# Morphological Parsing

# Morphological Parsing

- It is used to identify and find the number of morphemes in a given word.

- Morphemes are the smallest indivisible meaningful units of a language which builds a word.

- Example

Mangoes

Mango (Stem) + es (Suffix)

Unhappy

Un (prefix) + Happy (stem)

Word (Two types of morphemes)

Stem (root word)

Affixes (prefix, Suffix, Infix)

- Steps to design Morphological parser
  - ✓ Lexicon
  - ✓ Morphotactic
  - ✓ Orthographic Rules.

# Morphological Parsing

## Lexicon

- ✓ Stores basic information about a word.
- ✓ Word is stem or affix
- ✓ If Stem, then whether a verb stem or noun stem.
- ✓ If affix, then whether a prefix, infix or suffix

## Morphotactic

- ✓ Set of rules to make a decisions
- ✓ Decides a word appear/not appear before, after or in between other words.
- ✓ For example

Use able ness

Useableness (Valid rule)

Able use ness

Ableuseness (Invalid rule)

## Orthographic rules

Set of rules used to decide spelling changes.

For example

Baby + S =Babys

Baby+s = Babies

# Morphological Analysis

- Morphology means study of word / making of word.
- Some words has their own meaning.
- Example

Camera Board Pen Table

- Some words are there which when divided into different words, those new words have their own meaning.

Example

Showcase

Show case

Cocktail

Cock tail

# Morphological Analysis

- Some words are there which does not have their own meaning but when they are combined with other words, they become meaningful.

Example:

ing

Killing Booking Running Playing

# Finite State Automata

- A finite state automata is defined as M = {Q,Σ, $\delta$, $q_0$, F}

Where

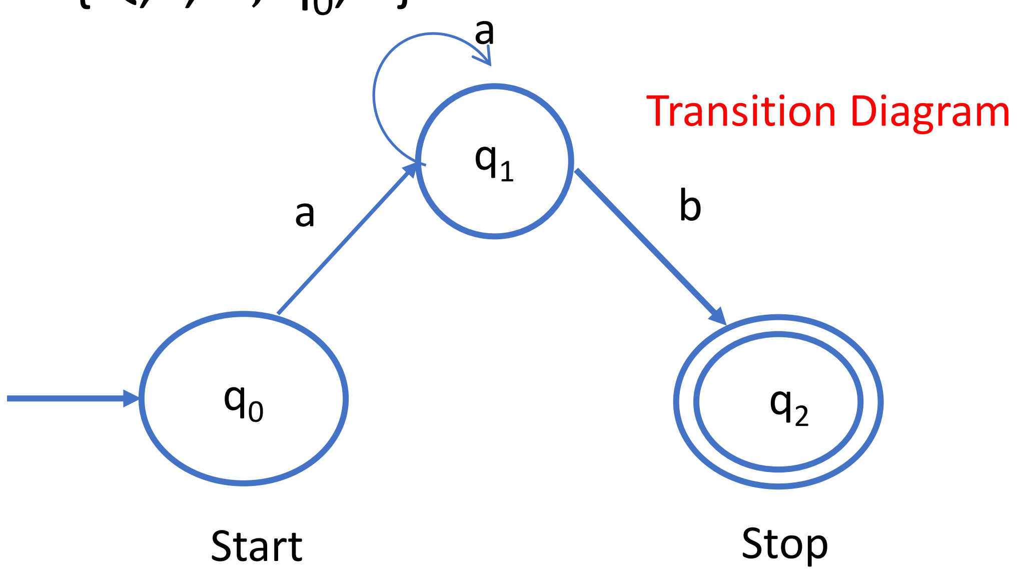
✓Q: Finite Non-empty Set of States

✓Σ : Finite Set of Input symbols

✓$\delta$ :Transition Mapping: $Q \times \Sigma \longrightarrow Q$

✓$q_0$: Initial State of the Finite Automata

✓F : Finite Set of Final States




Example = $a^+b$

$Q = \{q_0,q_1,q_2\}$
$\Sigma = \{a,b\}$

# Finite State Automata

Let's begin with the "sheep language" we discussed previously. Recall that we defined the sheep language as any string from the following (infinite) set:

baa!
baaa!
baaaa!
baaaaa!
...

