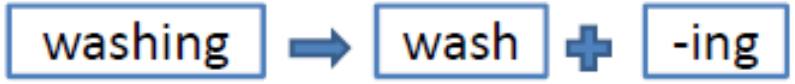
# Unit 3

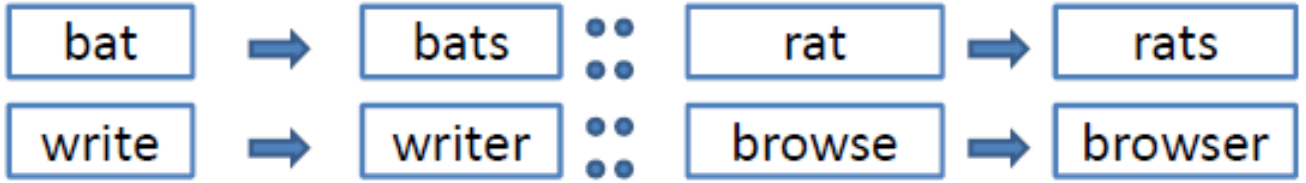
# Morphology and Finite-State Transducers

# Morphology

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




  - How they are **formed**?




- Morphology tries to formulate rules

# What is Morphology

- **Morph** = form or shape, **ology** = study of
- Morphology is the study of the way words are built up from smaller meaning-bearing units, ***morphemes***.
- A morpheme is often defined as the minimal meaning-bearing unit in a language.

**Example:**

| Singular | Plural |
|---|---|
| Fox | Foxes |
| Peccary | Peccaries |
| Goose | Geese |
| Fish | Fish |

- The word fox consists of a single morpheme (the morpheme fox)
- The word cats consists of two: the morpheme cat and the morpheme -s

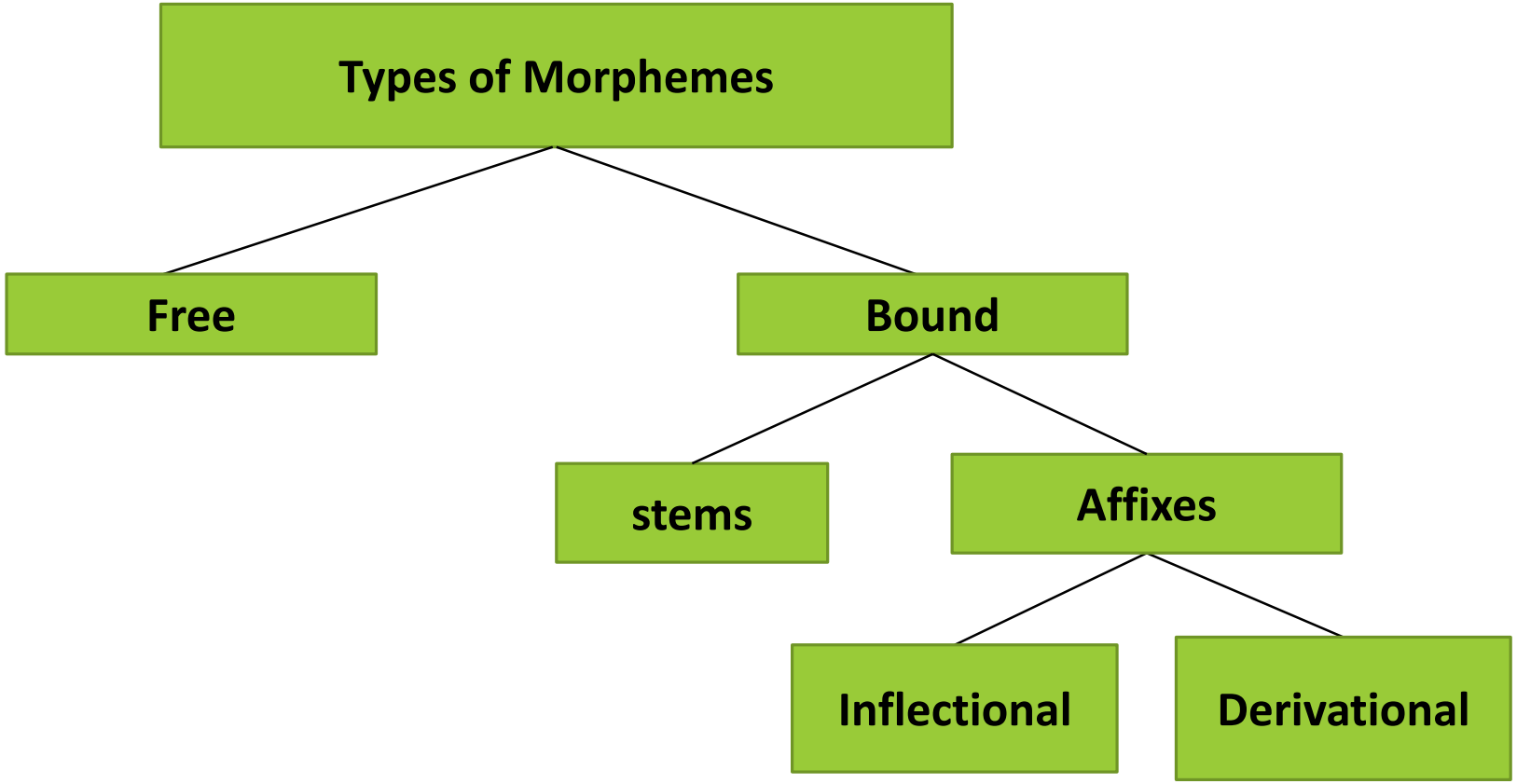
# Morphological Parsing

***The problem of recognizing that foxes break down into two morphemes fox and -es is called morphological parsing***

- ***Parsing*** means taking an input and producing some sort of structure for it
- In information retrieval domain, the similar problem of mapping from ***foxes*** to ***fox*** is called ***stemming***

- It takes two kinds of knowledge to correctly search for singulars and plurals of these forms:
  - **Spelling rules :** tells that English words ending in **-y** are pluralized by changing the **-y** to **-i** and adding an **- es**.
  - **Morphological rules :** tells that **fish** has null plural, and the plural of **goose** is formed by changing the vowel.

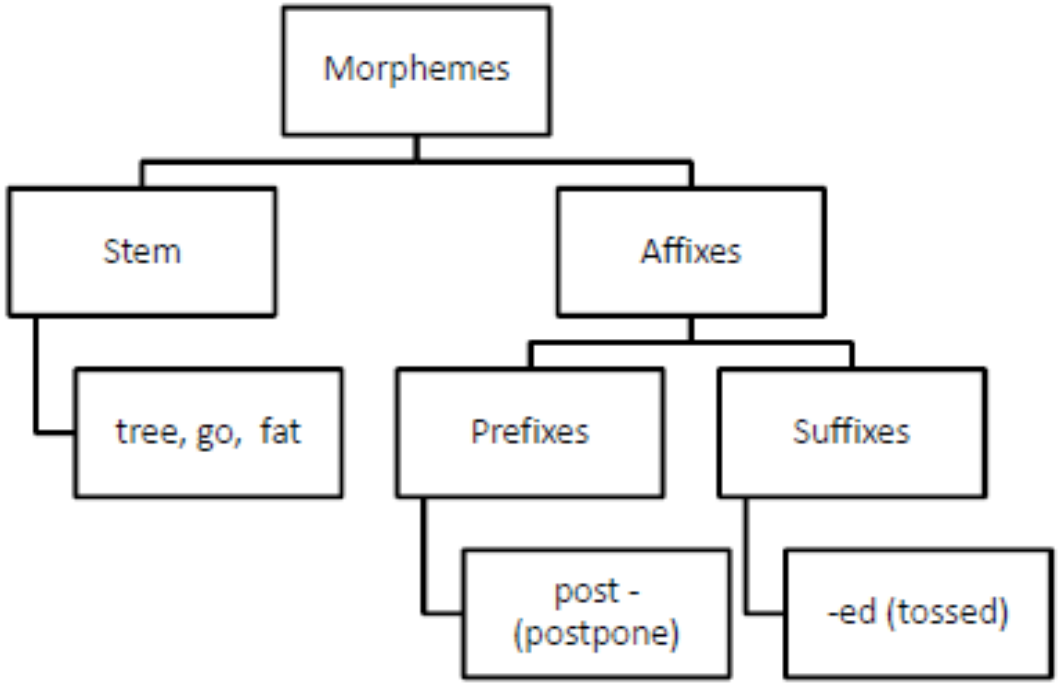
# Types of morphemes

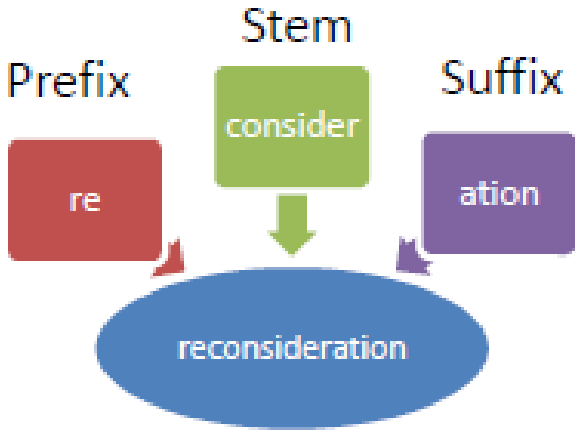


# Survey of English Morphology

- Two broad classes of morphemes
  - **Stems** – is the "***main***" morpheme of the word, supplying the main meaning
  - **Affixes** – add "***additional***" meanings of various kinds
    - **Prefixes** – precede the stem  Eg: the word **unbuckle** composed of ***buckle***  and prefix ***-un***
    - **Suffixes** – follow the stem  Eg: the word **eats** composed of stem ***eat*** and suffix ***-s***
    - **Circumfixes** – do both prefix and suffix operation  Ex: the word **enlighten** has ***en*** as prefix and suffix
    - **Infixes** – inserted inside the stem and generally found in plural forms in English. Eg: plural form of ***cupful*** is ***cupsful*** and ***spoonful*** is ***spoonsful***.

