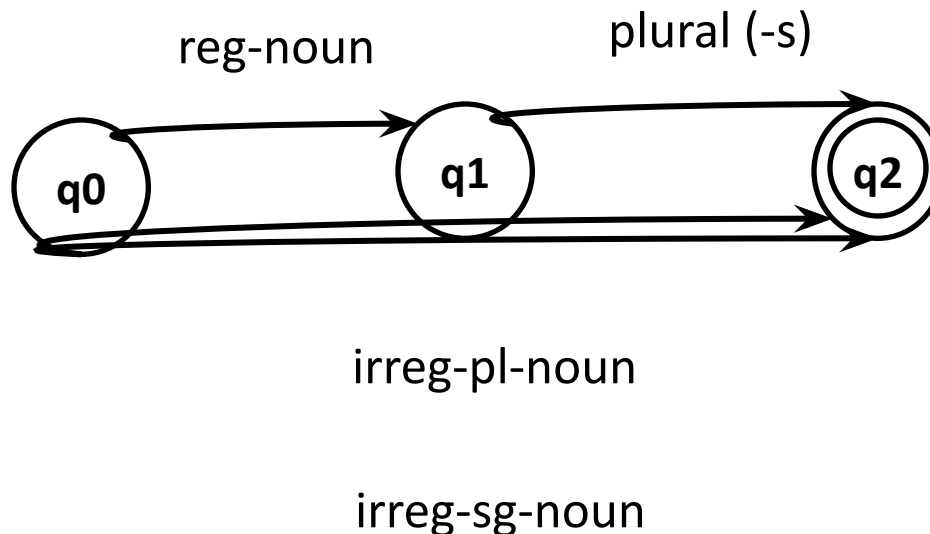


# Morphological Recognition: Nouns

## Lexicon

reg-noun	irregular-sg-noun	irregular-pl-noun	plural
flower	goose	geese	-s
cat	sheep	sheep	
dog	mouse	mice	

## FSA



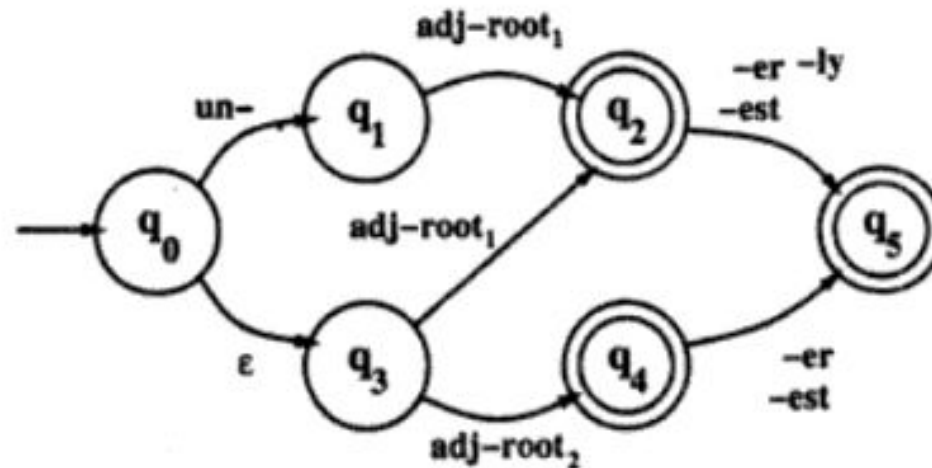
Note: Here, we are ignoring the nouns which take the suffix -es for pluralization

# Adjectives

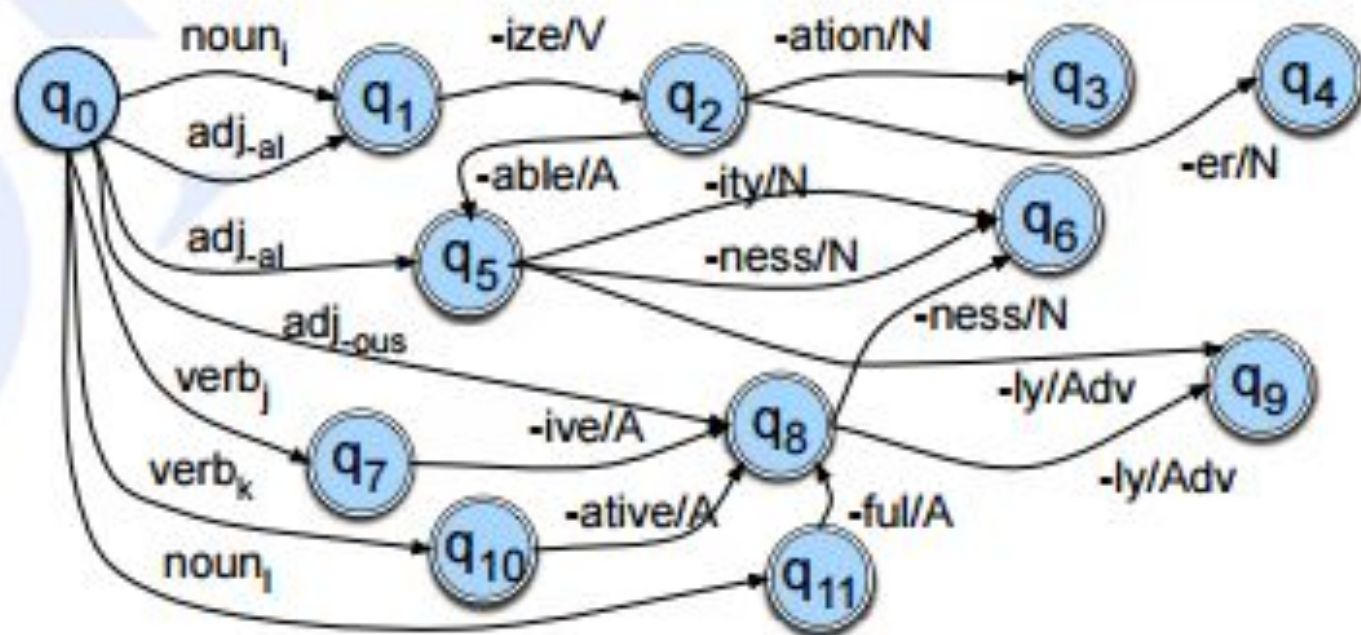
Type	Properties	Examples
adj-root1	Occur with un- and -ly	happy, real
Adj-root2	Can't occur with un- and -ly	big, red

# Adjectives

Type	Properties	Examples
adj-root1	Occur with un- and -ly	happy, real
Adj-root2	Can't occur with un- and -ly	big, red



**Figure 3.5** An FSA for a fragment of English adjective morphology: Antworth's Proposal #2.



**Figure 3.6** An FSA for another fragment of English derivational morphology.

# Recognition Vs Parsing

- Morphological recognition
  - `is_pasttense_verb(loved)` --> TRUE
  - Finite State Automata can do this
- Morphological parsing: what is its breakdown?
  - `parse(loved)` --> *take/VERB -n/PAST-TENSE*
  - Finite State Transducers can do this

# Finite State Transducers

- FSAutomata
  - An FSA represents a set of strings. e.g.  
*{walk, walks, walked, love loves, loved}*
    - *Regular language.*
  - A recognizer function.  
`recognize(str) -> true or false`
- FSTransducers
  - An FST represents a set of *pairs of strings* (think of as input,output pairs)  
*{ (walk, walk+V+PL), (walk, walk+N+SG), (walked, walk+V+PAST) ... }*
    - *Regular relation.* (Not a function!)
  - A transducer function: maps input to zero or more outputs.  
`transduce(walk) --> {walk+V+PL, walk+N+SG}`  
Can return multiple answers if ambiguity: e.g. if you don't have POS-tagged input, "walk" could be the verb "They walk to the store" versus the noun "I took a walk".
  - Generic *inversion* and *composition* operations.



# Finite State Transducers

- FSAutomata have *input* labels.
- FSTransducers have *input:output* pairs on labels.

$Q$  a finite set of  $N$  states  $q_0, q_1, \dots, q_{N-1}$

$\Sigma$  a finite set corresponding to the input alphabet

$q_0 \in Q$  the start state

$F \subseteq Q$  the set of final states

$\delta(q, w)$  the transition function or transition matrix between states; Given a state  $q \in Q$  and a string  $w \in \Sigma^*$ ,  $\delta(q, w)$  returns a set of new states  $Q' \subseteq Q$ .  $\delta$  is thus a function from  $Q \times \Sigma^*$  to  $2^Q$  (because there are  $2^Q$  possible subsets of  $Q$ ).  $\delta$  returns a set of states rather than a single state because a given input may be ambiguous in which state it maps to.

- FS Automata have *input* labels.
- FS Transducers have *input:output* pairs on labels.

New →

$Q$  a finite set of  $N$  states  $q_0, q_1, \dots, q_{N-1}$   
 $\Sigma$  a finite set corresponding to the input alphabet  
 $\Delta$  a finite set corresponding to the output alphabet  
 $q_0 \in Q$  the start state  
 $F \subseteq Q$  the set of final states  
 $\delta(q, w)$  the transition function or transition matrix between states; Given a state  $q \in Q$  and a string  $w \in \Sigma^*$ ,  $\delta(q, w)$  returns a set of new states  $Q' \in Q$ .  $\delta$  is thus a function from  $Q \times \Sigma^*$  to  $2^Q$  (because there are  $2^Q$  possible subsets of  $Q$ ).  $\delta$  returns a set of states rather than a single state because a given input may be ambiguous in which state it maps to.

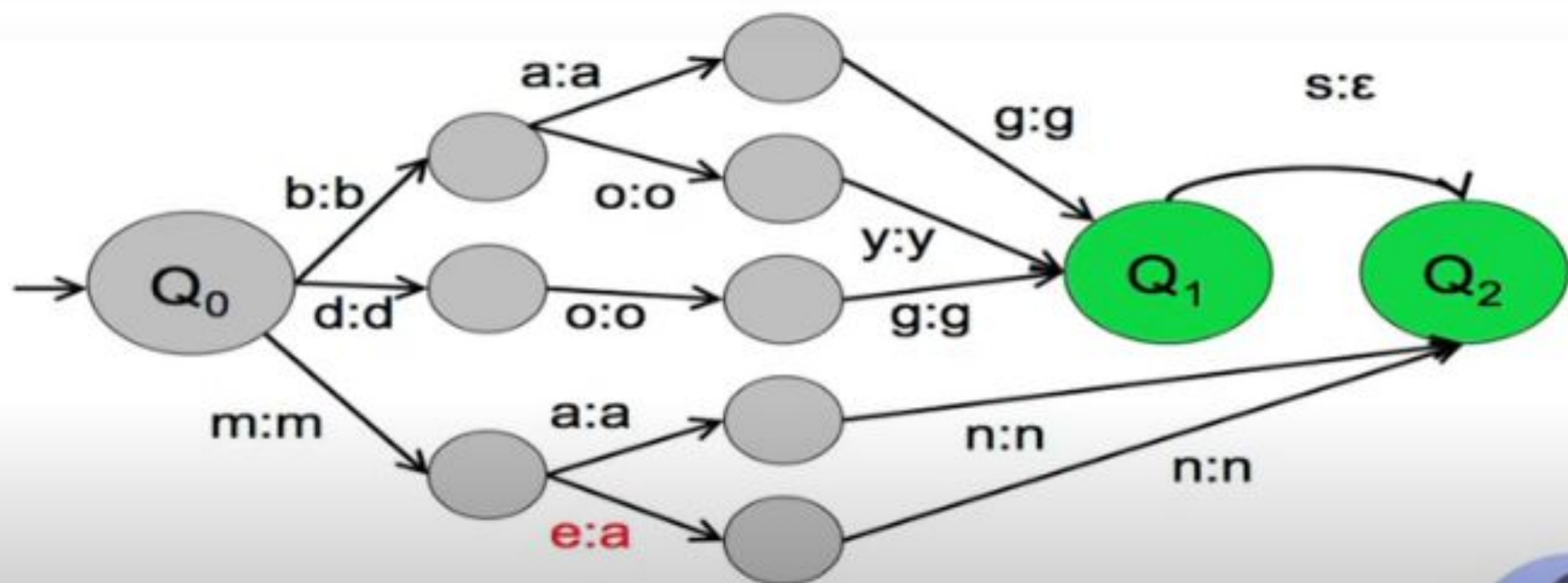
New →

$\sigma(q, w)$  the output function giving the set of possible output strings for each state and input. Given a state  $q \in Q$  and a string  $w \in \Sigma^*$ ,  $\sigma(q, w)$  gives a set of output strings, each a string  $o \in \Delta^*$ .  $\sigma$  is thus a function from  $Q \times \Sigma^*$  to  $2^{\Delta^*}$