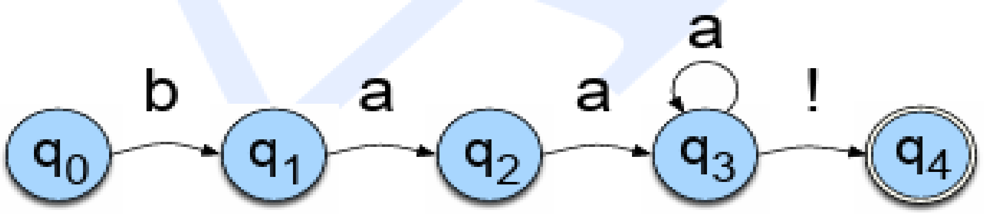



**Figure 2.10** A finite-state automaton for talking sheep.

# Algorithm for FSA

```
function D-RECOGNIZE(tape, machine) returns accept or reject

  index ← Beginning of tape
  current-state ← Initial state of machine
  loop
   if End of input has been reached then
     if current-state is an accept state then
        return accept
     else
        return reject
   elsif transition-table[current-state,tape[index]] is empty then
       return reject
   else
       current-state ← transition-table[current-state,tape[index]]
       index ← index + 1
  end
```

**Figure 2.12** An algorithm for deterministic recognition of FSAs. This algorithm returns *accept* if the entire string it is pointing at is in the language defined by the FSA, and *reject* if the string is not in the language.