# Survey of English Morphology



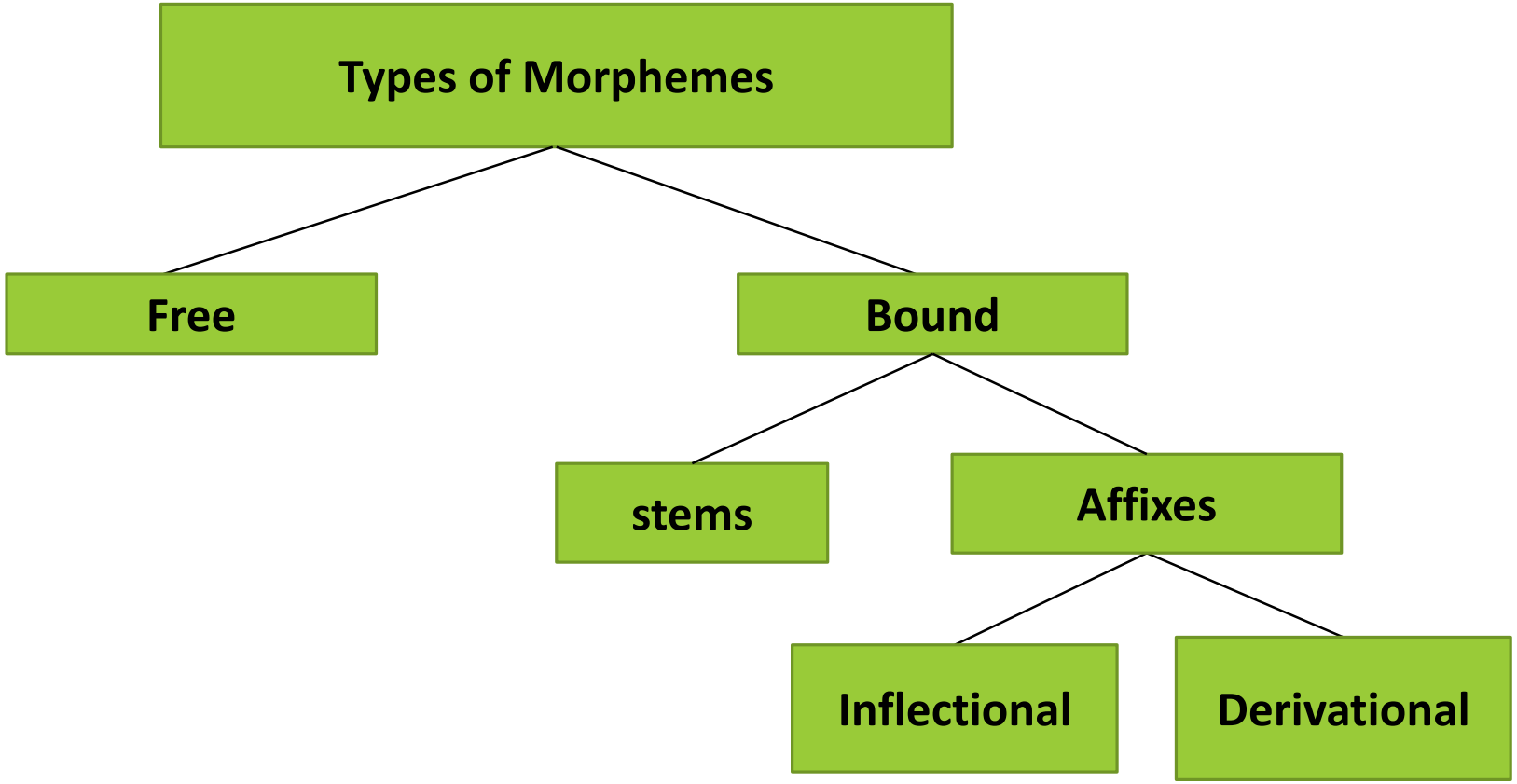
# Survey of English Morphology - Another classification of morphology

- **Concatenative morphology** –Prefixes and suffixes are often called concatenative morphology
- word is composed of a number of morphemes concatenated together or two morphemes are ordered one after another i.e. affixation and compounding
  - ***Affixation***: involves attachment of morphemes to stems. (Suffix, prefix, circumfixes, infix)
    - Example: multiple affixations: **anti- inter- govern -ment -al-ist .** Here govern is root or stem.
  - ***Compounding***: New words formed by combining two stems. Can be formed with many parts of speech.
    - Example**: noun-noun : horse-shoe,**
      **noun-verb : trouble-shoot,**
      **adjective- verb : high-jump,**
      **adjective-adjective: bitter - sweet**

# Survey of English Morphology - Another classification of morphology Contd.

- **Non-Concatenative morphology (templatic morphology)-** morphemes are combined in more complex ways
  - ***Reduplication*** : process that involves taking part of the base and attaching it as an affix; description involves how much is copied.
    - Examples: *bang-bang* : sound of a gun when firing, *bye-bye* : goodbye
  - ***Internal modification (vowel modification)*** : grammatical opposition expressed via a vowel alternation
    - Examples: *sing – sang – sung- song, begin-began, goose-geese , bind- bound*
  - ***Consonant modification :*** changes made for other than vowel
    - Examples: *belief-believe, grief-grieve*
  - ***Mixed modification :*** present/past , verb/noun
    - Examples: *catch-caught, seek-sought, live-life, defence-defend, bent-bend*

# Types of morphemes



# Ways to form words from Morphemes

Two broad classes to form words from morphemes:

I. **Inflectional Morphology** : the combination of a word stem with a grammatical morpheme usually resulting in a word of the same class as the original stem

II. **Derivational Morphology** : the combination of a word stem with a grammatical morpheme usually resulting in a word of a different class, often with a meaning hard to predict exactly

# Inflectional Morphology

- In English, only nouns, verbs, and sometimes adjectives can be inflected. The number of affixes is small
- English nouns have only two kinds of inflection:
  - An affix that marks plural
  - An affix that marks possessive
- Regular nouns are nouns that can be converted into their plural form by simply adding **"-s"** and **"-es"** to their end, whereas irregular nouns are nouns that do not follow a standard rule in converting plurals.

An affix marking plural

| | |
|---|---|
| cat(-s) | finch(-es), |
| ibis(-es), | box(-es) |
| thrush(-es) | butterfly(-lies) |
| ox (oxen) [irregular nouns] | waltz(-es) |
| mouse (mice) [irregular nouns] | |

An affix that marks possessive (realized by apostrophe's)

| | |
|---|---|
| **Regular singular noun** | Llama's |
| **Plural nouns not ending in –s** | Children's |
| **Regular plural nouns** | Llamas' |
| **Names ending in –s or –z** | Euripides' , comedies' |

# Inflectional morphology

- Verbal inflection is more complicated than nominal inflection.
- English has three kinds of verbs:
  - Main verbs *(eat, sleep, impeach)*
  - Modal verbs *(can, will, should)*
  - Primary verbs *(be, have, do)*
- Of these verbs, main and primary verbs are the ones having inflectional endings.
- All verbs of this class have the same endings marking the same functions.

# Inflectional morphology Contd.

**Morphological forms of Regular verbs**

- Just by knowing the stem we can predict the other forms
  - By adding one of the three predictable endings
  - Making some regular spelling changes
- These regular verbs and forms are significant in the morphology of English because they cover majority of verbs and regular class is productive.
- Regular verbs have four morphological forms, as follows:

| **Morphological form classes** | **Regularly Inflected Verbs** | | | |
|---|---|---|---|---|
| **Stem** | walk | merge | try | map |
| *-s* form | walks | merges | tries | maps |
| *-ing* participle | walking | merging | trying | mapping |
| Past form or *-ed* participle | walked | merged | tried | mapped |

# Inflectional morphology Contd.

**Morphological forms of Irregular verbs**

- Often have five different forms, but can have as many as eight (eg. Verb ***be***) or as few as three (eg. ***cut*** or ***hit***).

| **Morphological Form Classes** | **Irregularly Inflected Verbs** | | |
|---|---|---|---|
| **Stem** | eat | catch | cut |
| -*s* form | eats | catches | cuts |
| -*ing* participle | eating | catching | cutting |
| Past form | ate | caught | cut |
| -*ed* participle | eaten | caught | cut |

# Derivational Morphology

- **Normalization:** formation of new nouns, often from verbs or adjectives.

| Suffix | Base Verb/Adjective | Derived Noun |
|---|---|---|
| -ation | computerize (V) | computerization |
| -ee | appoint (V) | appointee |
| -er | kill (V) | killer |
| -ness | fuzzy (A) | fuzziness |

- Adjectives can also be derived from nouns and verbs.

| Suffix | Base Noun/Verb | Derived Adjective |
|---|---|---|
| -al | computation (V) | computational |
| -able | embrace (V) | embraceable |
| -less | clue (N) | clueless |

# Derivational Morphology

- Derivation in English is more complex than inflection because
  - It is generally less productive
  - A nominalizing affix like *–ation* cannot be added to every verb. Thus, cannot be said as *eatation, spellation*
  - There are subtle and complex meaning differences among nominalizing suffixes.

# Finite-State Morphological Parsing

- Aim is to take the input forms like those in first column and produce output forms like those in second column

- Second column contains stem of each word and asserted morphological features. (asserted features specify additional information of the stem)
  - Example: *+N for Noun, +SG for singular, +PL for plural.*

| Input | Morphological Parsed Output |
|---|---|
| cats | cat + N + PL |
| cat | cat + N + SG |
| cities | city + N + PL |
| geese | goose + N + PL |
| goose | (goose + N + SG) |
| merging | (merge + V + PRES-PART) |
| caught | (catch + V + PAST-PART) or (catch + V + PAST ) |

# Finite-State Morphological Parsing

To build a morphological parser, the following are needed:

i. **Lexicon:** the list of stems and affixes, together with basic information about them

   Eg., Noun stem or Verb stem, etc.

ii. **Morphotactics:** the model of morpheme ordering that explains which classes of morphemes can follow other classes of morphemes inside a word.