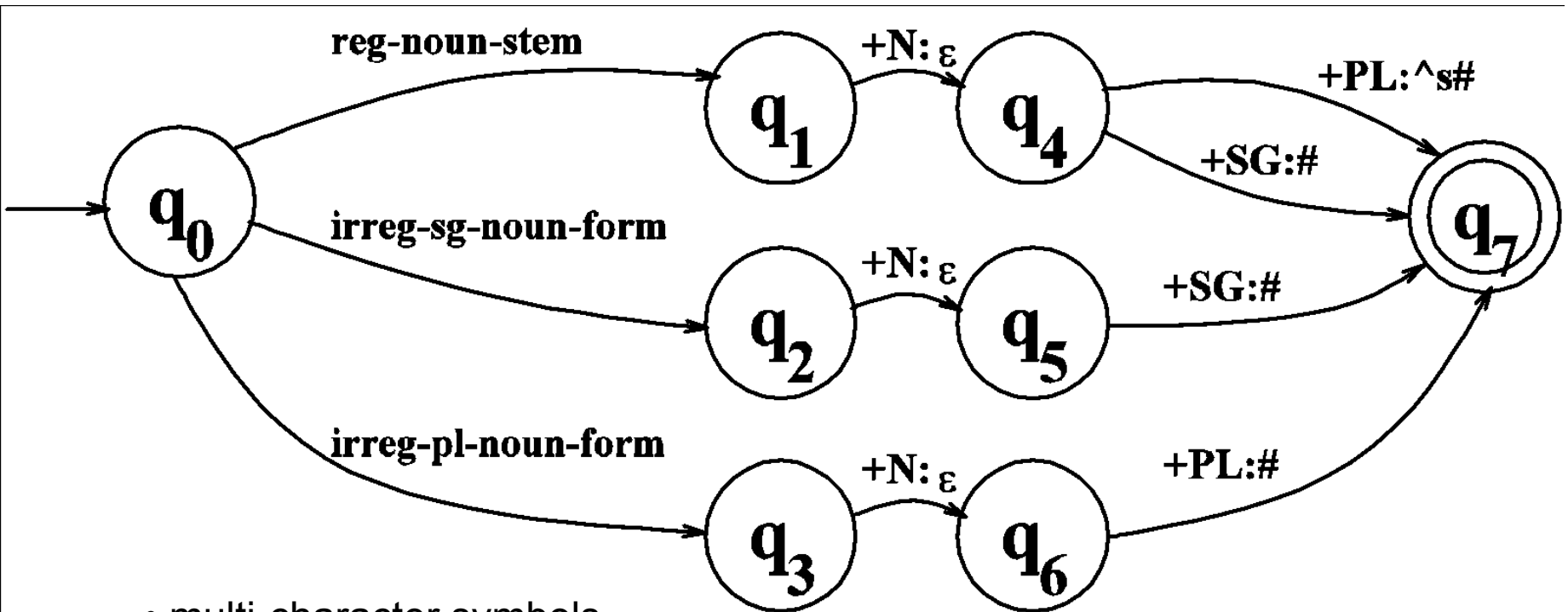


# By Dan Jurafsky book

- **FST as recognizer:** a transducer that takes a pair of strings as input and outputs *accept* if the string-pair is in the string-pair language, and *reject* if it is not.
- **FST as generator:** a machine that outputs pairs of strings of the language. Thus the output is a yes or no, and a pair of output strings.
- **FST as translator:** a machine that reads a string and outputs another string
- **FST as set relater:** a machine that computes relations between sets.

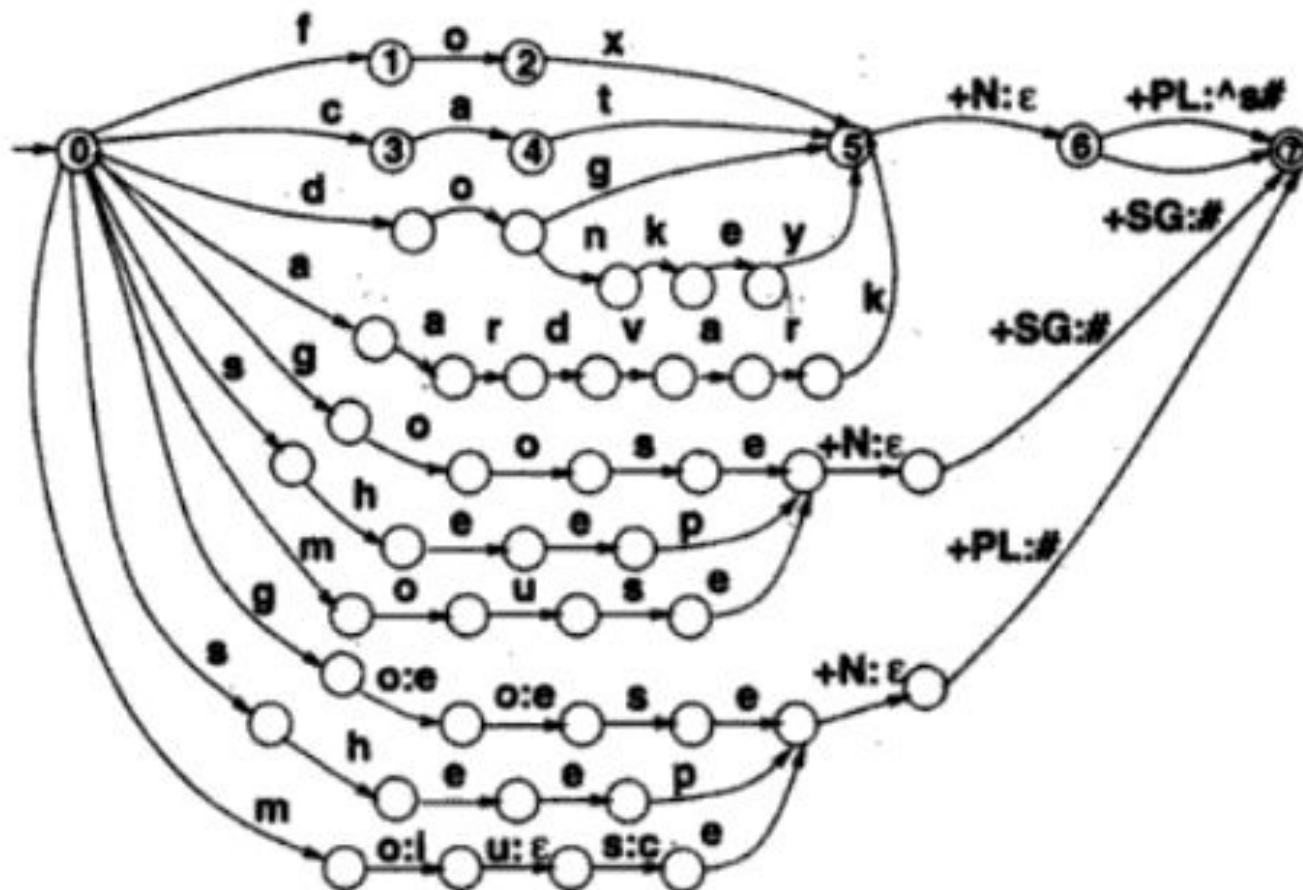


# 1. $T_{\text{num}}$ : Noun Number Inflection



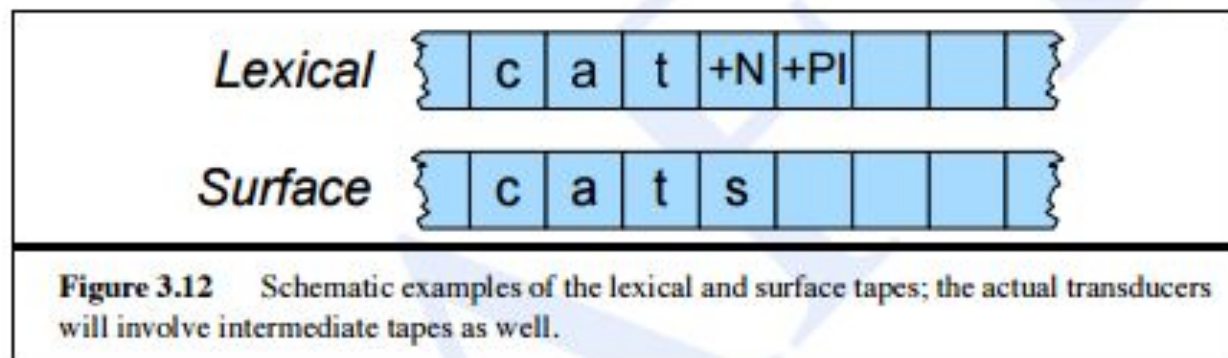
- multi-character symbols
- morpheme boundary ^
- word boundary #

<https://books.google.co.in/books?id=LCnx6xaDZsIC&pg=PA91&lpg=PA91&dq=orthographic+rules+in+morphological+parsing+for+sheep&source=bl&ots=h6RI9xSFNx&sig=ACfU3U38lgTS2S2ai6Z5By4Asfd3yXFFGg&hl=en&sa=X&ved=2ahUKewjg0Lyb8ZPnAhURWCsKHcrsCLU4ChDoATABegQlChAB#v=onepage&q=orthographic%20rules>



**Figure 3.11** A fleshed-out English nominal inflection FST  $T_{lex} = T_{num} \circ T_{stems}$ .

# FSTs for morph parsing



# Computational Morphology: Problems/Challenges

1. **Ambiguity:** one word can correspond to multiple structures (more critical in morphologically richer languages)
2. **Spelling changes:** may occur when two morphemes are combined  
e.g. butterfly + -s -> butterfl**ies**

# 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!

# Ambiguity: more complex example

- What's the right parse for **Unionizable**?
  - Union-ize-able
  - Un-ion-ize-able
- Each would represent a valid path through an FST for derivational morphology.
- Both Adj.....