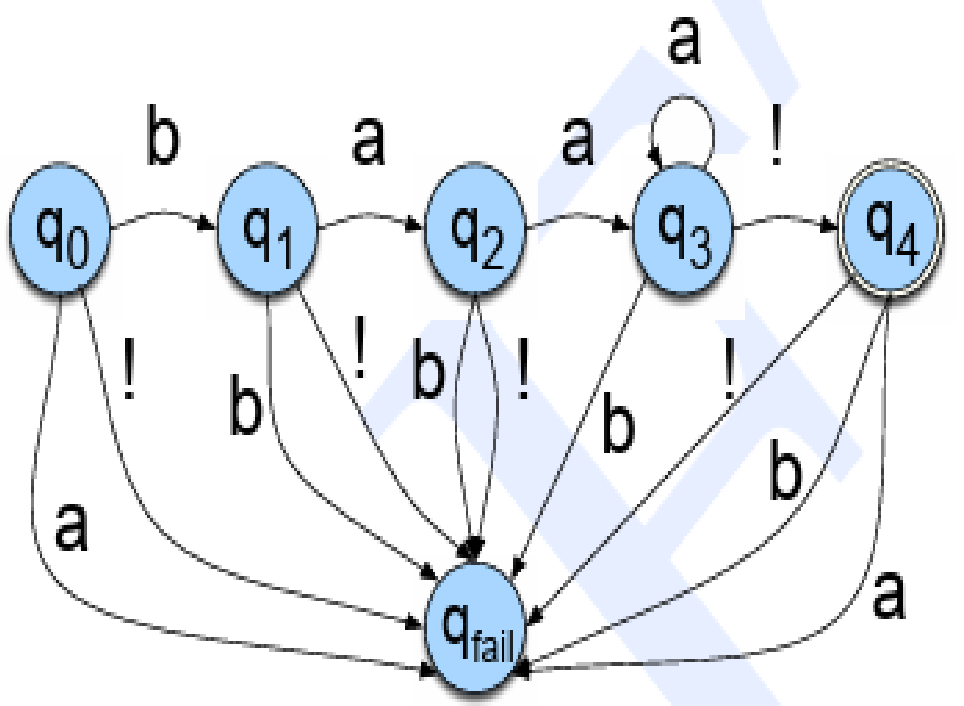
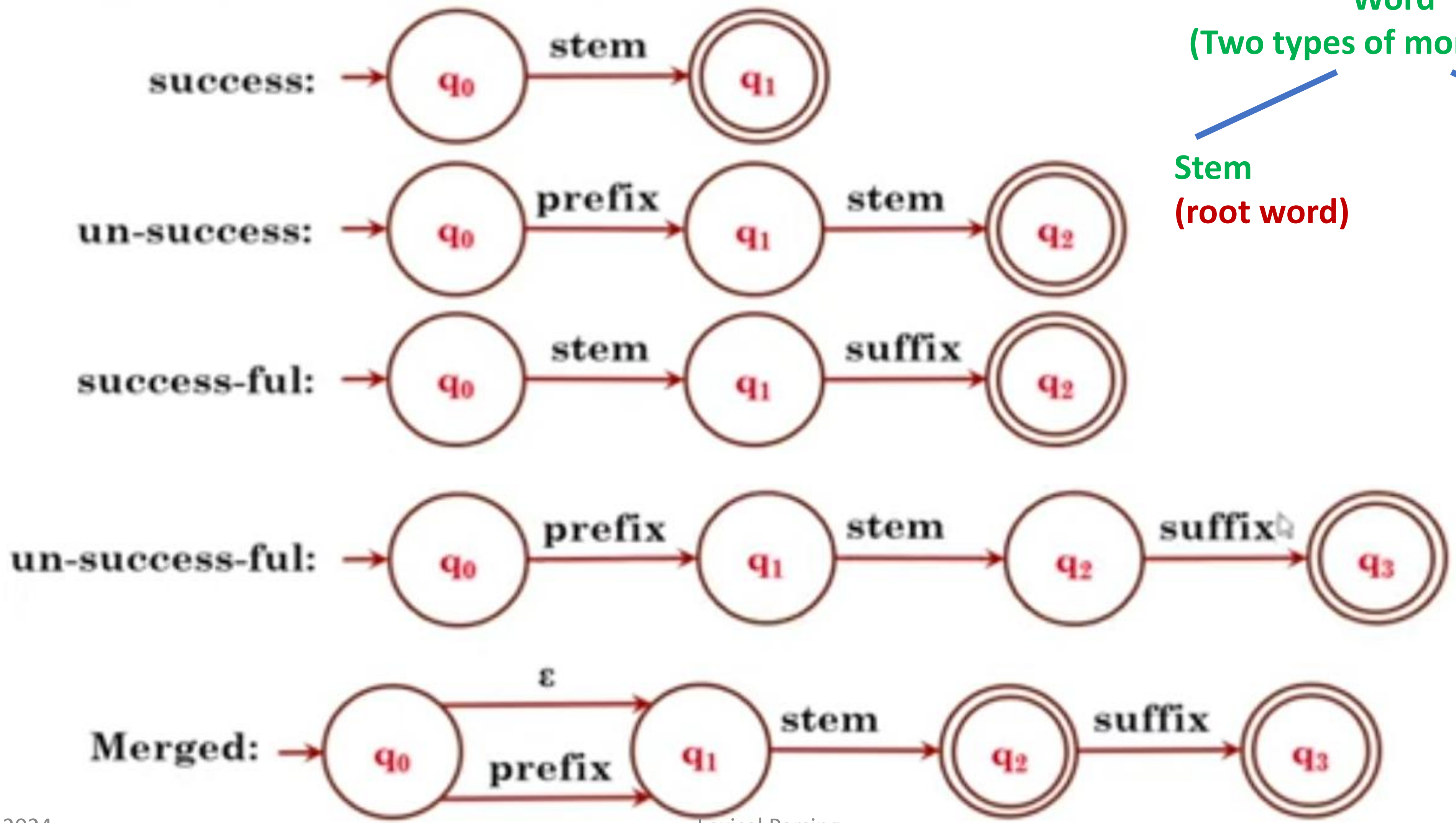



Figure 2.14 Adding a fail state to Fig. 2.10.