   E.g., the rule that English plural morpheme follows the noun rather than preceding it.

iii. **Orthographic rules: spelling rules** are used to model the changes that occur in a word, usually when two morphemes combine

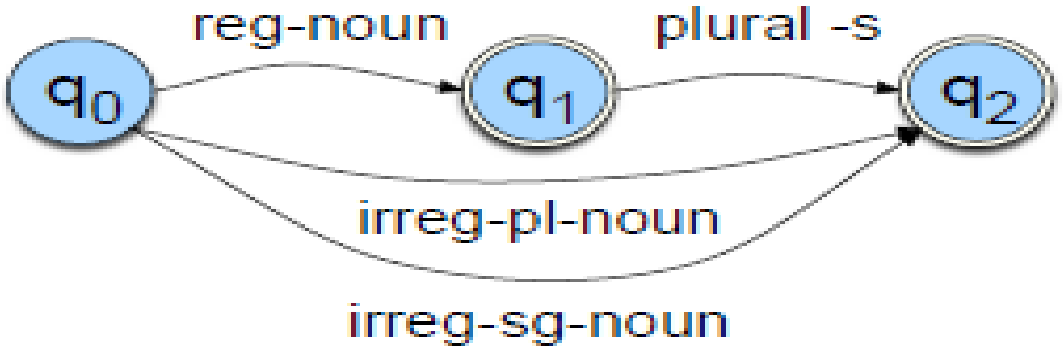
   E.g., the **y→ie** spelling rule changes **city + -s** to **cities** instead of **citys**.

# Lexicon and Morphotactics

- A lexicon is a repository for words.
  - The simplest one would consist of an explicit list of every word of the language. i.e. including abbreviations and proper names.
  - Example: a, AAA, AA, Aachen, aardvark, aba, abaca, ……
- Computational lexicons are usually structured with
  - a list of each of the stems and
  - Affixes of the language together with a representation of morphotactics telling us how they can fit together.
- The most common way of modeling morphotactics is the **finite-state automaton**.

# Lexicon and Morphotactics Contd…

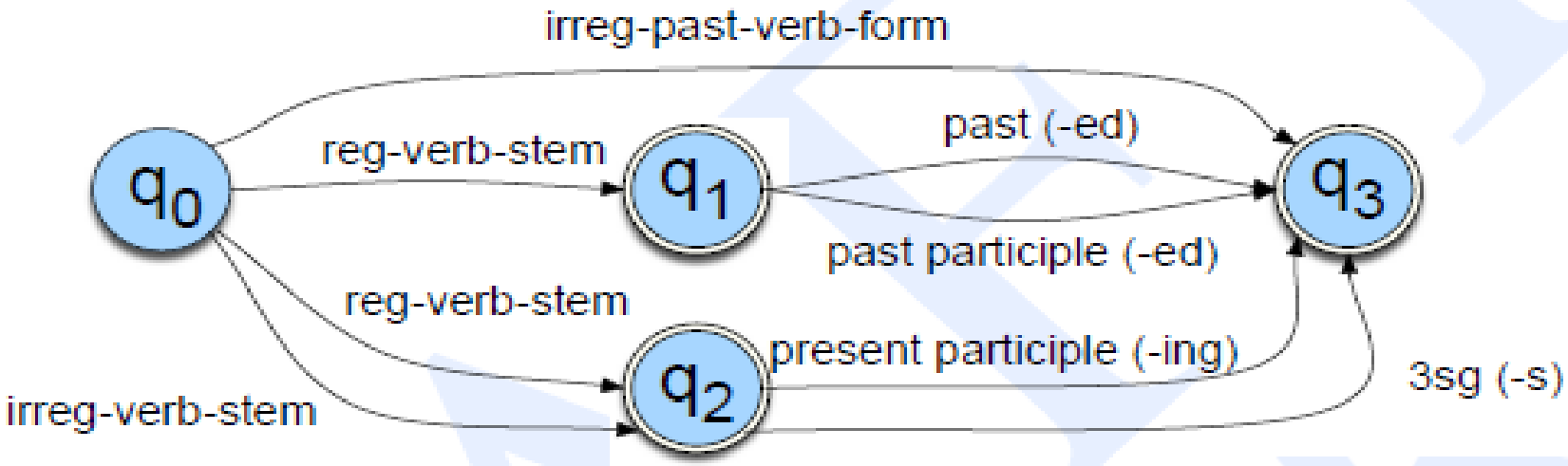
## A finite-state automaton for English nominal inflection




| reg-noun | irreg-pl-noun | irreg-sg-noun | plural |
|---|---|---|---|
| fox | geese | goose | -s |
| cat | sheep | sheep | |
| aardvark | mice | mouse | |

# Lexicon and Morphotactics Contd.

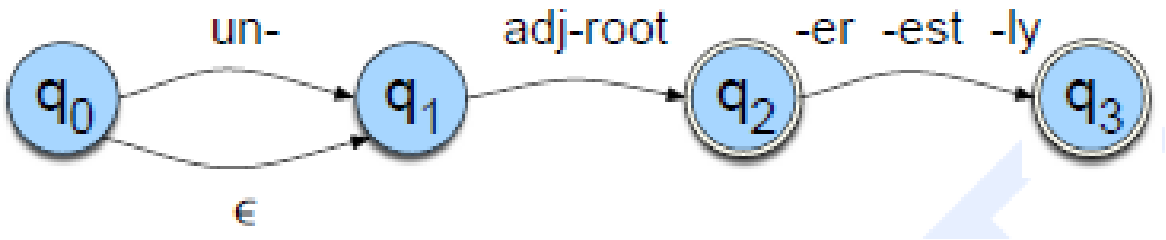
**A finite-state automaton for English verbal inflection**




| reg-verb-stem | irreg-verb-stem | irreg-past-verb | past | past-part | pres-part | 3sg |
|---|---|---|---|---|---|---|
| walk | cut | caught | -ed | -ed | -ing | -s |
| fry | speak | ate | | | | |
| talk | sing | eaten | | | | |
| impeach | | sang | | | | |

# Lexicon and Morphotactics Contd.

- English derivational morphology is more complex than English inflectional morphology, and so automata of modeling English derivation tends to be quite complex.
  - Some are even based on Context free Grammers
- A small part of morphosyntactics of English adjectives is shown:




**An FSA for a fragment of English adjective morphology: #1**

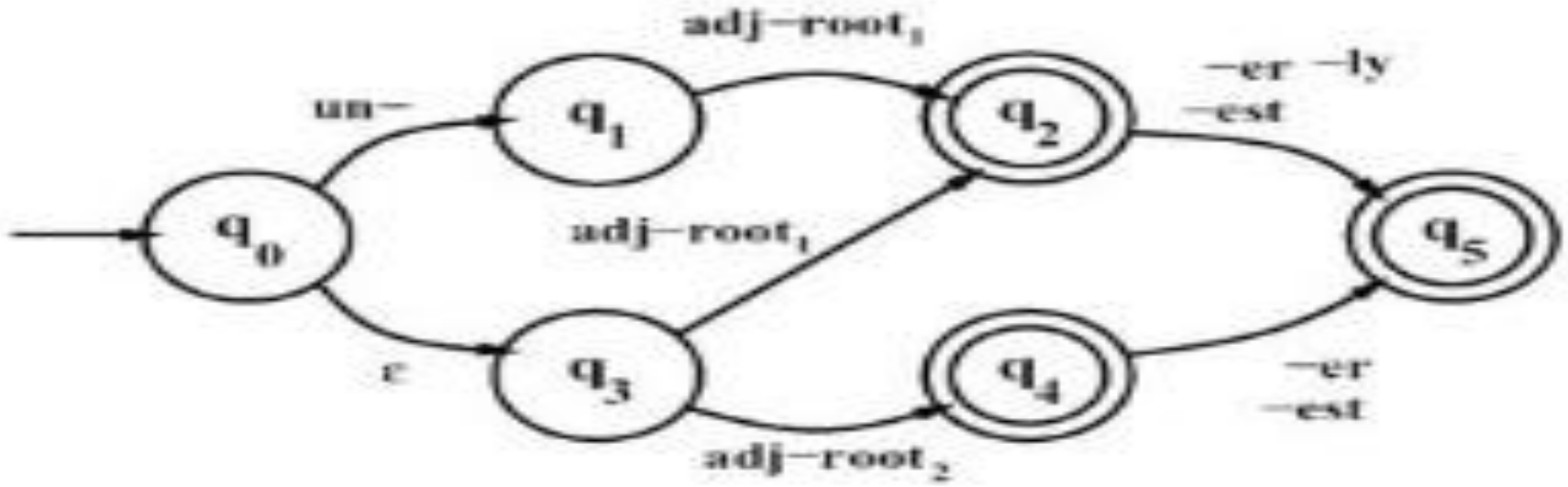
| |
|---|
| big, bigger, biggest |
| cool, cooler, coolest, coolly |
| red, redder, reddest |
| clear, clearer, clearest, clearly, unclear, unclearly |
| happy, happier, happiest, happily |
| unhappy, unhappier, unhappiest, unhappily |
| real, unreal, really |

# Lexicon and Morphotactics Contd.

- The FSA#1 recognizes all the listed adjectives, and ungrammatical forms like unbig, redly, and realest.
- Need to setup classes of roots and specify which can occur with which suffixes. Thus #1 is revised to become #2.
  - adj-root$_1$ would include adjectives that can occur with un- and -ly (clear, happy)
  - adj-root$_2$ will include adjectives that can't occur with un- and -ly (big, red)