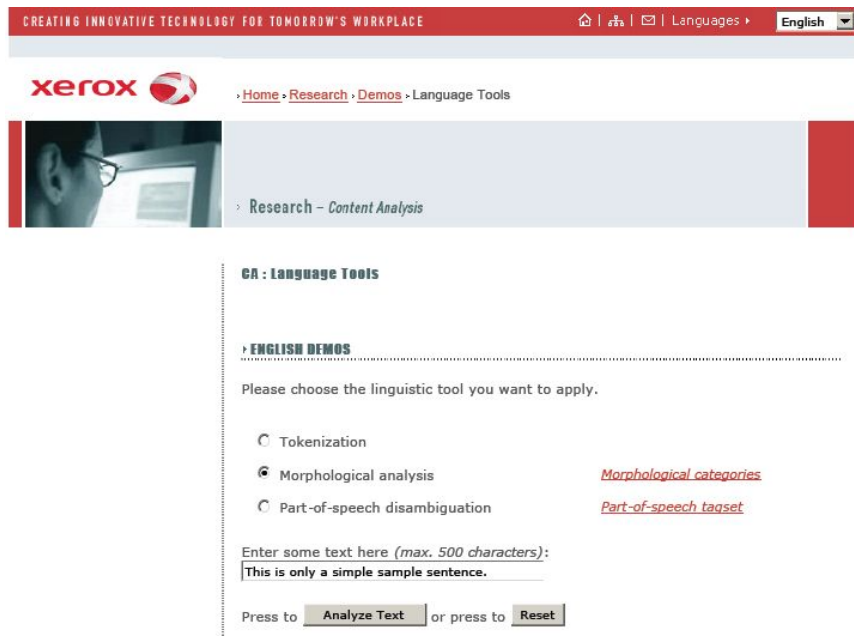
\_\_\_\_\_



# Deal with Morphological Ambiguity

- Find all the possible outputs (all paths) and return them all (without choosing)

*Then  
Part-of-speech  
tagging  
to choose..... look  
at the neighboring  
words*



CREATING INNOVATIVE TECHNOLOGY FOR TOMORROW'S WORKPLACE

Home | Research | Demos | Language Tools

English

xerox

Home » Research » Demos » Language Tools

Research - Content Analysis

CA : Language Tools

ENGLISH DEMOS

Please choose the linguistic tool you want to apply.

☐ Tokenization

☒ Morphological analysis [Morphological categories](#)

☐ Part-of-speech disambiguation [Part-of-speech tagset](#)

Enter some text here (max. 500 characters):

This is only a simple sample sentence.

Press to **Analyze Text** or press to **Reset**

## (2) Spelling Changes

When morphemes are combined inflectionally the spelling at the boundaries may change

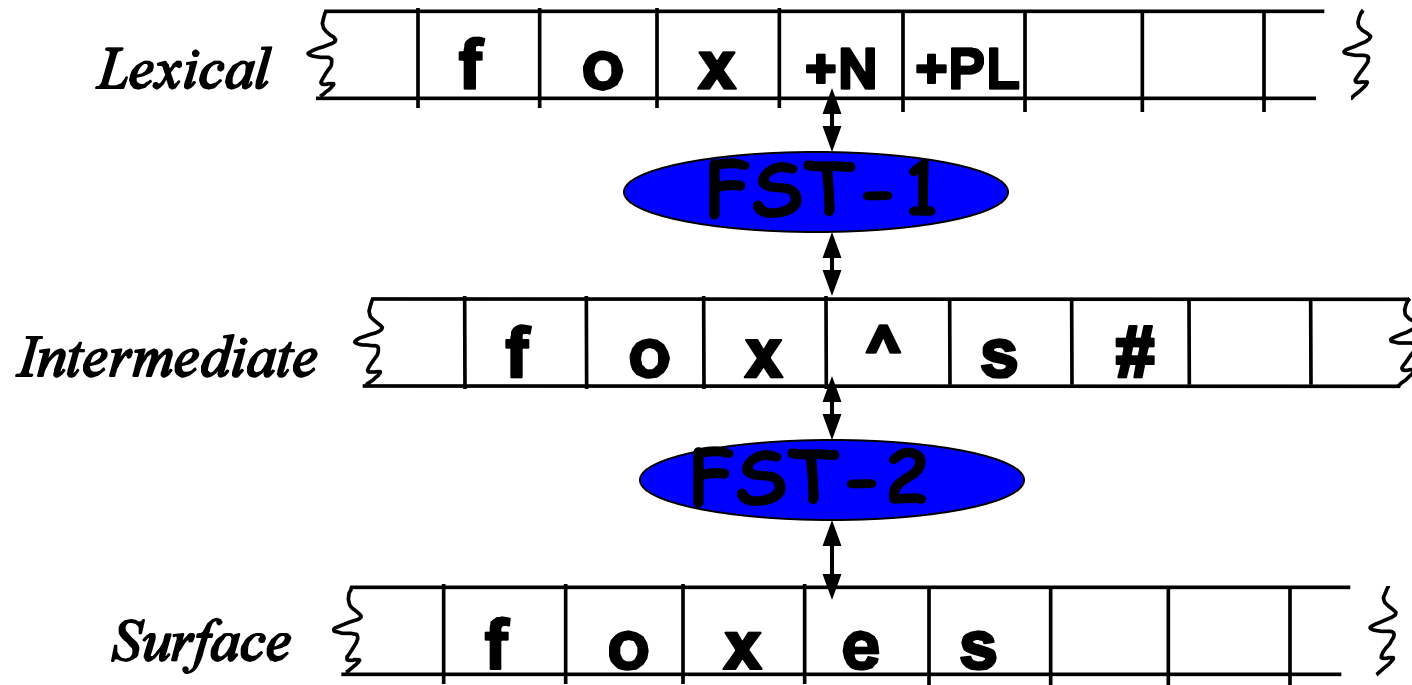
### Examples

- **E-insertion:** when **-s** is added to a word, **-e** is inserted if word ends in **-s**, **-z**, **-sh**, **-ch**, **-x** (e.g., *kiss*, *miss*, *waltz*, *bush*, *watch*, *rich*, *box*)
- **Y-replacement:** when **-s** or **-ed** are added to a word ending with a **-y**, **-y** changes to **-ie** or **-i** respectively (e.g., *butterfly*, *try*)

# Solution: Multi-Tape Machines

- Add intermediate tape
- Use the output of one tape machine as the input to the next
- Add intermediate symbols
  - ^ morpheme boundary
  - # word boundary

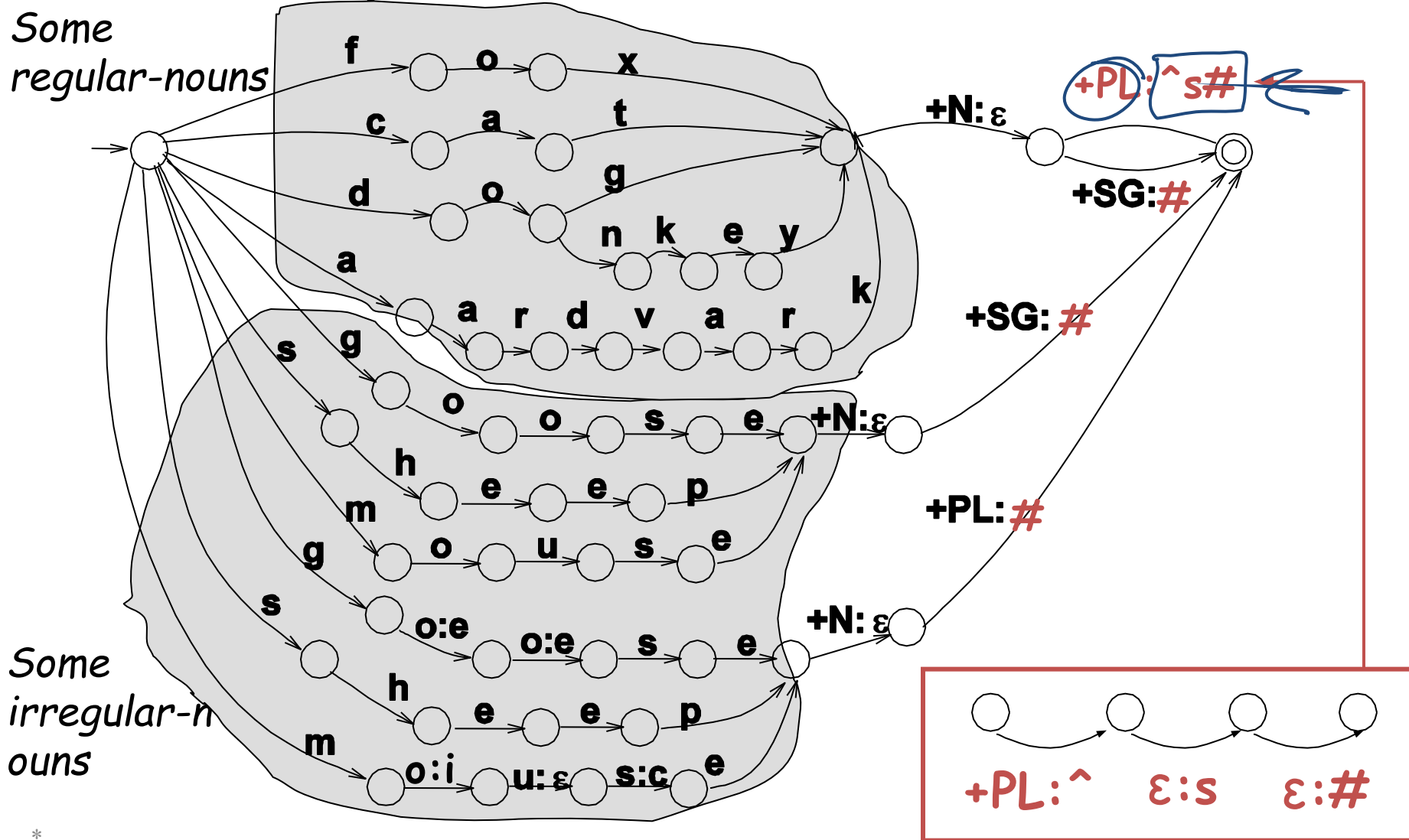
# Multi-Level Tape Machines



- FST-1 translates between the lexical and the intermediate level
- FTS-2 handles the spelling changes (due to one rule) to the surface tape

# FST-1 for inflectional morphology of plural

(Lexical  $\leftrightarrow$  Intermediate)



# Example

*lexical*

f	o	x	+N	+PL		
---	---	---	----	-----	--	--

*intemediate*

--	--	--	--	--	--	--

*lexical*

m	o	u	s	e	+N	+PL
---	---	---	---	---	----	-----

*intemediate*

--	--	--	--	--	--	--

# Intermediate Form to Surface

- The reason we need to have an intermediate form is that funny things happen at morpheme boundaries, e.g.

cat<sup>s</sup>  $\Leftrightarrow$  cats

fox<sup>s</sup>  $\Leftrightarrow$  foxes

fly<sup>s</sup>  $\Leftrightarrow$  flies

- The rules which describe these changes are called orthographic rules or "spelling rules".