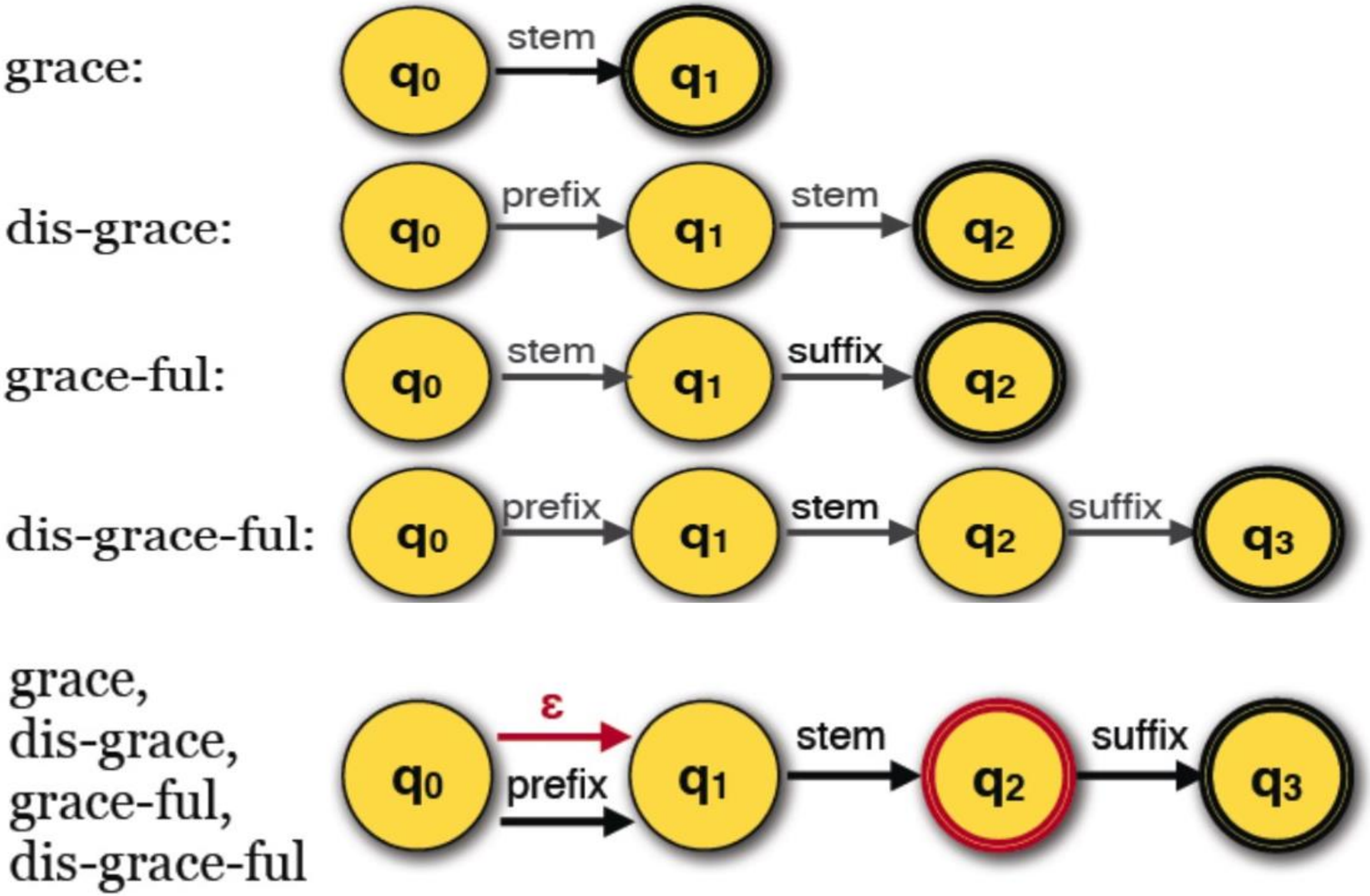
# Finite State Automata for Morphology




Word
(Two types of morphemes)