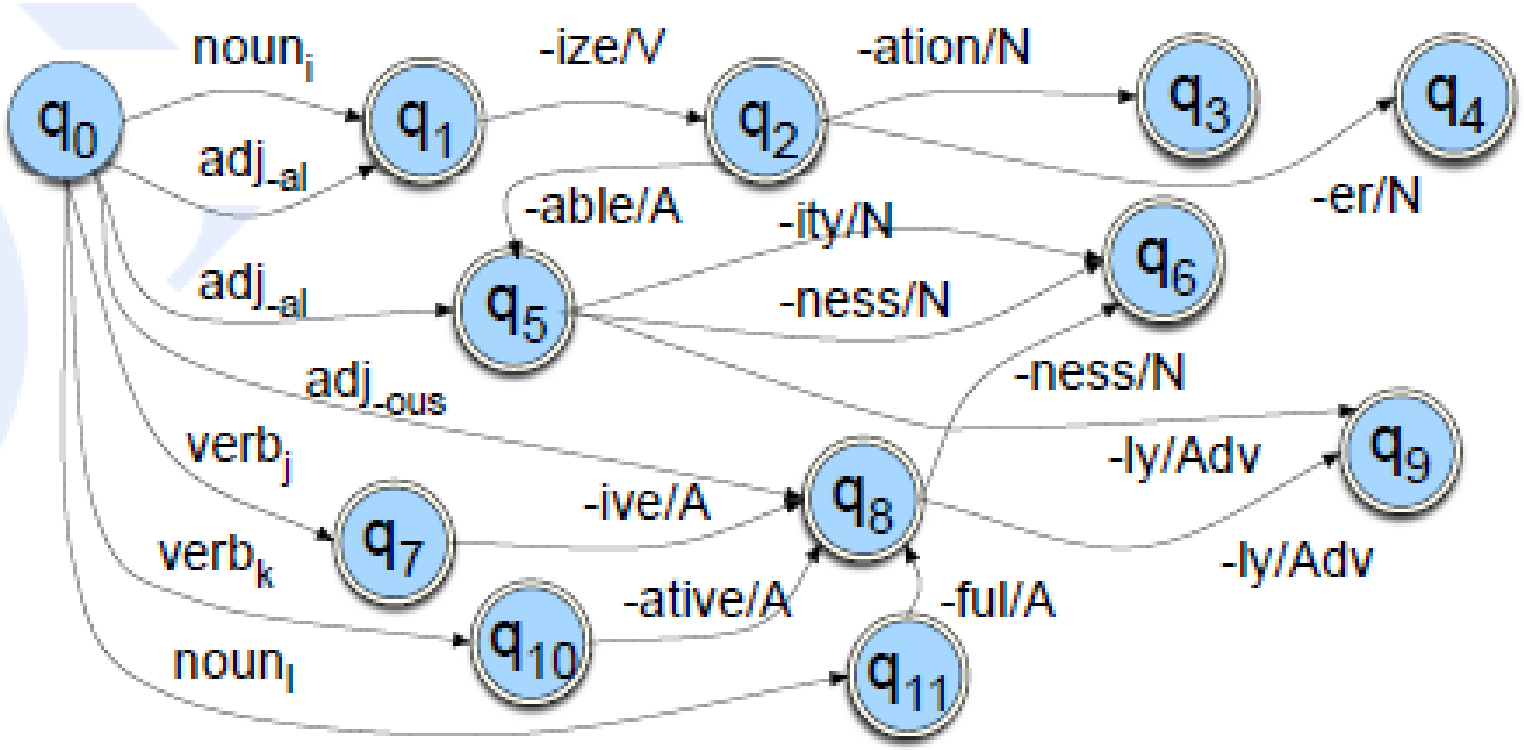
**An FSA for a fragment**

**of English adjective Morphology #2**



# Lexicon and Morphotactics Contd.

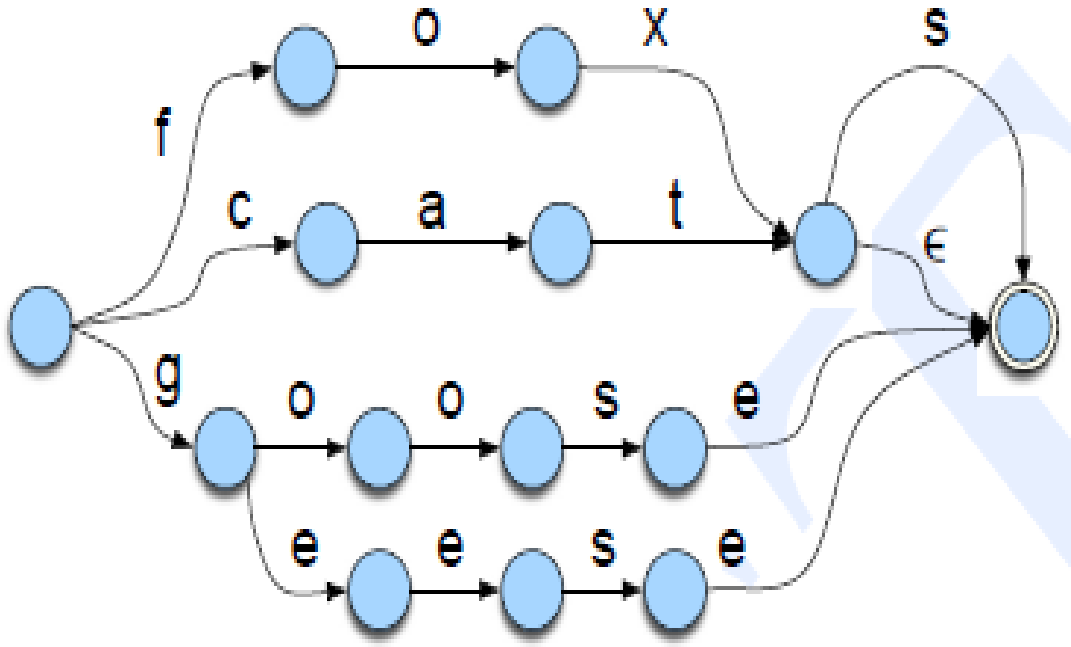
**A FSA for English nominal and verbal derivational morphology**




- Verb ending with ***-ize*** can be followed by nominalizing suffix ***-ation***
- For the word ***fossilize*** , the word ***fossilization*** can be predicted by following the states $q_0$, $q_1$, $q_2$.

# Lexicon and Morphotactics Contd.

- FSAs can be used to solve the problem of **morphological recognition**.
- **Morphological recognition**: determining whether an input string of letters makes up a legitimate English word or not.
- This is done by taking morphotactic FSAs, and plugging in each "***sub-lexicon***" into FSA. (i.e. expand each arc (eg. the **reg-noun-stem** arc)with all the morphemes that make up the set of **reg-noun-stem**.)
- The resulting FSA can then be defined at the level of the individual user.




**Compiled FSA for a few English nouns with their inflections**

# Finite State Transducers

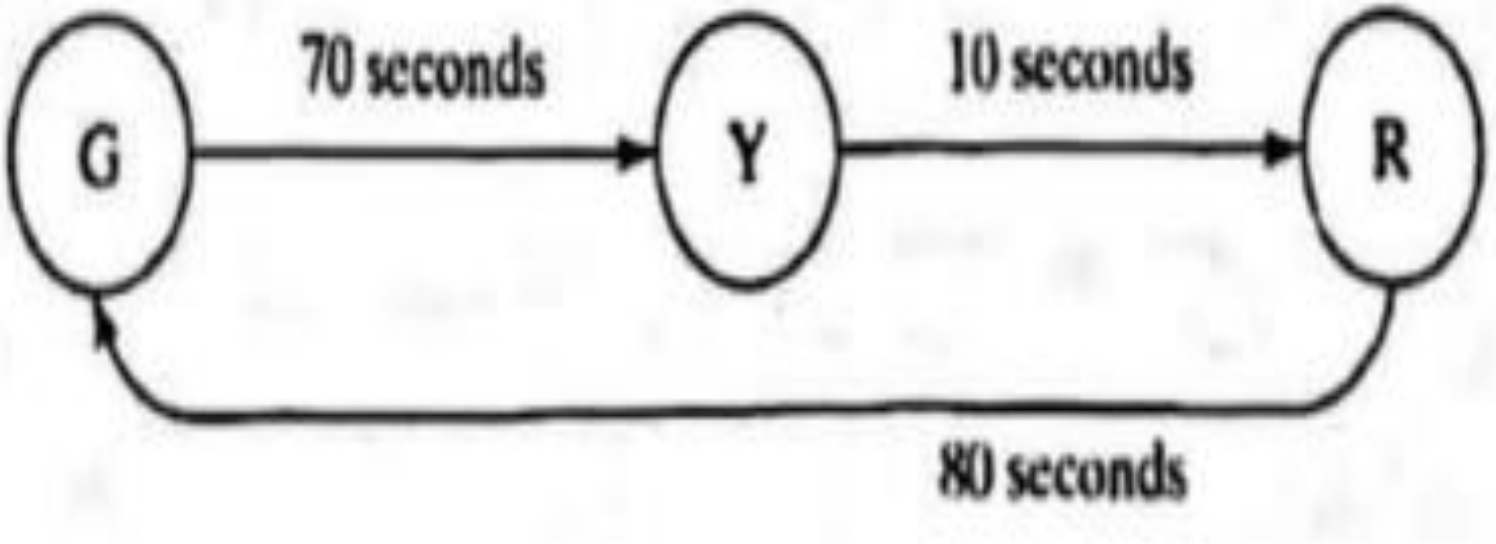
Finite State Transducers are loops that simply run forever, processing inputs

- An automaton that produces outputs based on current input and/or previous state is called a transducer
- Transducers can be of two types:
  - Moore Machine -The output depends only on the current state
  - Mealy Machine- The output depends both on the current state and the current input