# Orthographic rules and FST

- Spelling changes at morpheme boundaries by introducing rules
- Define additional FSTs to implement rules such as

consonant doubling/gemination (**beg** -> **begging**), edit→editing

```
define ConsonantDoubling g -> g g || C V _ "^" V;
```

'e' deletion (**make** -> **making**),

'e' ii

Name	Description of Rule	Example
Consonant doubling	1-letter consonant doubled before <i>-ing/-ed</i>	beg/begging
E deletion	Silent e dropped before <i>-ing</i> and <i>-ed</i>	make/making
E insertion	e added after <i>-s,-z,-x,-ch, -sh</i> before <i>-s</i>	watch/watches
Y replacement	-y changes to <i>-ie</i> before <i>-s</i> , <i>-i</i> before <i>-ed</i>	try/tries
K insertion	verbs ending <b>with</b> vowel + <i>-c</i> add <i>-k</i>	panic/panicked



# A Particular Spelling Rule

“insert an  $e$  on the surface tape just when the lexical tape has a morpheme ending in  $x$  (or  $z$ , etc) and the next morphemes is  $-s$ ”

$$\varepsilon \rightarrow e / \left\{ \begin{array}{c} x \\ s \\ z \end{array} \right\} \wedge \_ s \#$$

“rewrite  $a$  as  $b$  when it occurs between  $c$  and  $d$ ”

$$a \rightarrow b / c \_ d$$

This syntax is from the seminar paper of Chomsky and Halle (1968)

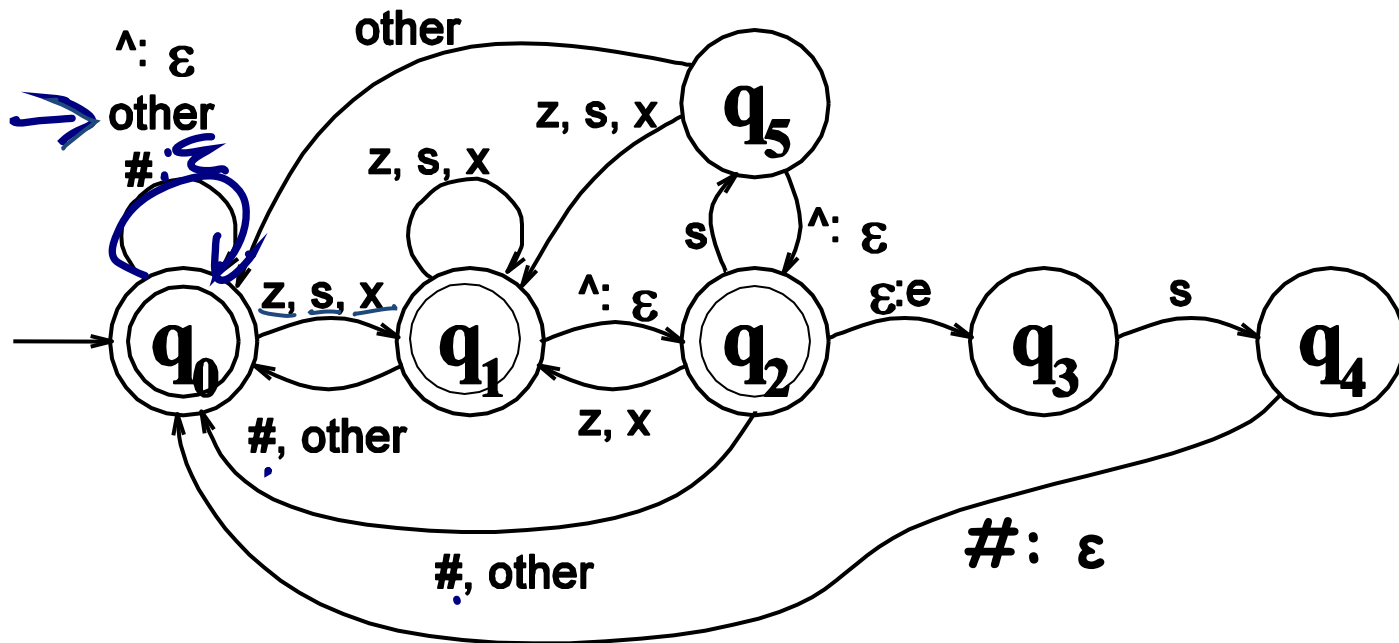
---

# FST-2 for E-insertion

(Intermediate  $\leftrightarrow$  Surface)

E-insertion: when  $-s$  is added to a word,  $-e$  is inserted if word ends in  $-s$ ,  $-z$ ,  $-sh$ ,  $-ch$ ,  $-x$

...as in  $fox^s\# \leftrightarrow foxes$



# Examples

*intermediate*

<b>f</b>	<b>o</b>	<b>x</b>	<b>^</b>	<b>s</b>	<b>#</b>	
----------	----------	----------	----------	----------	----------	--

*surface*

--	--	--	--	--	--	--

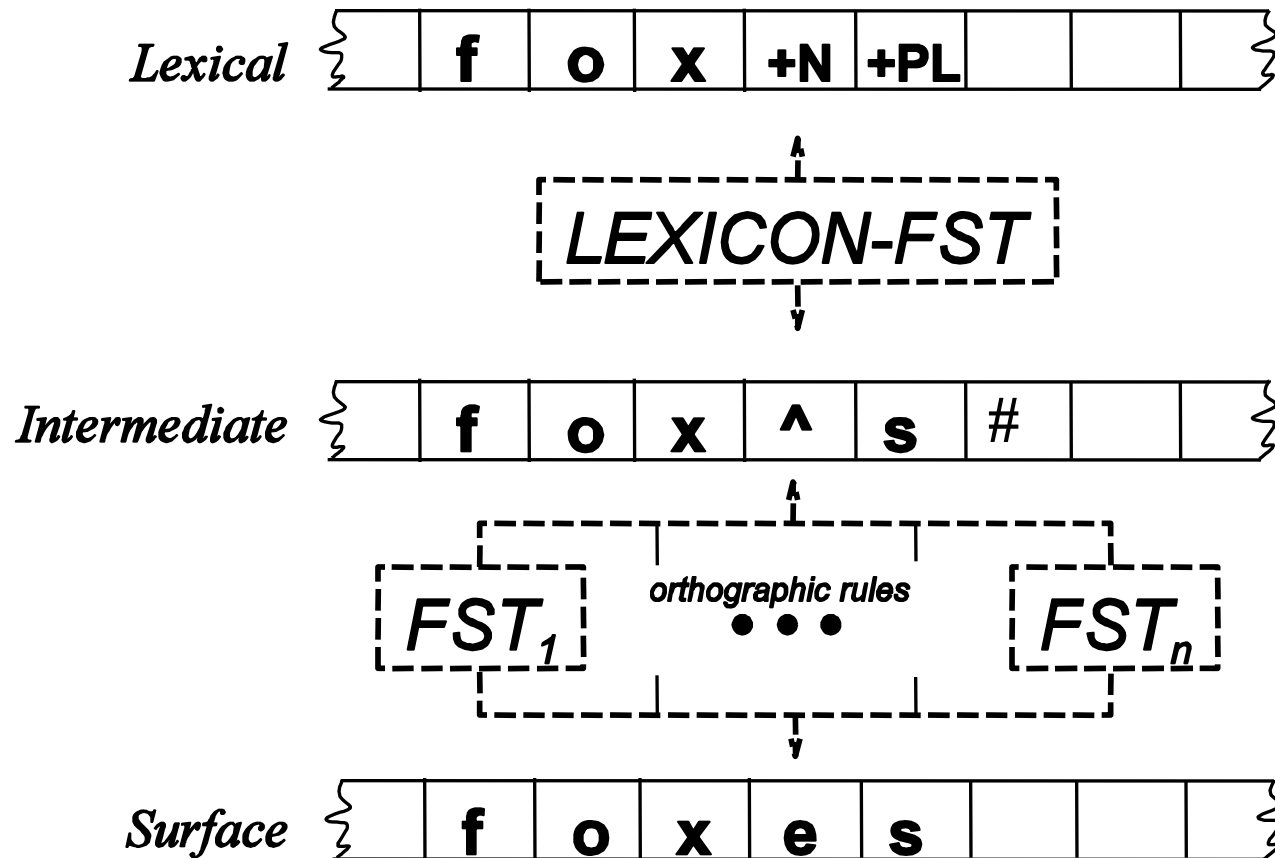
*intermediate*

<b>b</b>	<b>o</b>	<b>x</b>	<b>^</b>	<b>i</b>	<b>n</b>	<b>g</b>	<b>#</b>
----------	----------	----------	----------	----------	----------	----------	----------