Stem
(root word)

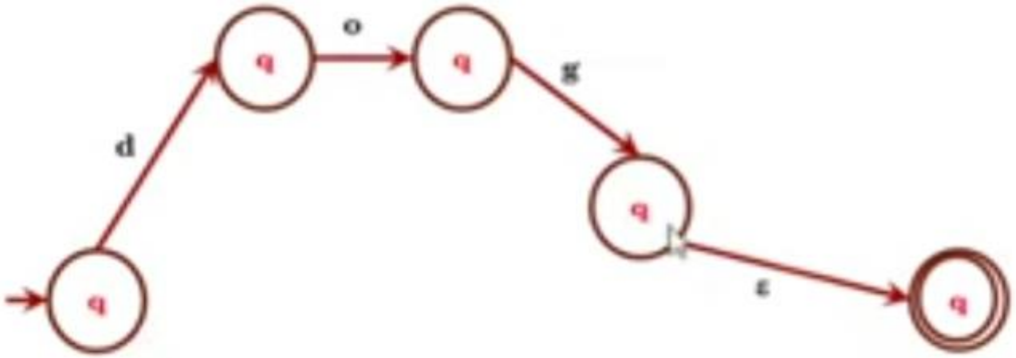
Affixes
(prefix,
Suffix,
Infix)

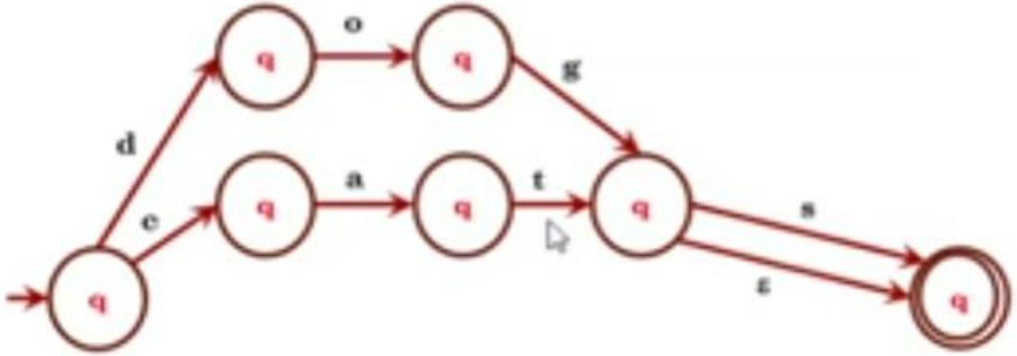

grace:
q0
stem
q1
dis-grace:
q0
prefix
q1
stem
q2
grace-ful:
q0
stem
q1
suffix
q2
dis-grace-ful:
q0
prefix
q1
stem
q2
suffix
q3
grace,
dis-grace,
grace-ful,
dis-grace-ful
q0
ε
prefix
q1
stem
q2
suffix
q3