# Finite State Transducers

***Moore Machine:*** The output depends only on the current state.

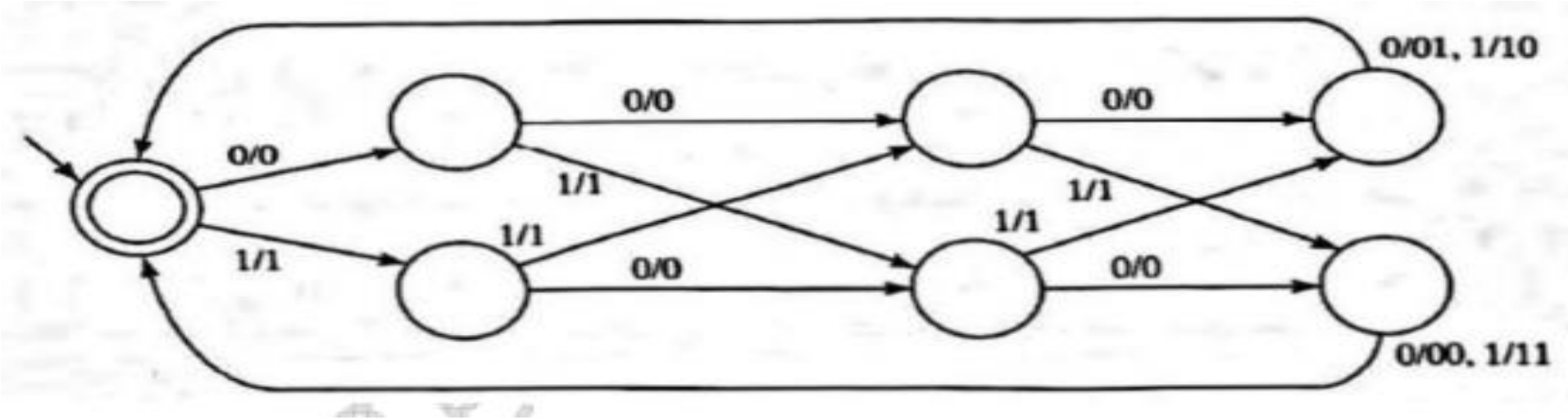
Example: Traffic light



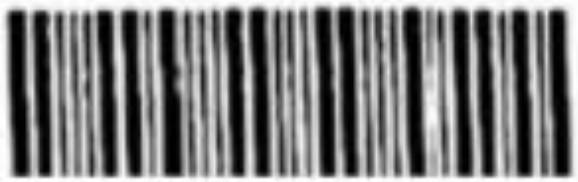
# Finite-State Transducers

**2. *Mealy Machine:*** The output depends both on the current state and the current input

I. Generating parity bits (adds odd parity bit after every four bits)

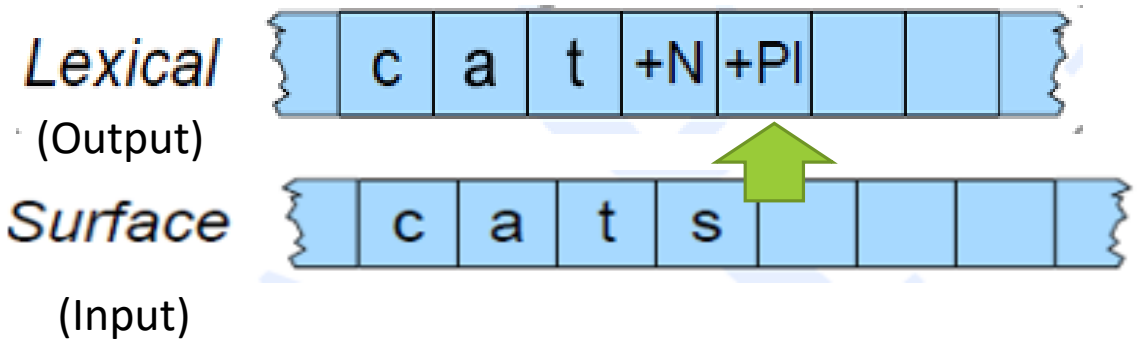



II. Bar code reader



# Morphological Parsing with Finite-State Transducers

- Given the input, for example, ***cats***, we would like to produce ***cat +N +PL***.
- Two-level morphology, by Koskenniemi (1983)
  - **The lexical level** represents a simple concatenation of morphemes making up a word
  - **The surface level** which represents the actual spelling of the final word.
- Morphological parsing is implemented by building mapping rules that maps letter sequences like ***cats*** on the surface level into morpheme and features sequence like ***cat +N +PL*** on the lexical level.




- Figure shows two levels: lexical and surface level for the word ***cats***.
- Lexical level has the stem word followed by morphological information ***+N +PL*** that tells ***cats*** is a plural noun

# Morphological Parsing with Finite-State Transducers

- The automaton we use for performing the mapping between these two levels is the **Finite-State Transducer or FST**.
- A transducer maps one set of symbols with another set via a **finite automaton**.
- Thus an FST can be seen as a **two-tape automaton which recognizes or generates pairs of strings**.
- The FST has a more general function than an FSA:
  - An FSA defines a formal language by defining set of strings, whereas FST defines a relation between sets of strings.

1) FSA **recogniser** for "fox"


2) FST **transducers** for fox/fox; goose/geese


# Morphological Parsing with Finite-State Transducers

- Another view of an FST is that it is a machine that reads one string and generates another.

**$L_{IN}$ = {cat, cats, fox ,foxes, …}**

**$L_{out}$={cat +N +SG, cat +N +PL, fox +N +SG, fox +N +PL …}**

**T = {<cat, cat +N +SG>,< cats, cat +N +PL>, < fox, fox +N +SG>, < foxes, fox +N +PL >…}**

# Morphological Parsing with Finite-State Transducers

Four-fold way thinking about transducers

- **FST as recognizer:** a transducer that takes (recognizes) a pair of strings as input and outputs ***accept*** if the string-pair is in the string-pair language, and a ***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 transducer(translator):** A machine that reads a string and outputs another string.
  - Morphological parsing: letter(input), morphemes (output)
- **FST as set relater:** A machine that computes relations between sets.

# Morphological Parsing with Finite-State Transducers

- A formal definition of FST (Mealy machine extension to a simple FSA):
  - **Q**: a finite set of N states $q_0, q_1, \ldots, q_N$
  - **Σ**: a finite alphabet of complex symbols.

    Each complex symbol is composed of an input-output pair i : o. I and O may each also include the epsilon symbol ε.
  - $\mathbf{q_0}$: the start state
  - **F**: the set of final states, $F \subseteq Q$
  - **δ(q, i:o):** the transition function or transition matrix between states. Given a state **q ϵ Q** and complex symbol **i:o ϵ Σ**, **δ(q, i:o)** returns a new state **q' ϵ Q**. **δ** is thus a relation from **Q x Σ** to **Q**

# Morphological Parsing with Finite-State Transducers

- **Two useful closure properties of FSTs:**
  - ***Inversion:*** Inversion of transducers switches input and output labels. Thus, If **T** maps from Input alphabet **I** to output alphabet **O**, then the inverse of **T**, $\mathbf{T^{-1}}$ maps from **O** to **I**.
  - Inversion is useful because it makes it easy to convert a **FST-as-parser into an FST-as-generator**.
  - ***Composition:*** If $\mathbf{T_1}$ is a transducer from $\mathbf{I_1}$ to $\mathbf{O_1}$ and $\mathbf{T_2}$ a transducer from $\mathbf{I_2}$ to $\mathbf{O_2}$, then $\mathbf{T_1 \circ T_2}$ maps from $\mathbf{I_1}$ to $\mathbf{O_2}$
  - Composition is useful because it allows us to take two transducers that run in series and replace them with one complex transducer.
  - $\mathbf{T_1 \circ T_2(S) = T_2(T_1(S))}$

# Morphological Parsing with Finite-State Transducers

- Morphological parser maps the surface forms to lexical forms by ***cascading*** the lexicon with singular or plural automaton.
- **Cascading:** running two automata in series with the output of first feeding the input to the second
- lexicon of stems are represented as FST $T_{stems}$ . Example: dog to ***reg-noun-stem***
- For suffixes: $T_{stems}$ allows the forms to be followed by ***@:@*** (any feasible pair)
- Alternative to cascading is **composing** the transducer using **composition operator**.
- ***Composing*** is a way of taking a cascade of transducers with many different levels of inputs and outputs and converting them into single ***"two level"*** transducer with one input tape and one output tape.
- Composed automaton is $T_{lex}=T_{num} \circ T_{stems}$

# Morphological Parsing with Finite-State Transducers

- Regular expressions can be written in the complex alphabet **Σ** as in FSA
- For two-level morphology, FST is viewed as having two level tapes.
  - **The upper or lexical tape, The lower or surface tape**
- Each symbol **a:b** in the transducer alphabet **Σ** expresses how the symbol ***a*** from one tape is mapped to the symbol ***b*** on the another tape
  - Example: ***a:ε*** means ***a*** on the upper tape will correspond to nothing in lower tape.
- Generally, symbols map to themselves, in two level morphology. Hence called as ***default pairs.*** Example**: a:a (referred to this by single letter a)**
- A simple pair of symbols in Σ are called ***feasible pairs***. Example**: a:!, a:ε**

# Morphological Parsing with Finite-State Transducers

- FSA accepts a language stated over a finite alphabet of single symbols.

  Example: alphabet of sheep language **Σ={b,a,!}**

- FST accepts a language stated over a pair of symbols

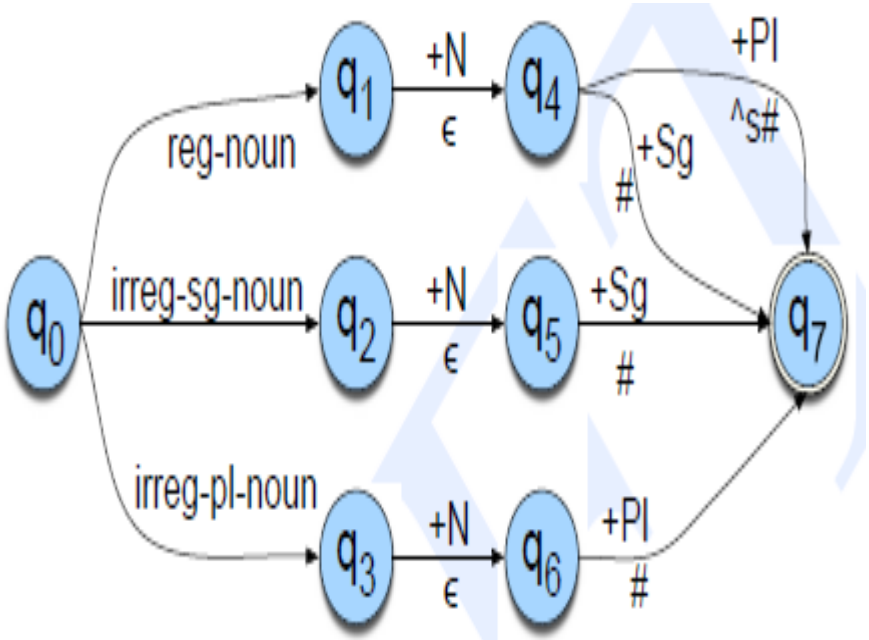
  Example: **Σ={a:a, b:b, !:!, a:!, a: ε, ε:!}**