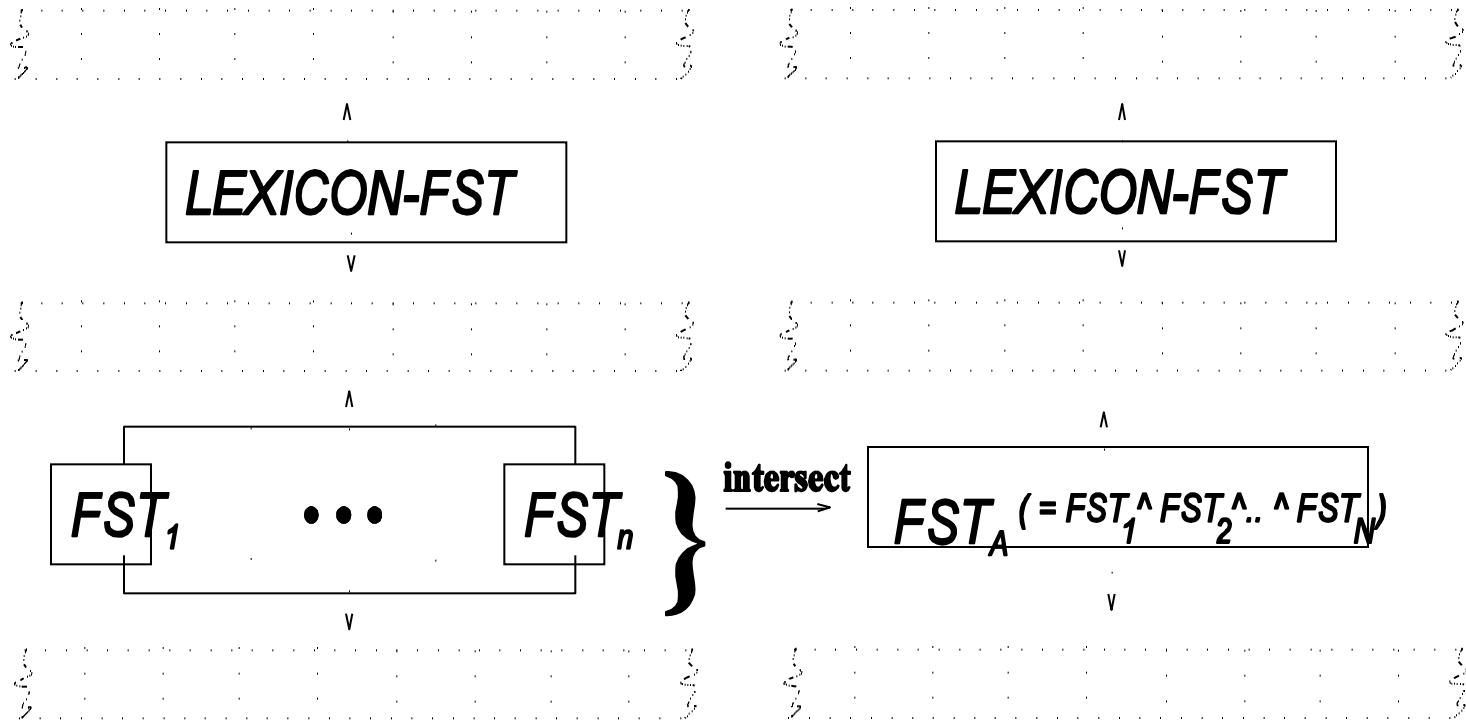
*surface*

--	--	--	--	--	--	--

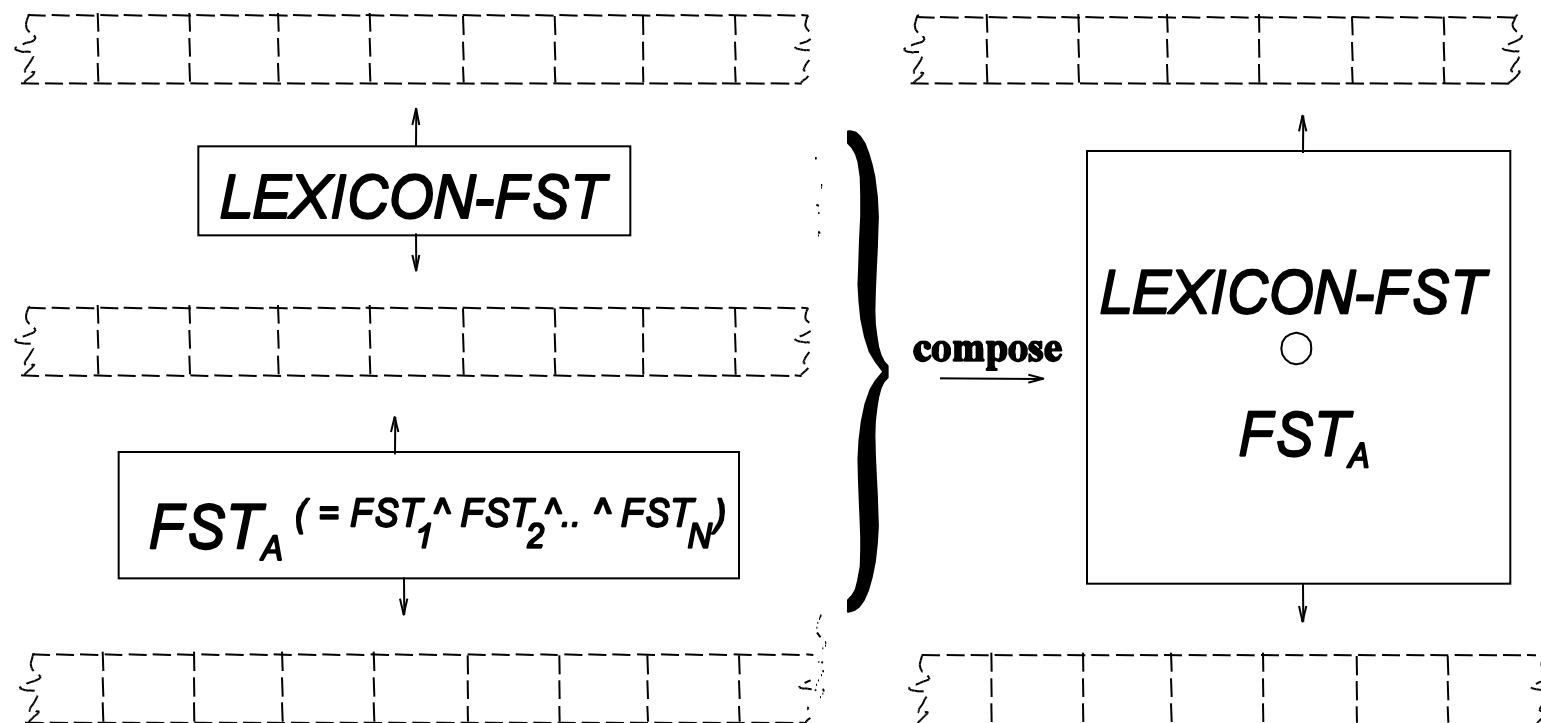
# Where are we?



# Final Scheme: Part 1



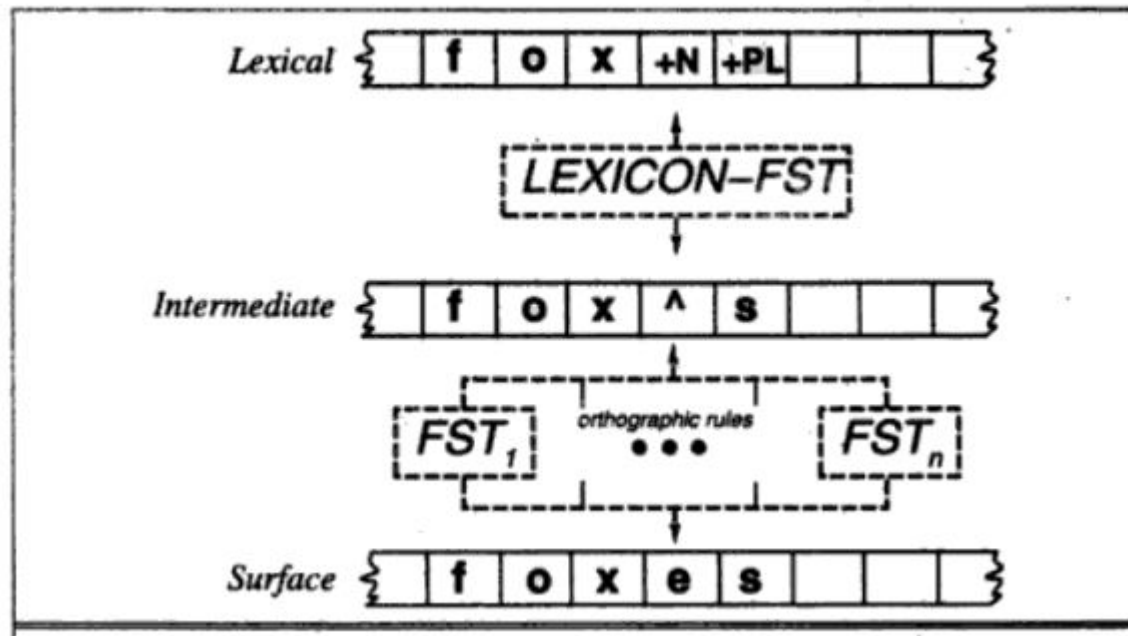
# Final Scheme: Part 2



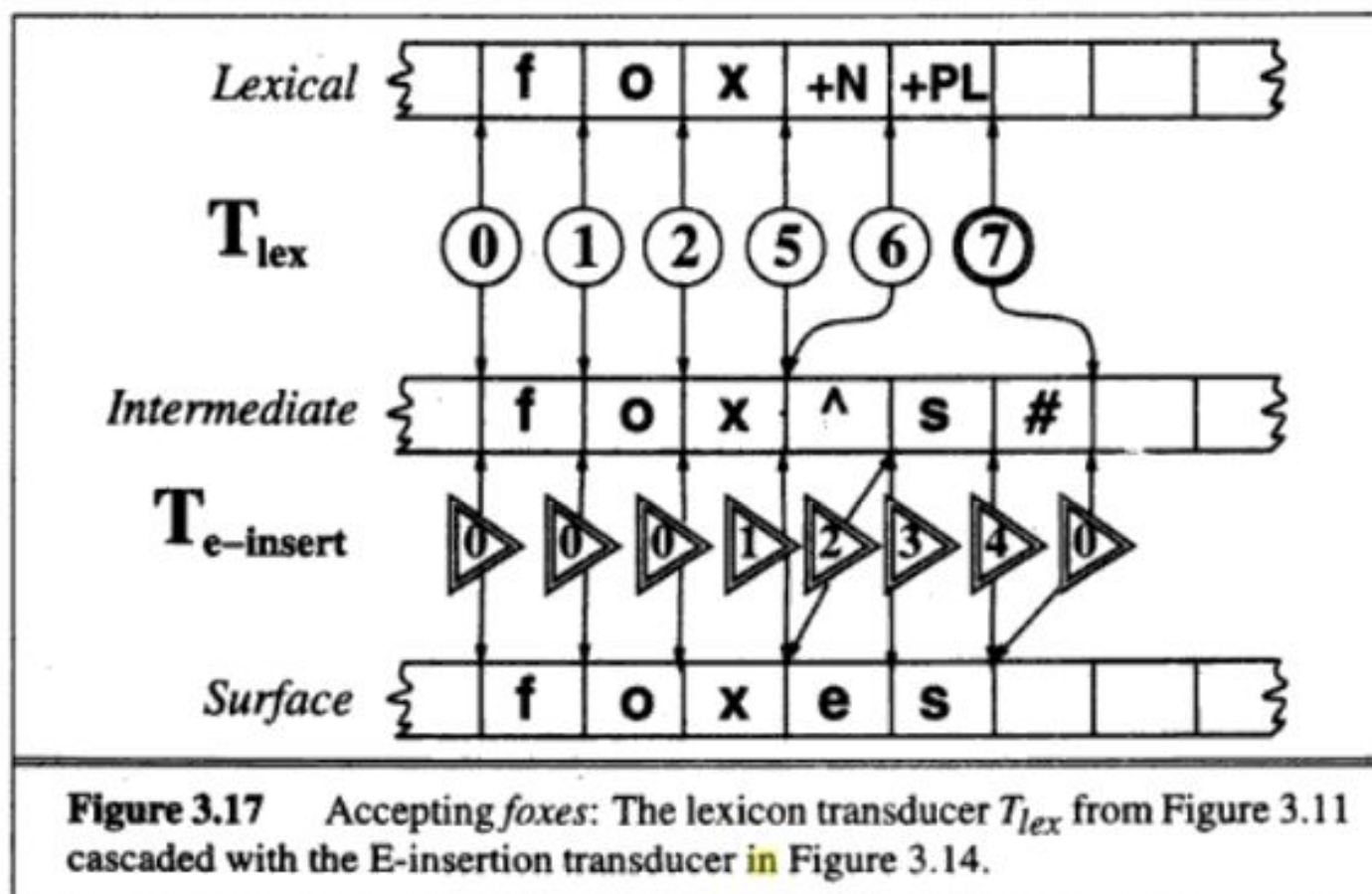
# Incorporating Spelling Rules

- Spelling rules, each corresponding to an FST, can be run *in parallel* provided that they are "aligned".
- The set of spelling rules is positioned between the surface level and the intermediate level.
- Parallel execution of FSTs can be carried out:
  - by simulation: in this case FSTs must first be aligned.
  - by first constructing a a single FST corresponding to their *intersection*.