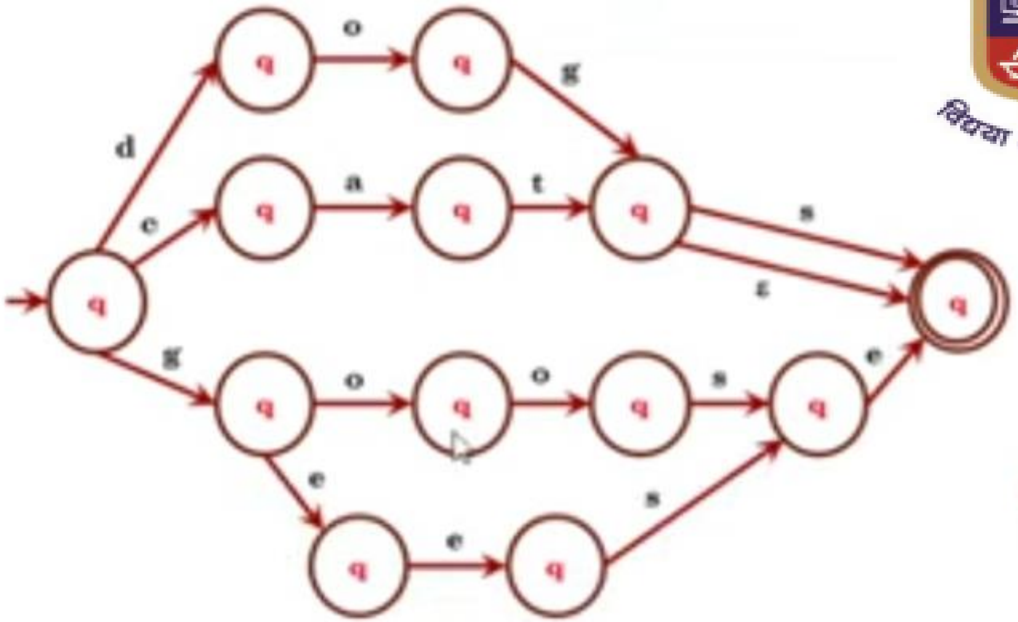
# STEM CHANGES

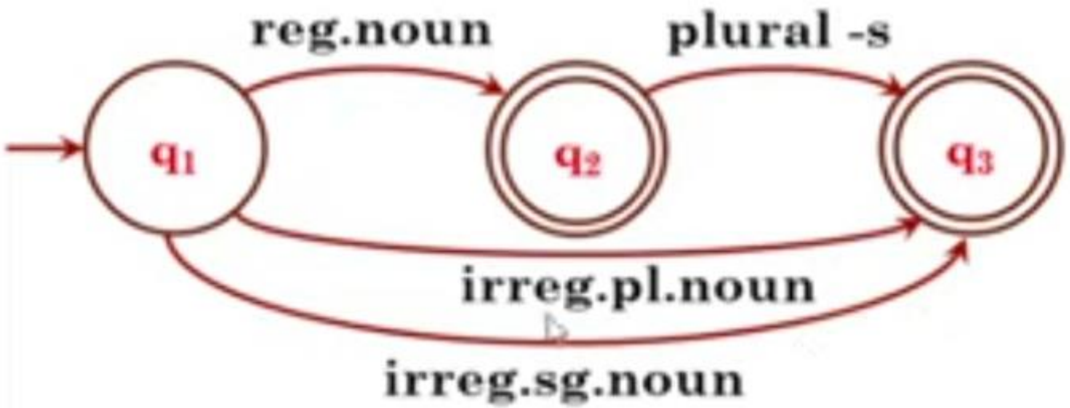
- Some irregular word requires stem changes.
- Example
  - ✓Goose -> geese
  - ✓Mouse ->Mice
  - ✓Teach -> Taught
  - ✓Go ->  went

| Reg-noun | Irreg-pl-noun | Irreg –sg -noun | plural |
|---|---|---|---|
| rat | geese | goose | -s |
| cat | taught | teach | |
| dog | mice | mouse | |