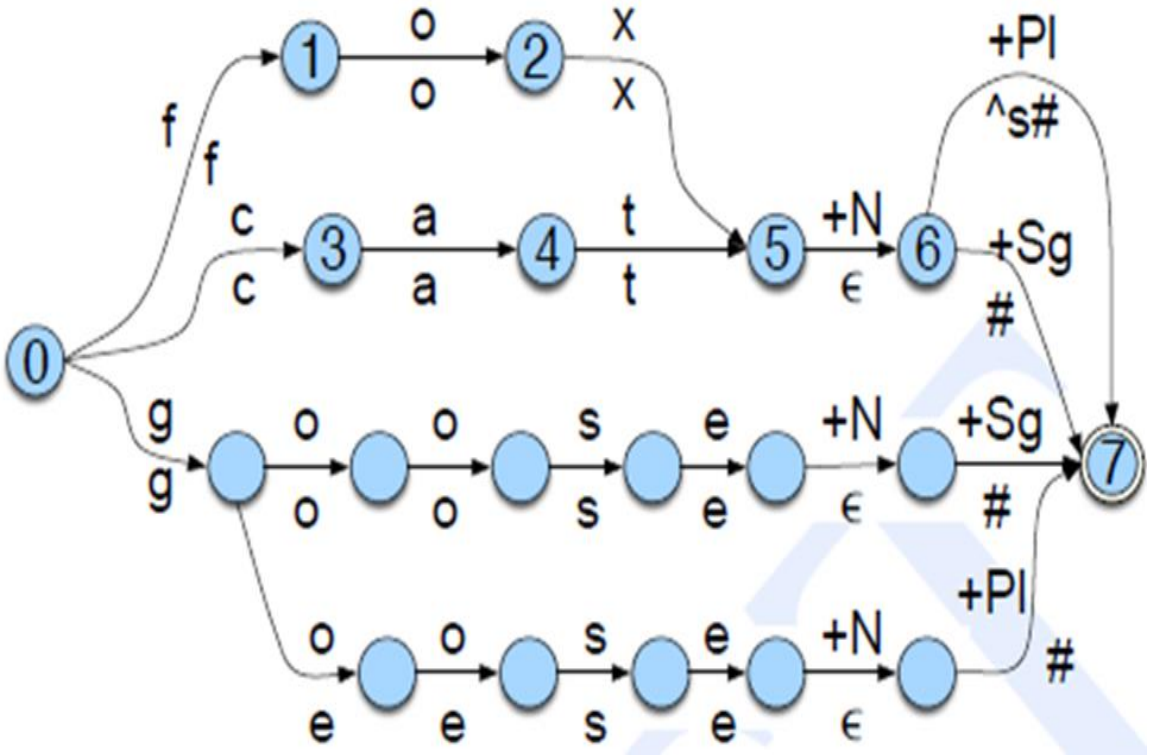
- FSAs are isomorphic to regular languages, FSTs are isomorphic to regular relations.
  - Regular relations are sets of pairs of strings, an extension of the regular language, which are sets of strings.

# Morphological Parsing with Finite-State Transducers

- **Application**
- An FST morphological parser is built for earlier morphotactic FSAs and lexica by adding an extra ***"lexical"*** tape and appropriate morphological features.
  - Nominal morphological features map to the empty string **ε** or the word/morpheme boundary symbol **#** since there is no segment corresponding to them on the output tape.

# Morphological Parsing with Finite-State Transducers



# Morphological Parsing with Finite-State Transducers

- For morphological noun parser, all the individual regular and irregular noun stems are augmented.
  - Lexicon of the transducer is updated so that regular plurals like **geese** will parse into correct stem **goose +N +PL**.

    Example: surface geese maps to underlying goose, the new lexical entry will be “g:g o:e o:e s:s e:e”.

  - Regular forms are simpler. Two level entry for fox will be **“f:f o:o x:x”**, but relying on orthographic convention that **f** stands for **f:f**, **o** stands for **o:o** etc. it can be referred to as "**fox**" and goose as **“g o:e o:e s e”**.

# Morphological Parsing with Finite-State Transducers

- Maps plural nouns into stem plus the morphological marker **+PL**, and singular nouns into stem plus **+SG**.
- Thus, surface cats will map to **cat +N +PL** as :
  **c:c a:a t:t +N: ε +PL: ^ s #**
- Symbol **^** indicates morpheme boundary
- Symbol **#** indicates word boundary.

| reg-noun | irreg-pl-noun | irreg-sg-noun |
|---|---|---|
| fox | g o:e o:e s e | goose |
| cat | sheep | sheep |
| aardvark | m o:i u:e s:c e | mouse |

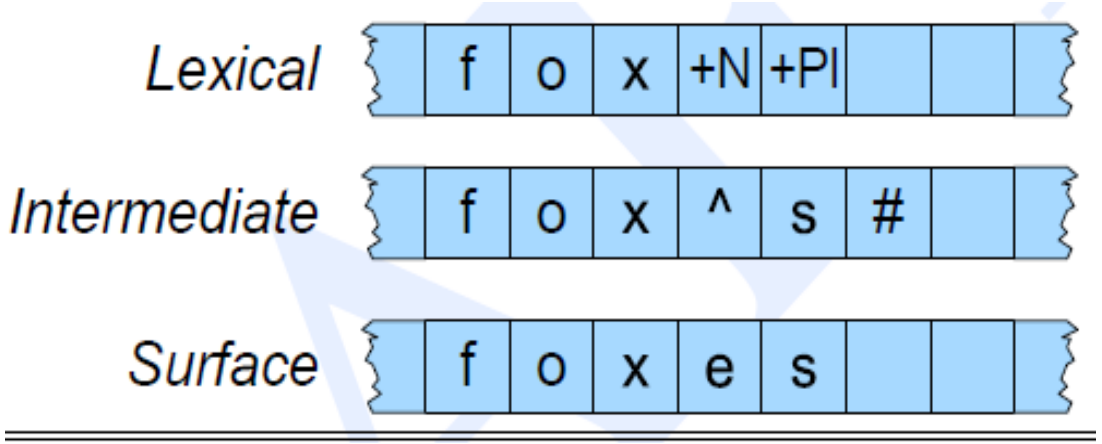
# Orthographic rules and Finite-State Transducers

- Concatenating the morphemes can work to parse the words like **“dog”, “cat”, “fox”.**
- But this simple method does not work. *Example:* **“foxes”** is to be parsed into lexicons **“fox +N +PL”** etc.
- This requires introduction of spelling rules (also called orthographic rules).
- Some of the spelling rules:

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

# Orthographic rules and Finite-State Transducers

- To account for the spelling rules, another tape is introduced, called intermediate tape.
- ***Intermediate tape:*** produces the output slightly modified, thus going from 2-level to 3-level morphology
- Between each level of tapes is a two-level transducer.
  - **Lexical transducer** - between lexical and intermediate levels
  - **E-insertion spelling rule** - between the intermediate and surface levels




- E-insertion spelling rule inserts an **e** on the surface tape when the intermediate tape has a morpheme boundary **^** followed by the morpheme **-s**

# Orthographic rules and Finite-State Transducers

- The E-insertion rule states that : "***insert an e on the surface tape just when the lexical tape has a morpheme ending in x (or z, etc) and the next morpheme is -s***".

Formalization of the rule :

$$\epsilon \rightarrow e / \left\{ \begin{matrix} x \\ s \\ z \end{matrix} \right\} \hat{} \_\_\_ s\#$$

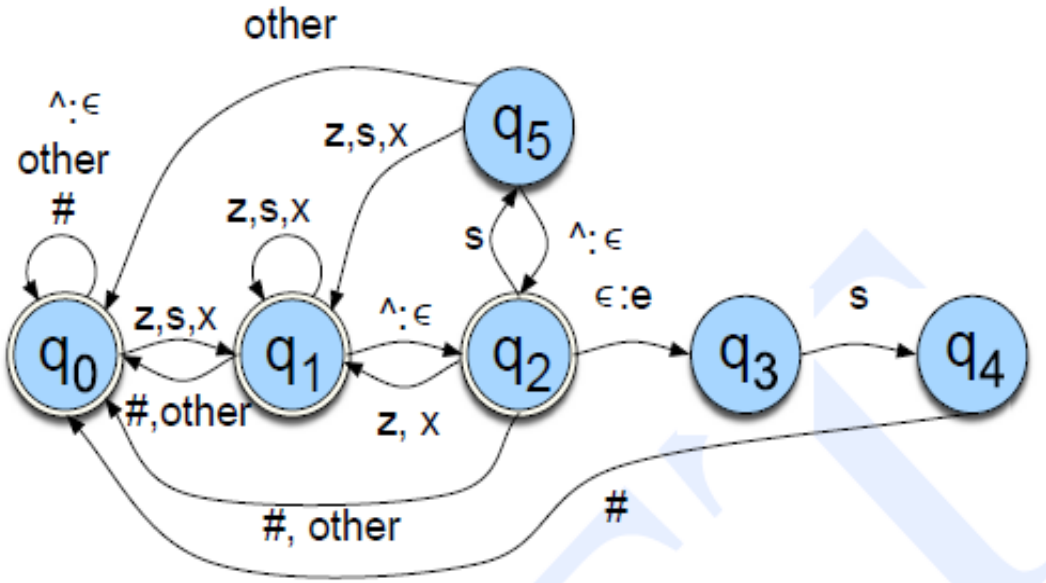
**Meaning: Insert e after a morpheme-final x, s, or z, and before the morpheme s**

This follows the rule notation of Chomsky and Halle(1968); (***a*** ⟶ ***b/c ___d*** means ***"rewrite a as b when it occurs between c and d"***)

- **ε** is the empty transition, replacing it means inserting something.
- the symbol **^** indicates morpheme boundary. Boundaries are deleted by including the symbol **^: ε** in the default pairs of transducer. Thus morpheme boundary markers are deleted on the surface level by default.
- The **#** symbol marks the word boundary.