# Generating or parsing with FST lexicon and rules





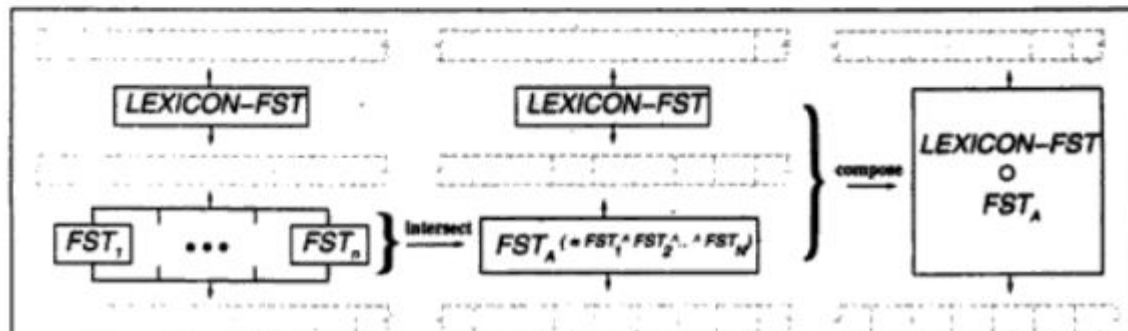


# Intersection and composition of transducer

## INTERSECTION

ries into a single more complex transducer. Transducers in parallel can be combined by **automaton intersection**. The automaton intersection algorithm just takes the Cartesian product of the states, i.e., for each state  $q_i$  in machine 1 and state  $q_j$  in machine 2, we create a new state  $q_{ij}$ . Then for any input symbol  $a$ , if machine 1 would transition to state  $q_n$  and machine 2 would transition to state  $q_m$ , we transition to state  $q_{nm}$ .

Figure 3.18 sketches how this intersection ( $\wedge$ ) and composition ( $\circ$ ) process might be carried out.



# Useful Operations on Transducers

- **Cascade**: running 2+ FSTs in sequence
- **Intersection**: represent the common transitions in FST1 and FST2 (ASR: finding pronunciations)
- **Composition**: apply FST2 transition function to result of FST1 transition function
- **Inversion**: exchanging the input and output alphabets (recognize and generate with same FST)
- cf AT&T FSM Toolkit cf AT&T FSM Toolkit and papers by Mohri cf AT&T FSM Toolkit and papers by Mohri, Pereira, and Riley

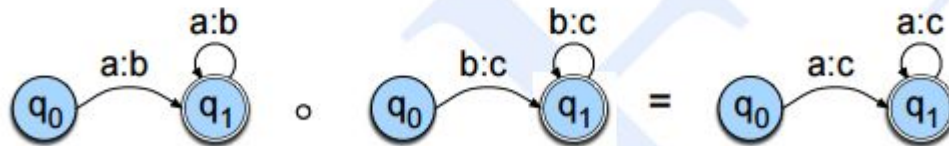
# Inversion

- **inversion:** The inversion of a transducer  $T$  ( $T^{-1}$ ) simply switches the input and output labels. Thus if  $T$  maps from the input alphabet  $I$  to the output alphabet  $O$ ,  $T^{-1}$  maps from  $O$  to  $I$ .

$$T = \{ (a, a1), (a, a2), (b, b1), (c, c1), (c, a) \}$$
$$T^{-1} = \{ (a1, a), (a2, a), (b1, b), (c1, c), (a, c) \}$$

# Composition

- **composition:** If  $T_1$  is a transducer from  $I_1$  to  $O_1$  and  $T_2$  a transducer from  $O_1$  to  $O_2$ , then  $T_1 \circ T_2$  maps from  $I_1$  to  $O_2$ .
- As transducer functions,  
 $(T_1 \circ T_2)(x) = T_1(T_2(x))$



**Figure 3.9** The composition of  $[a:b]^+$  with  $[b:c]^+$  to produce  $[a:c]^+$ .

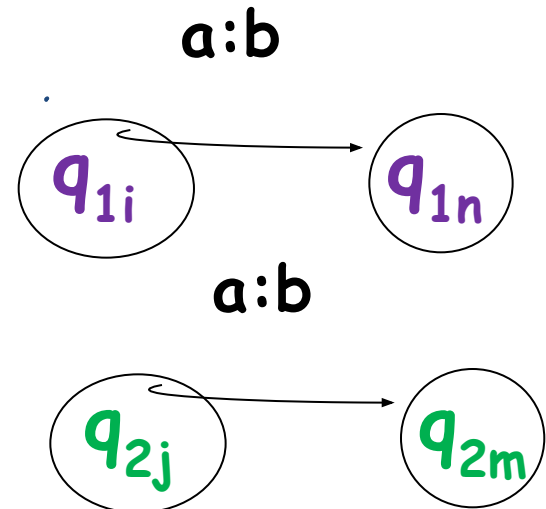
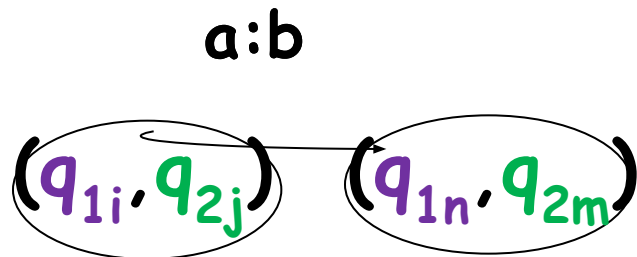
# Intersection (FST1, FST2) = FST3

- States of FST1 and FST2 :  $Q_1$  and  $Q_2$
- States of intersection:  $(Q_1 \times Q_2)$
- Transitions of FST1 and FST2 :  $\delta_1, \delta_2$
- Transitions of intersection :  $\delta_3$

For all  $i, j, n, m, a, b$   $\delta_3((q_{1i}, q_{2j}), a:b) = (q_{1n}, q_{2m})$  iff

-  $\delta_1(q_{1i}, a:b) = q_{1n}$  AND

-  $\delta_2(q_{2j}, a:b) = q_{2m}$



# Composition(FST1, FST2) = FST3

- States of FST1 and FST2 :  $Q_1$  and  $Q_2$
- States of composition :  $Q_1 \times Q_2$
- Transitions of FST1 and FST2 :  $\delta_1, \delta_2$
- Transitions of composition :  $\delta_3$

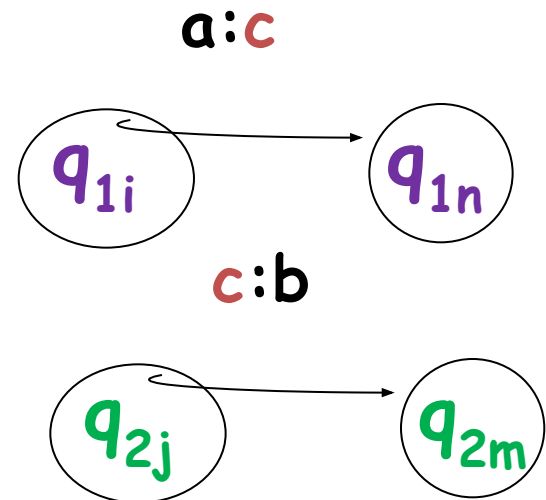
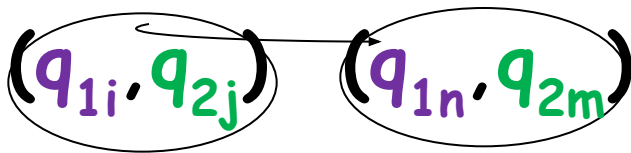
For all  $i, j, n, m, a, b$   $\delta_3((q_{1i}, q_{2j}), a:b) = (q_{1n}, q_{2m})$  iff

- There exists  $c$  such that

-  $\delta_1(q_{1i}, a:c) = q_{1n}$  AND

-  $\delta_2(q_{2j}, c:b) = q_{2m}$

$a:b$



# FSTs in Practice

- Install an FST package..... (pointers)
- Describe your “formal language” (e.g, lexicon, morphotactic and rules) in a RegExp-like notation (pointer)
- Your specification is compiled in a single FST

Ref: “Finite State Morphology” (Beesley and Karttunen, 2003, CSLI Publications)

## Complexity/Coverage:

- FSTs for the morphology of a natural language may have  $10^5$  –  $10^7$  states and arcs
- Spanish (1996)  $46 \times 10^3$  stems;  $3.4 \times 10^6$  word forms
- Arabic (2002?)  $131 \times 10^3$  stems;  $7.7 \times 10^6$  word forms



# Other important applications of FST in NLP

From **segmenting words into morphemes** to...

- **Tokenization:**

- finding word boundaries in text (?!) ...*maxmatch*
- Finding sentence boundaries: punctuation... but . is ambiguous look at example in
- **Shallow syntactic parsing:** e.g., find only noun phrases

- **Phonological Rules**

# A computer therapist?

**Computer:** TELL ME MORE ABOUT YOUR FAMILY

**Human:** My mother takes care of me.

**Computer:** WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU

**Human:** My father.

**Computer:** YOUR FATHER

**Human:** You are like my father in some ways.

**Computer:** WHAT RESEMBLANCE DO YOU SEE

**Human:** You are not very aggressive but I think you don't want me to notice that

**Computer:** WHAT MAKES YOU THINK I AM NOT VERY AGGRESSIVE

**Human:** You don't argue with me.

**Computer:** WHY DO YOU THINK I DON'T ARGUE WITH YOU

**Human:** You are afraid of me.

*Weizenbaum (1966), ELIZA.*

# ELIZA as a FST cascade

**Human:** *You don't argue with me.*

**Computer:** *WHY DO YOU THINK I DON'T ARGUE WITH YOU*

1. Replace **you** with **I** and **me** with **you**:

*I don't argue with you.*

2. Replace **<...>** with **Why do you think <...>**:

*Why do you think I don't argue with you.*

# What about compounds?

Compounds have hierarchical structure:

`((ice cream) cone) bakery)`

not `(ice ((cream cone) bakery))`

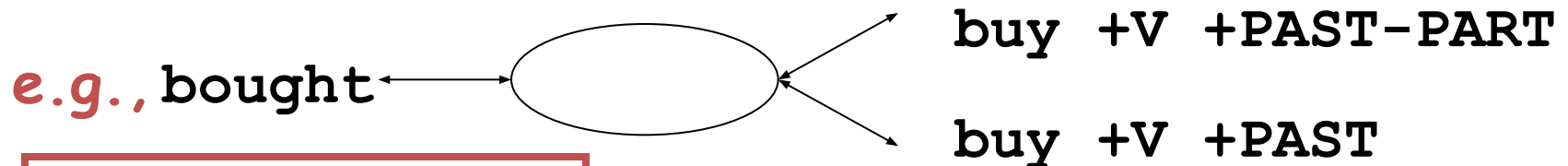
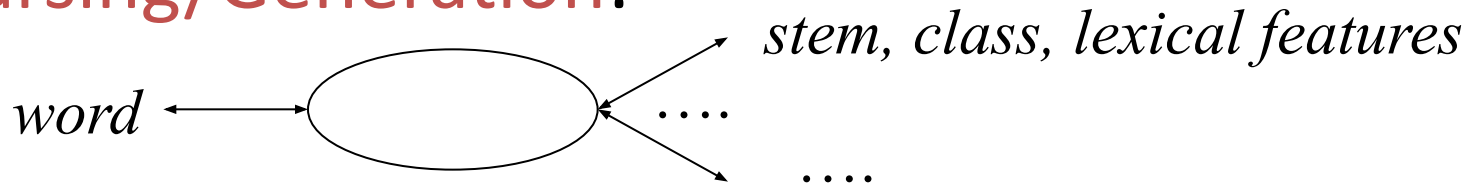
`((computer science) (graduate student))`

not `(computer ((science graduate) student))`

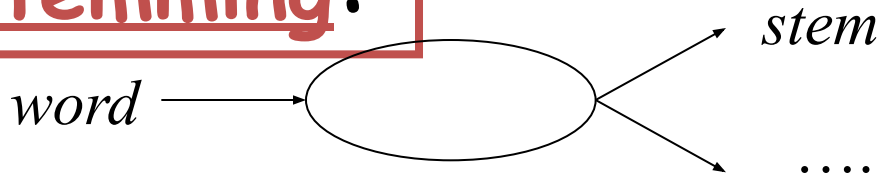
We will need context-free grammars to capture this underlying structure.