q
d
o
g
ε


q
d
o
g
c
a
t
s
ε


q
d
o
g
c
a
t
s
ε
o
o
s
e
e
e
s

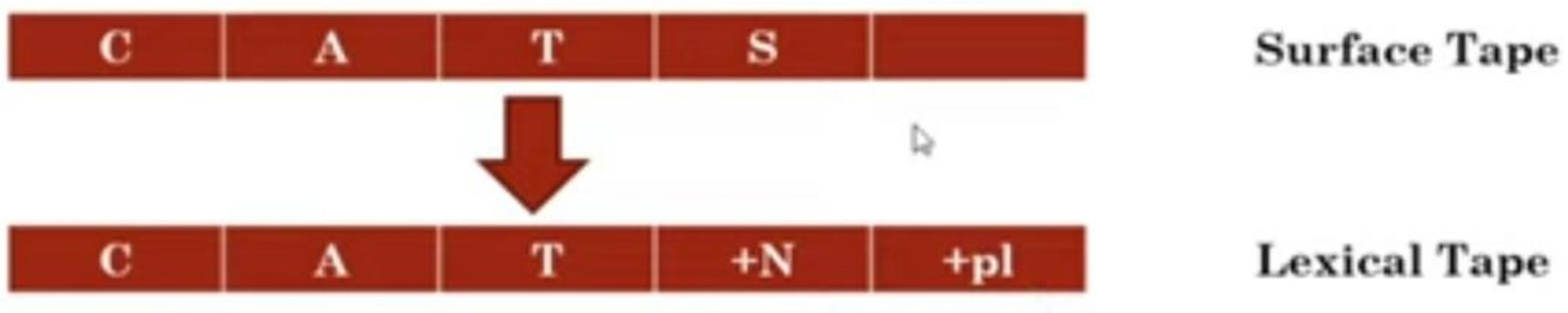

reg.noun
plural -s
$q_1$
$q_2$
$q_3$
irreg.pl.noun
irreg.sg.noun

# RECOGNITION vs ANALYSIS

- Finite State Automata can recognize/accept a string, but they cannot tell its internal structure.
- Thus, a machine is required to map/transduce the input string into an output string that encodes its internal structure.
- Finite State Transducers has two tapes for input and output as:

    Lexical Tape and Surface Tape.

Any one of the two tape can be either input tape or output tape.



# Finite State Transducer

- A formal language is not the natural language, but it can be used to model part of natural languages such as phonology, morphology, etc.
- FSTs are FSAs with two tapes.
- AFST is 7 tuple, T= $(Q, \Sigma, \Gamma, q0, F, \delta, \lambda)$ where
  - ✓ Q: Finite Set of States
  - ✓ $\Sigma$ : Finite set of Input Symbols
  - ✓ $\Gamma$ : Finite set of output symbols
  - ✓ $q_0$ : Initial State
  - ✓ F : Set of final states
  - ✓ $\delta$ : Transition Function Mapping $\delta$:Q x $\Sigma$ -> 2Q
  - ✓ $\lambda$ : Output Function Mapping $\lambda$ :Q x{$\Sigma$ U ε} -> Q x{$\Gamma$U ε}

# Example on FST

- $Q = \{q_0 , q_1\}$
- $\Sigma = \{a, b\}$
- $\Gamma = \{x, y, z\}$
- $\{\Sigma \cup \varepsilon\} = \{a, b, \varepsilon\}$
- $\{\Gamma \cup \varepsilon\} = \{x, y, z, \varepsilon\}$
- $\lambda$ :

$$\left\{ \begin{array}{l} \langle 0,a\rangle , \langle 0,b\rangle, \langle 0,c\rangle, \langle 0, \varepsilon \rangle \\ \langle 1,a\rangle , \langle 1,b\rangle, \langle 1,c\rangle, \langle 1, \varepsilon \rangle \end{array} \right\} \times \left\{ \begin{array}{l} \langle 0,x\rangle , \langle 0,y\rangle, \langle 0,z\rangle, \langle 0, \varepsilon \rangle \\ \langle 1,x\rangle , \langle 1,y\rangle, \langle 1,z\rangle, \langle 1, \varepsilon \rangle \end{array} \right\}$$

# THANK YOU