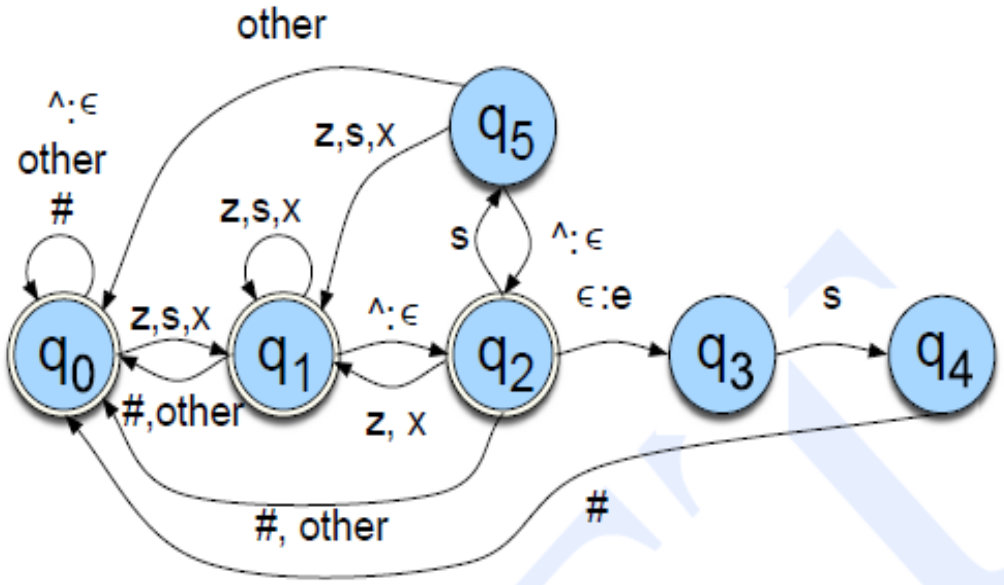
# Orthographic rules and Finite-State Transducers

- Building a transducer for any rule is to express only the constraints necessary for that rule, allowing any other strings of symbols to pass through unchanged. (i.e. it ensures only ***ε:e*** pair is seen in proper context)
- State $q_0$ models when seeing only default pairs unrelated to the rule. It is an accepting state.
- State $q_1$ models having seen **z, s,** or **x** and it's an accepting state
- State $q_2$ models having seen the morpheme boundary. It is also an accepting state
- State $q_3$ models having seen the **E-insertion**; ***it is not an accepting state since the insertion is only allowed if it is followed by the s morpheme and then the end-of-the-word symbol #***



# Orthographic rules and Finite-State Transducers

- The ***other*** symbol is used to pass through any parts of words that don't have a role in E-insertion rule. ***(other means "any feasible pair that is not in this transducer")***

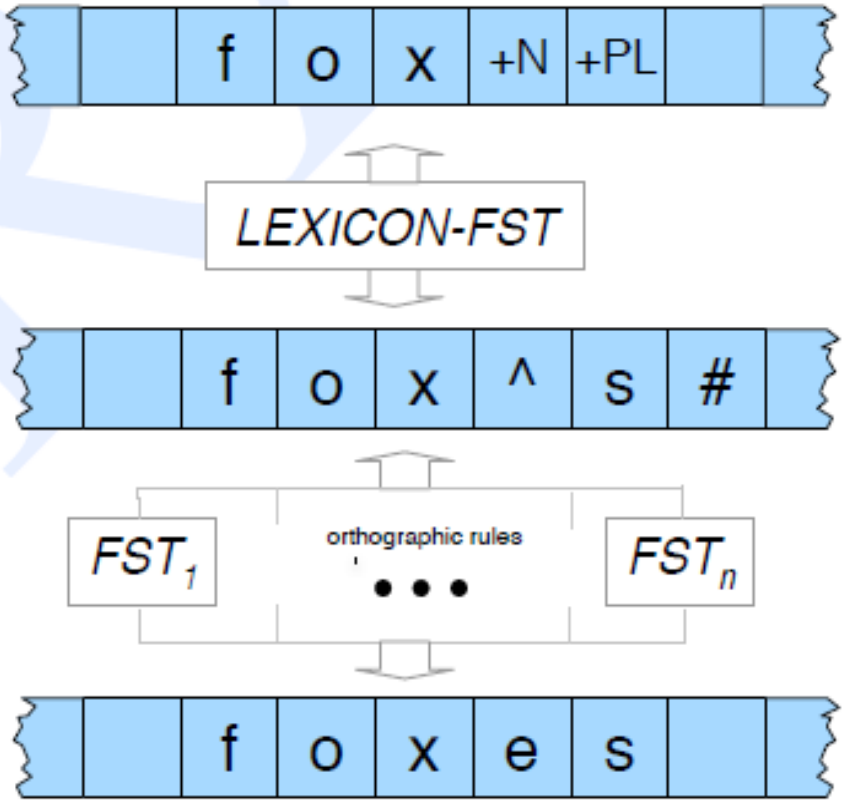



| State\Input | s:s | x:x | z:z | ^:ϵ | ϵ:e | # | other |
|---|---|---|---|---|---|---|---|
| $q_0$: | 1 | 1 | 1 | 0 | - | 0 | 0 |
| $q_1$: | 1 | 1 | 1 | 2 | - | 0 | 0 |
| $q_2$: | 5 | 1 | 1 | 0 | 3 | 0 | 0 |
| $q_3$ | 4 | - | - | - | - | - | - |
| $q_4$ | - | - | - | - | - | 0 | - |
| $q_5$ | 1 | 1 | 1 | 2 | - | - | 0 |

**State-transition table for E-insertion rule**

# Combining FST Lexicon and Rules

- Two-level morphology system used for **parsing or generating.**
- **Lexicon transducer** – maps between the ***lexical level*** (stems and morphological features) and an ***intermediate level*** (represents concatenation of morphemes)
- **Host of transducers** – ***(each represents a single spelling rule)*** – run in parallel; maps between ***intermediate level*** and ***surface level***
- ***Putting the spelling rules in parallel or in series (cascading) is a design choice.***
- The architecture is a **two level cascade of transducers**
- The output from one transducer acts as an input to another transducer
- ***The cascade can be run top-down to generate a string, or bottom-up to parse it***

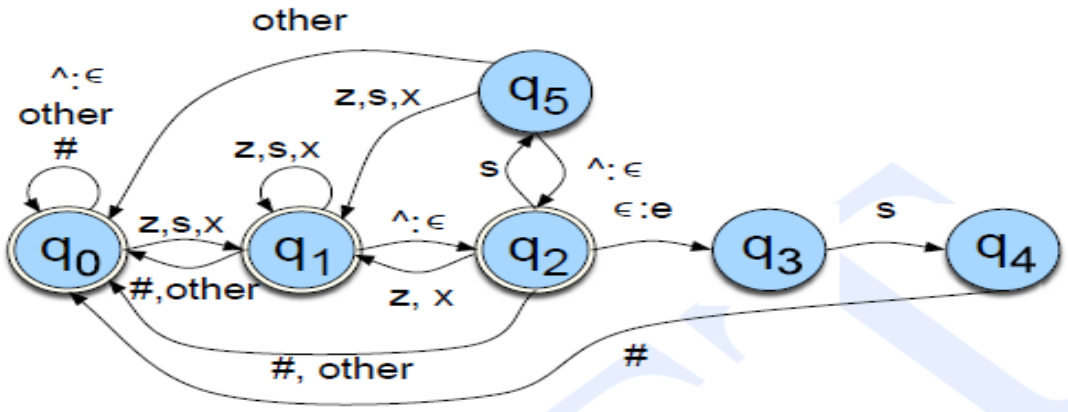


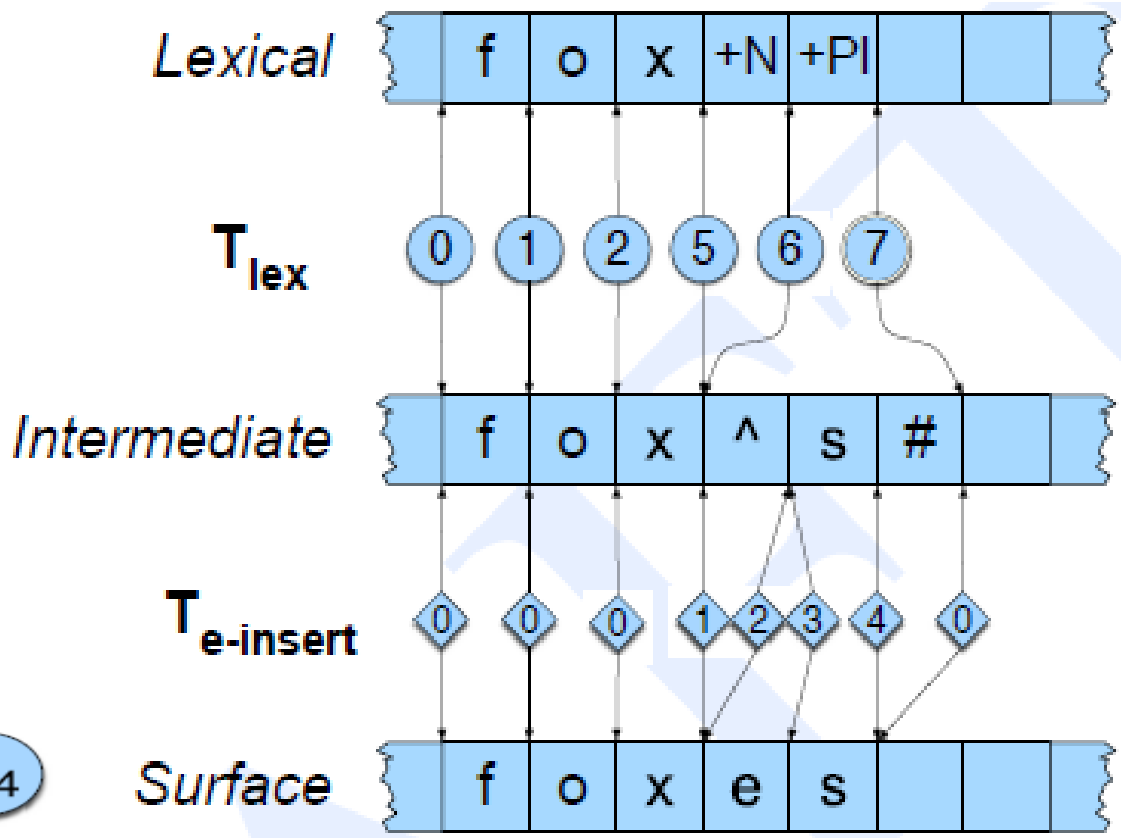

**Generating or parsing with FST lexicon and rules**

# Combining FST Lexicon and Rules

- Figure shows a trace of the system ***accepting*** the mapping from **fox +N +PL** to ***foxes***.