# Computational tasks in Morphology

- **Recognition**: recognize whether a string is an English word (FSA)
- **Parsing/Generation**:



- **Stemming**:



# Lexical and Functional morpheme

- Lexical, Free: Nouns, Verbs, Adj, Adv cat,  
town, call, house, hall, smart, fast
- Lexical, Bound: including derivational affixes  
rasp- [raspberry], cran- [cranberry] , -ceive  
[conceive, receive], un-, re-, pre-
- Functional, Free: Prepositions, Articles, Conj  
with, at, and, an, the, because
- Functional, Bound: inflectional affixes  
-s, -ed, -ing [eats, walked, laughing]

Identify the lexical and functional morphemes in the following words. Mention if they are free or bound.

1. politically
2. beautiful
3. between
4. writing
5. raspberries
6. unable
7. nationalization

# Look at the following words and break them up into smaller units

useless

copilot

psychology

sickness

communism

socialism

artist

dentist

curable

impossible

microstructure

macrostructure

departure



# answers

- ▣ useless – endless, meaningless – “without”
- ▣ copilot – co-author, co-editor – “with”
- ▣ psychology – physiology, biology – “study of”
- ▣ sickness – dizziness, happiness – “state of being”
- ▣ communism – socialism, feminism – “belief, doctrine”
- ▣ artist – activist, tourist – “one who”
- ▣ impossible – impatient, immoral – “not”
- ▣ microstructure – microscope, microbiology – “small”
- ▣ macrostructure – macroeconomics, macro - “large”

What about coalition, departure, important, dentist?

# FST Based Morphological Analyzer for Hindi Language Deepak Kumar , Manjeet Singh , and Seema Shukla

corpus collected from the LDC-IL  
(Linguistic Data Consortium)

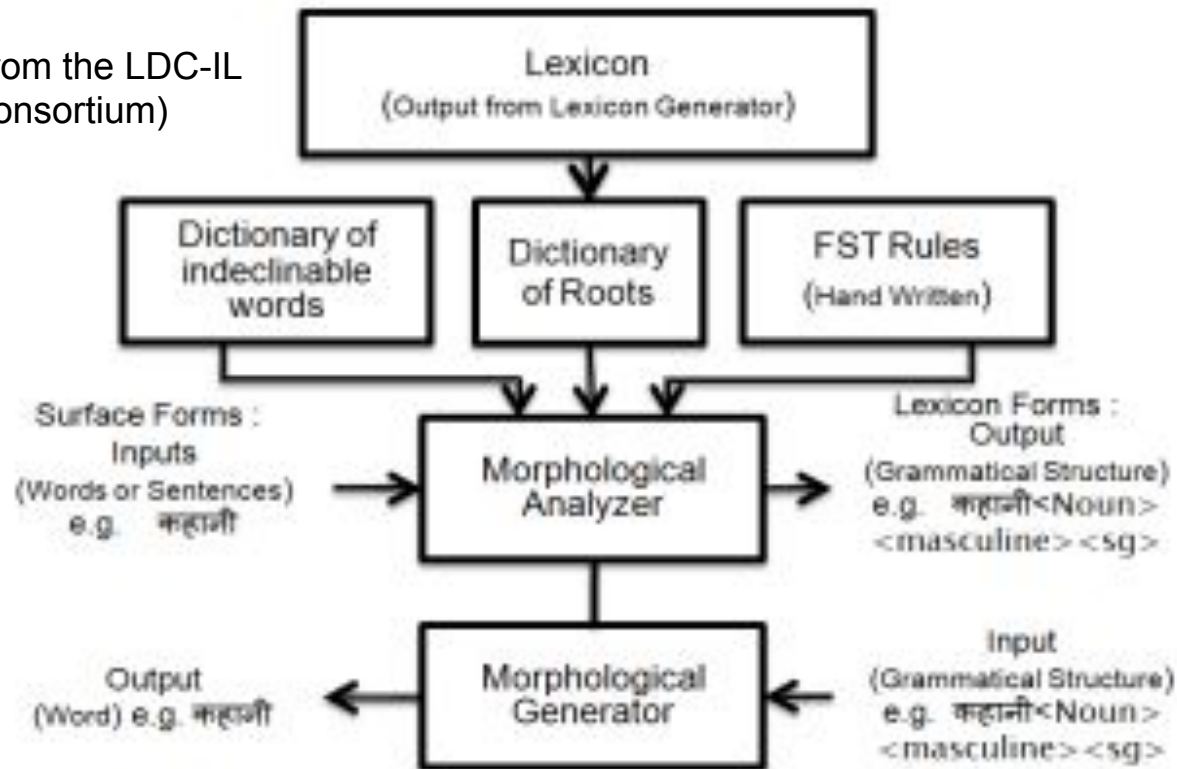


Fig. 2 Morphological Processor.

Table 1 Results for Noun Inflections

Input Words	Inflection Type	Output
लडका लडकी	Gender Inflection	लडका<Noun><masculine><sg> लडकी<Noun><feminine><sg>

Table 3 Results for Verb Inflection

Input Words	Inflection Type	Output
जा रहा	Person Inflection	जा<Verb><Indicative><Masculine><Progressive><sg>
जा रहे	Person Inflection	जा<Verb><Indicative><Masculine><Progressive><pl>

^ठेचा/ठेचा<n><m><sg><nom>\$  
 ^मिरची/मिरची<n><f><sg><nom>\$  
 ^/,<cm>\$  
 ^चिंच/चिंच<n><f><sg><nom>\$  
 ^व/व<cnjcoo>\$  
 ^मीठापासून/मीठ<n><nt><sg><obl>+पासून<post><adv>\$  
 ^तयार केला/तयार करणे<vblex><perf><p3><m><sg>\$  
 ^जातो/जाणे<vblex><impf><p3><m><sg>\$  
 ^/,<sent>\$

**Figure 1:** Example output from the analyser for the sentence ठेचा मिरची, चिंच व मीठापासून तयार केला जातो *thēcā miracī, cīmnca va mīṭhāpāsūna tayāra kelā jāto* “Pickles are prepared using chilis, tamarind and salt.” Note that the example has been manually disambiguated for brevity. The tag *cnjcoo* is coordinating conjunction, and *cm* is comma.

Finite-State Morphological Analysis  
for Marathi Vinit  
Ravishankar, Francis M. Tyers

## Morphological analysis and word Generation :

### Example in Hindi :

The forms of 'खेल'(khela) are the following:

खेल(khela), खेला(khelaa), खेली(khelii), खेलूंगा(kheluungaa), खेलूंगी(kheluungii), खेलेगा(khelegaa), खेलेगी(khelegii), खेलते(khelate), खेलती(khelatii), खेलने(khelane), खेलकर(khelakar)

### Example in English :

forms: 'play' are 'plays', 'played' and 'playing'.

Grace,disgrace, graceful,ungraceful, ungracefully, undisgraceful

disgracefully

dis    grace    ful    ly

Prefix   stem    suffix   suffix

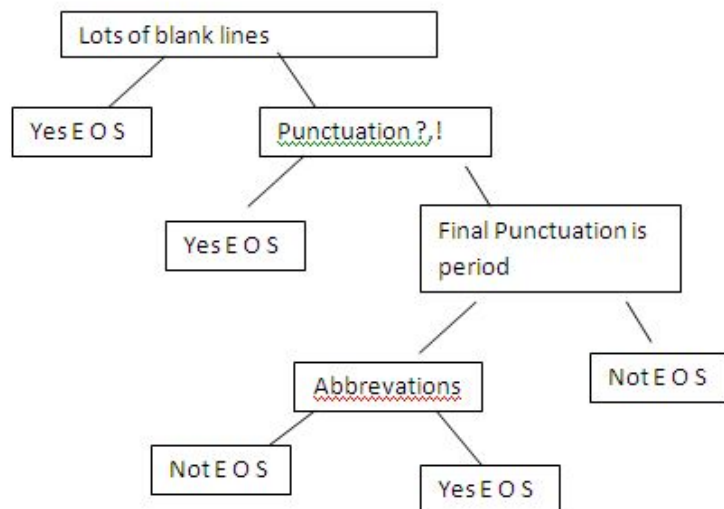
Neg    grace+N    +adj    +adv

# Text Normalization

- **Tokening (segmenting ) words:**the process of segmenting a string of characters into words.

```
cat WAR_AND_PEACE_By_LeoTolstoi.txt | tr -cs "[:alnum:]" "\n" | tr "[:lower:]"  
"[:upper:]" | awk '{h[$1]++}END{for (i in h){print h[i]" "i}}'|sort -nr | cat -n | head  
-n 30    ..most frequent used words
```

- sentence segmentation (the problem of deciding where the sentences begin or end)



# Word tokenization normalization and substitution

- Done by cascades of simple regular expressions substitutions or Finite automata.

Basic operations in RE include disjunction(`[]`,`|`,`.`), counters(`*`,`+`,`{n,m}`), anchors(`^`,`$`) and precedence operators(`(`,`)`)

```
from nltk.tokenize import RegexpTokenizer
```

```
tokenizer = RegexpTokenizer("[\w']+")  
text = "Let's see how it's working."  
tokenizer.tokenize(text)
```

- Case folding: convert to lower case  
eg: Bag and bag
- remove misspelling, punctuations, contradictions  
isn't->is not
- Substitution: doc1 : km/hr doc 2:m/s doc 3: mpsec



सत्यम्ब्रूयात्प्रियम्ब्रूयान्नब्रूयात्सत्यमप्रियम्प्रियञ्चनानृतम्ब्रूयादेषधर्मःसनातनः

*satyaṃbrūyātpriyaṃbrūyānnabrūyātsatyamapriyaṃpriyaṃcanāṇṛtambrūyād-  
eṣadharmahsanātanah.*

"One should tell the truth, one should say kind words; one should neither tell harsh truths, nor flattering lies; this is a rule for all times."

**Segmented Text:**

*satyam brūyāt priyam brūyāt na brūyāt satyam apriyam priyam ca na anṛtam  
brūyāt eṣaḥ dharmah sanātanah.*

# Text Segmentation : Sanskrit

## *General assumption behind the design*

Sentences from Classical Sanskrit may be generated by a regular relation  $R$  of the Kleene closure  $W^*$  of a regular set  $W$  of *words* over a finite alphabet  $\Sigma$ .

- $W$ : vocabulary of (inflected) words (*padas*) and
- $R$ : sandhi

## *Analysis of a sentence*

A candidate sentence  $w$  is analyzed by inverting relation  $R$  to produce a finite sequence  $w_1, w_2, \dots, w_n$  of word forms, together with a proof that  $w \in R(w_1 \cdot w_2 \dots \cdot w_n)$ .



✓Undo (120 Solutions)

satyambrūyātpriyambrūyānnabrūyātsatyam a priyamprīyañcanānṛtambrūyādeśadharmāsana ā tanah

# Lemmatization

lemmatization reduces the words to a word existing in the language.

```
import nltk
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize("Machine", pos='n'))
# pos: parts of speech tag, verb
print(lemmatizer.lemmatize("hopping", pos='v'))
```

Machine

hop

# Porter Stemmer

Stemming is the process of producing morphological variants of a root/base word.

# Stemming

- Reducing terms to their stems, used in information retrieval
- Crude chopping of affixes
  - language dependent
  - *automate(s), automatic, automation* all reduced to *automat*

CONNECT  
CONNECTED  
CONNECTING  
CONNECTION  
CONNECTIONS

Performance of an IR System would increase if term groups such as this are conflated into a single term.

*Suffix stripping reduce the total number of terms and reduce size and complexity of the data.*

# Porter Stemmer

- based on a series of sets of simple cascaded rewrite rules:
- (condition) S1->S2
  - ATIONAL → ATE (relational → relate)
  - (\*v\*) ING → ε if stem contains vowel (motoring → motor)

Cascade of rules applied to: **computerization**

- ization -> -ize **computerize**
- ize -> ε **computer**
- Errors occur:
  - organization → organ, university → universe

There are mainly two errors in stemming – **Overstemming** and **Understemming**. Overstemming occurs when two words are stemmed to same root that are of different stems. Under-stemming occurs when two words are stemmed to same root that are not of different stems

This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all things-names and heights and soundings-with the single exception of the red crosses and the written notes.

produces the following stemmed output:

Thi wa not the map we found in Billi Bone s chest but an accur copi complet in all thing name and height and sound with the singl except of the red cross and the written note

## THE ALGORITHM

A *consonant* in a word is a letter other than A, E, I, O or U, and other than Y preceded by a consonant. (The fact that the term ‘consonant’ is defined to some extent in terms of itself does not make it ambiguous.) So in TOY the consonants are T and Y, and in SYZYGY they are S, Z and G. If a letter is not a consonant it is a *vowel*.

A consonant will be denoted by c, a vowel by v. A list ccc... of length greater than 0 will be denoted by C, and a list vvv... of length greater than 0 will be denoted by V. Any word, or part of a word, therefore has one of the four forms:

CVCV ... C  
CVCV ... V  
VCVC ... C  
VCVC ... V

These may all be represented by the single form

[C]VCVC ... [V]

where the square brackets denote arbitrary presence of their contents. Using  $(VC)^m$  to denote VC repeated m times, this may again be written as

$[C](VC)^m[V]$ .

m will be called the *measure* of any word or word part when represented in this form. The case  $m = 0$  covers the null word. Here are some examples:

$m=0$  TR, EE, TREE, Y, BY.

$m=1$  TROUBLE, OATS, TREES, IVY.

$m=2$  TROUBLES, PRIVATE, OATEN, ORRERY.

The *rules* for removing a suffix will be given in the form

(condition) S1 -> S2

This means that if a word ends with the suffix S1, and the stem before S1 satisfies the given condition, S1 is replaced by S2. The condition is usually given in terms of *m*, e.g.

( $m > 1$ ) EMENT ->

Here S1 is 'EMENT' and S2 is null. This would map REPLACEMENT to REPLAC, since REPLAC is a word part for which  $m = 2$ .

The 'condition' part may also contain the following:

- \*S - the stem ends with S (and similarly for the other letters).
- \*v\* - the stem contains a vowel.
- \*d - the stem ends with a double consonant (e.g. -TT, -SS).
- \*o - the stem ends cvc, where the second c is not W, X or Y (e.g. -WIL, -HOP).

And the condition part may also contain expressions with *and*, *or* and *not*, so that

( $m > 1$  and (\*S or \*T))

tests for a stem with  $m > 1$  ending in S or T, while

(\*d and not (\*L or \*S or \*Z))

tests for a stem ending with a double consonant other than L, S or Z. Elaborate conditions like this are required only rarely.



# Porter's algorithm

What is the purpose of including an identity rule such as  $SS \rightarrow SS$ ?

## Step 1a

- $sses \rightarrow ss$  (caresses  $\rightarrow$  caress)
- $ies \rightarrow i$  (ponies  $\rightarrow$  poni)
- $ss \rightarrow ss$  (caress  $\rightarrow$  caress)
- $s \rightarrow \phi$  (cats  $\rightarrow$  cat)

## Step 1b

- $(*v*)ing \rightarrow \phi$  (walking  $\rightarrow$  walk, king  $\rightarrow$  king)
- $(*v*)ed \rightarrow \phi$  (played  $\rightarrow$  play)

If the second or third of the rules in Step 1b is successful, the following is done:

AT	-> ATE	conflat(ed) -> conflate
BL	-> BLE	troubl(ed) -> trouble
IZ	-> IZE	siz(ed) -> size
(*d and not (*L or *S or *Z))	-> single letter	hopp(ing) -> hop
		tann(ed) -> tan
		fall(ing) -> fall
		hiss(ing) -> hiss
		fizz(ed) -> fizz
(m=1 and *o)	-> E	fail(ing) -> fail
		fil(ing) -> file



#### Step 1c

(\*v\*) Y -> I    happy -> happi  
                     sky    -> sky

Step 1 deals with plurals and past participles. The subsequent steps are much more straightforward.

## Porter's algorithm

#### Step 2

- ational → ate (relational → relate)
- izer → ize (digitizer → digitize)
- ator → ate (operator → operate)

#### Step 3

- al →  $\phi$  (revival → reviv)
- able →  $\phi$  (adjustable → adjust)
- ate →  $\phi$  (activate → activ)

(m>0) ICATE -> IC	triplicate -> triplic
(m>0) ATIVE ->	formative -> form
(m>0) ALIZE -> AL	formalize -> formal
(m>0) ICITI -> IC	electricity -> electric
(m>0) ICAL -> IC	electrical -> electric
(m>0) FUL ->	hopeful -> hope
(m>0) NESS ->	goodness -> good

# Step 4

## Step 4

(m>1) AL	->	revival	-> reviv
(m>1) ANCE	->	allowance	-> allow
(m>1) ENCE	->	inference	-> infer
(m>1) ER	->	airliner	-> airlin
(m>1) IC	->	gyroscopic	-> gyroscop
(m>1) ABLE	->	adjustable	-> adjust
(m>1) IBLE	->	defensible	-> defens
(m>1) ANT	->	irritant	-> irrit
(m>1) EMENT	->	replacement	-> replac
(m>1) MENT	->	adjustment	-> adjust
(m>1) ENT	->	dependent	-> depend
(m>1 and (*S or *T)) ION	->	adoption	-> adopt
(m>1) OU	->	homologou	-> homolog
(m>1) ISM	->	communism	-> commun
(m>1) ATE	->	activate	-> activ
(m>1) ITI	->	angulariti	-> angular
(m>1) OUS	->	homologous	-> homolog
(m>1) IVE	->	effective	-> effect
(m>1) IZE	->	bowdlerize	-> bowdler

The suffixes are now removed. All that remains is a little tidying up.

## Step 5

### Step 5a

(m>1) E -> probate -> probat  
rate -> rate  
(m=1 and not \*o) E -> cease -> ceas

Step 5b

(m > 1 and \*d and \*L) -> single letter    controll -> control  
roll        -> roll

Complex suffixes are removed as follows:-

## Surface word: Generalizations

## Step 1: Generalization

## Step2: Generalize

### Step3: General

## Step4: Gener

```
import nltk
from nltk.stem import PorterStemmer
ps = PorterStemmer()
```

```
sentence = "machine eat eating ate giving learning "
```

```
for word in sentence.split():
    print(ps.stem(word))
```

```
machin
eat
eat
ate
give
learn
```

# Online Porter Stemmer

## Stemming and Lemmatization with Python NLTK

This is a demonstration of **stemming** and **lemmatization** for the 17 languages supported by the **NLTK 2.0.4 stem** package.

Stem Text

Choose stemmer

Porter ▼

Enter text

doing analysis analyze  
analys  
asses  
  
connected  
connections  
connecting  
  
organize policy sparsity

Enter up to 50000 characters

Stem

Stemmed Text

do analysi analyz anali ass connect connect cconnect organ  
polici sparsiti

What rule should be added to correctly stem *pony*?  
The stemming for *ponies* and *pony* might seem strange. Does it have a deleterious effect on retrieval? Why or why not?

<https://text-processing.com/demo/stem/>

# Difference between Porter and Lancaster Stemmer

Word	Porter Stemmer	Lancaster Stemmer
friend	friend	friend
friendship	friendship	friend
friends	friend	friend
friendships	friendship	friend
stabil	stabil	stabl
destabilize	destabil	dest
misunderstanding	misunderstand	misunderstand
railroad	railroad	railroad
moonlight	moonlight	moonlight
football	footbal	footbal

# Lancaster stemmer..aggressive algo can be customized

**Stem Text**

Choose stemmer

Lancaster ▼

Enter text

abandonment  
university  
absorbency  
absorbent  
universe  
boss  
caress  
pony  
ponies  
cement  
optimum

Enter up to 50000 characters

Stem

**Stemmed Text**

abandon univers absorb absorb univers boss caress pony  
pony cem optim

# Other algorithms



# Stemming and Lemmatization

- both generate the root form of the inflected words. The difference is that stem might not be an actual word whereas, lemma is an actual language word.
- Stemming follows an algorithm with steps to perform on the words which makes it faster.
- Whereas, in lemmatization, you used WordNet corpus and a corpus for stop words as well to produce lemma which makes it slower than stemming.
- You also had to define a parts-of-speech to obtain the correct lemma.

# question

Assume that we modify the costs incurred for operations in calculating **Levenshtein distance**, such that both the insertion and deletion operations incur a cost of 1 each, while substitution incurs a cost of 2. Now, for the string 'ceating' which of the following set of strings will have an edit distance of 1

- (a) cheating, beating, eating
- (b) cheating, eating, creating
- (c) cheating, eating, casting
- (d) None of these

# References

- “Linguistics, An Introduction to Language and Communication” by Adrian Akmajian, Richard A. Demers, Ann K. Farmer and Robert M. Harnish (5th Edition)
- “SPEECH and LANGUAGE PROCESSING, An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition” by Daniel Jurafsky and James H. Martin (Second Edition)
- “Problems Encountered By EFL Secondary School Students in Using English Derivational Morphemes”  
Daifallah Siddig Mohammed Ahmed A Case Study of Singa Secondary School, Singa Locality, Sinnar State, Sudan (2017)