- **Generation:** **(surface tape from lexical tape)**
  - Running lexicon transducer, for ***fox +N +PL***, it will produce ***fox^#*** on the intermediate tape.
  - Running all possible orthographic transducers to run in parallel, will produce the ***foxes*** in the surface tape.




**Generating or parsing with FST lexicon and rules**

# Combining FST Lexicon and Rules

- **Parsing:** Complicated because of the problem of **"ambiguity"**
- Example: ***foxes*** can be verb and hence lexical parse for ***foxes*** could be ***foxes +V +3SG*** or ***fox +N +PL***

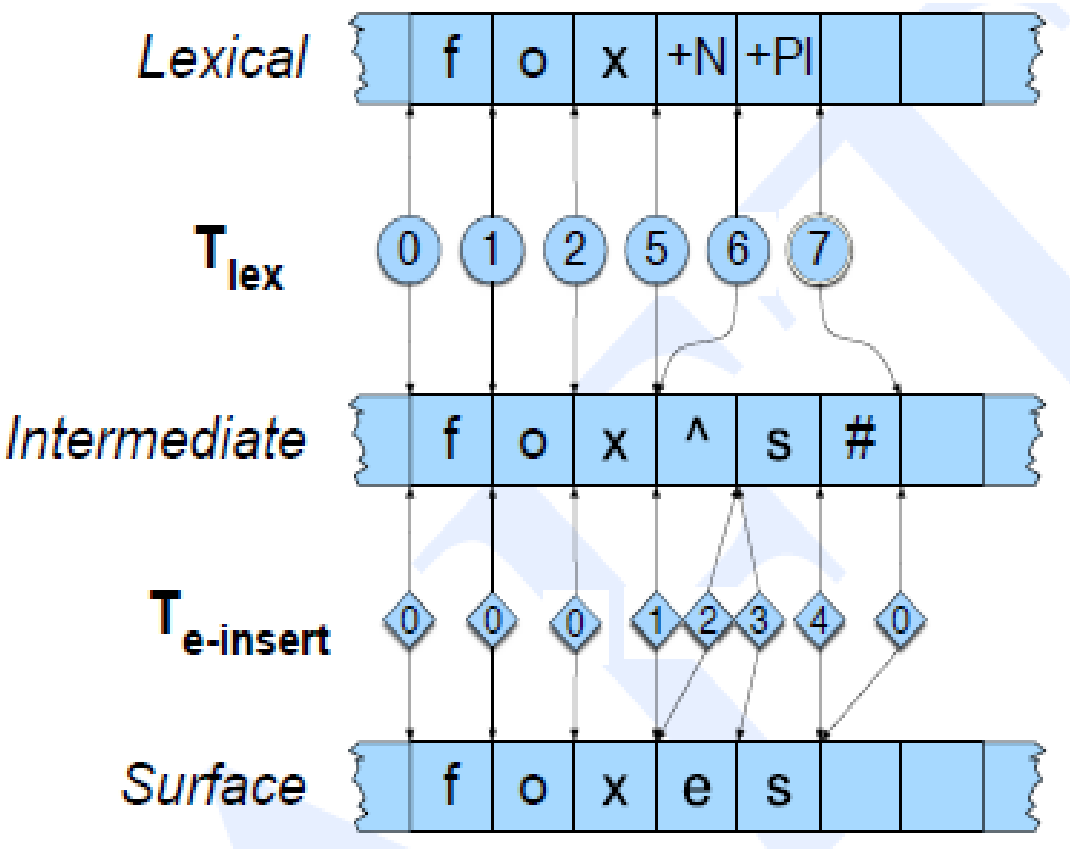
***(Cant decide on the correct parse by the transducer)***

- **Disambiguating:** require external evidence such as surrounding words

**Example:** ***"I saw two foxes yesterday".*** ***(Foxes will be noun)***

***"That trickster foxes me every time!"*** ***(Foxes will be verb)***

**Disambiguation algorithms are used to solve this. If not both the choices are considered. (Noun and verb)**



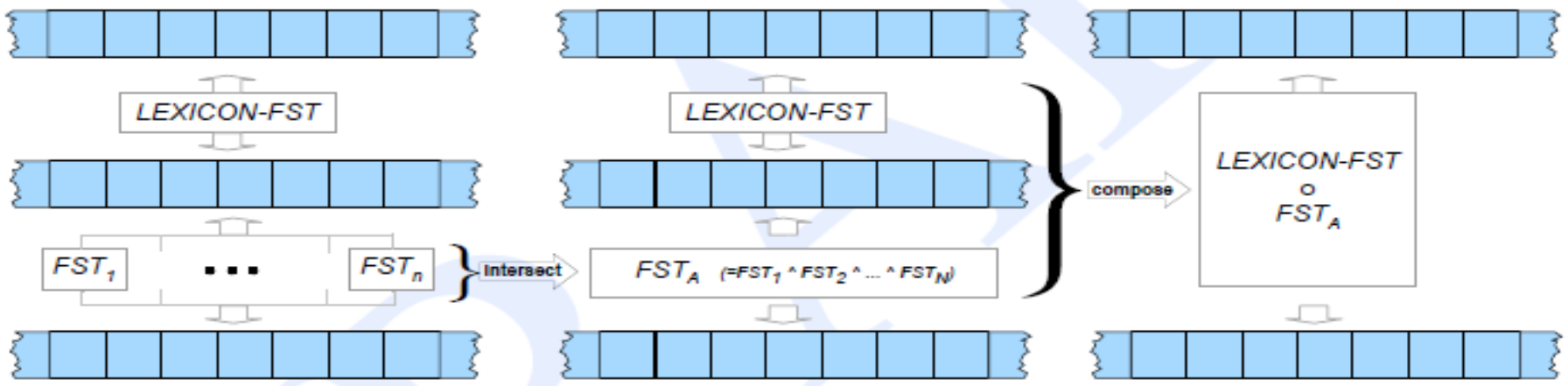

**Generating or parsing with FST lexicon and rules**

# Combining FST Lexicon and Rules

- **Local ambiguity**: occurs during the process of parsing
- **Example**:
  - Parsing the input verb *"assess"*
  - E-insertion transducer, after seeing the word *"ass"* may propose that, **e** that follows is inserted by spelling rule. Hence, the word *"asses"* would be parsed instead of *"assess"*
  - But **#** symbol is not present after *"asses"* instead another ***s*** is seen. Then its realized that wrong word is parsed
- Solution: FST-parsing algorithm need to incorporate search algorithms.

# Combining FST Lexicon and Rules

- ***Composing a cascade of transducers in series into a single complex transducer***
- Transducers running in parallel can be combined by **automaton intersection**
- **Automaton intersection algorithm takes 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 $q_{nm}$.




**Intersection and composition of transducers**

# Lexicon-Free FSTs: The Porter Stemmer

- **Information Retrieval (IR) tasks** like web search - a query such as a Boolean combination of relevant **keywords or phrases**, e.g., ***(kangaroo)*** returns documents that have these words in them.
- A document with the word ***kangaroos*** might not match the keyword ***kangaroo.*** Hence, some IR systems first **run a stemmer on the query and document words.**
- Morphological information in IR is used to determine that two words have the **same stem**; the suffixes are thrown away.
- The most widely used such **stemming algorithm** is the ***Porter algorithm (1980) .***

# Human Morphological Processing

- Deals with how multi-morphemic words are represented in the minds of speakers of English.

| Word | Derived forms |
|---|---|
| Walk | Walks, walked |
| Happy | Happily, happiness |

Are all three forms in human lexicon?
or is it just ***walk*** plus ***- ed*** or ***walk*** plus ***s ?***

- **Full listing hypothesis** - proposes that all words of a language are listed in mental lexicon without any internal morphological structure
  - Hence, walk, walks, walked etc. are stored separately listed in the lexicon
  - But this is not true for complex languages (Turkish)
- **The minimal redundancy hypothesis** - suggests that only the constituent morphemes are represented in lexicon
  - Hence, when processing **walks**, both morphemes must be accessed (i.e. **walk** and **-s**) and combine them

# Human Morphological Processing

- Modern experimental evidence suggests that both the theories (full listing and minimum redundancy) are not completely true
- Instead it states that some kinds of morphological relationships are mentally represented for some words (inflections and certain derivations) and for others, words are being fully listed.
  - **Example**: derived forms **happiness**, **happily** are stored separately from their stem **happy**
- But regularly inflected forms (**pouring**) are distinct in the lexicon from their stems (**pour**). This is done by using a **repetition priming experiment**.
  - *Repetition priming* – word is recognized faster if it has been seen before (**primed**)
- ***Marslen-Wilson (1994)*** found that spoken words can prime their stems, but only if the meaning of the derived form is closely related to the stem
  - **Example:** ***Government*** primes ***govern***, but ***department*** does not prime ***depart***

# Human Morphological Processing

***Speech errors (slips of the tongue)***

- In normal conversation, speakers often mix up the order of the words or initial sounds:

  ***Example:***

  - ***If you break it it'll drop***
  - ***I don't have time to work to watch television because I have to work***

- Inflectional and derivational affixes can appear separately from their stems:

  ***Example:***

  - ***Easily enoughly (for "easily enough")***
  - ***Words of rule formation (for "rules of word formation")***

  This shows that mental lexicon must contain some representation of the morphological structure of these words

  **Hence, Morphology play a role in the human lexicon**

# Summary

1. Morphological Parsing
2. Affixes: prefixes, suffixes
3. Survey of English Languages
4. Inflectional and Derivational morphology
5. Finite-state Morphological parsing
6. Lexicon and Morphotactics
7. Finite-state Transducers (FST) – FST Morphological parsing
8. Spelling rules (Orthographic rules) and Finite-state Transducers
9. Combining FST lexicon and spelling rules (composing and intersecting)
10. Lexicon-Free FSTs - (Porter stemmer algorithm)
11. Human Morphology processing

# END