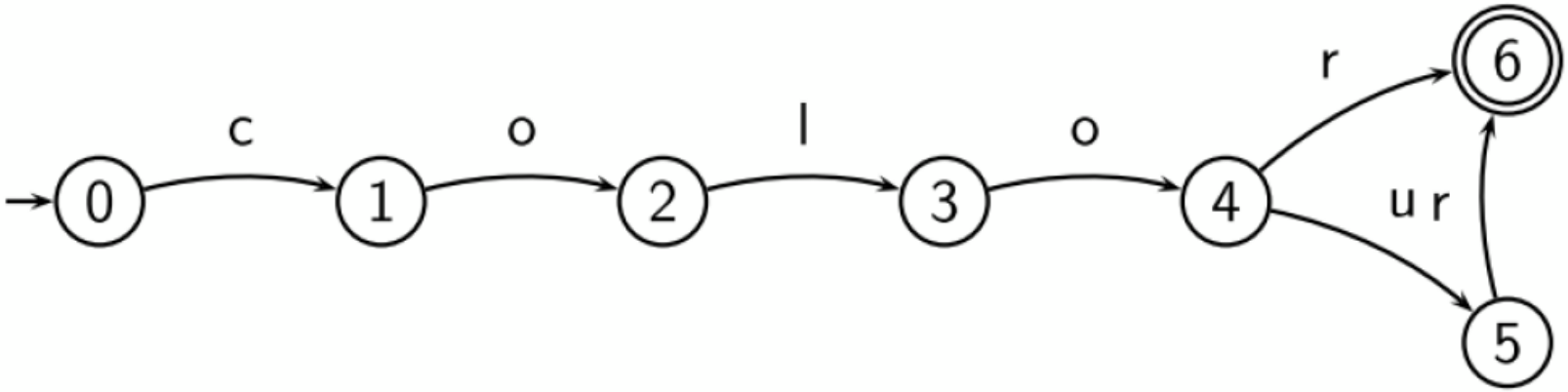
Finite-state methods for morphology

Pawan Goyal

CSE, IITKGP

Week 3: Lecture 3

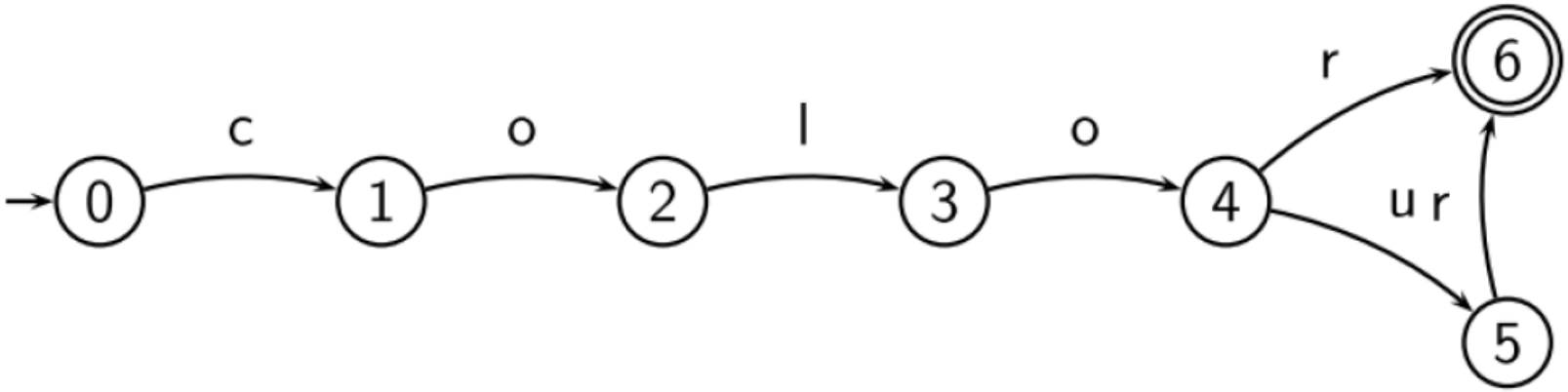
Finite State Automaton (FSA)



What is FSA?

- A kind of directed graph

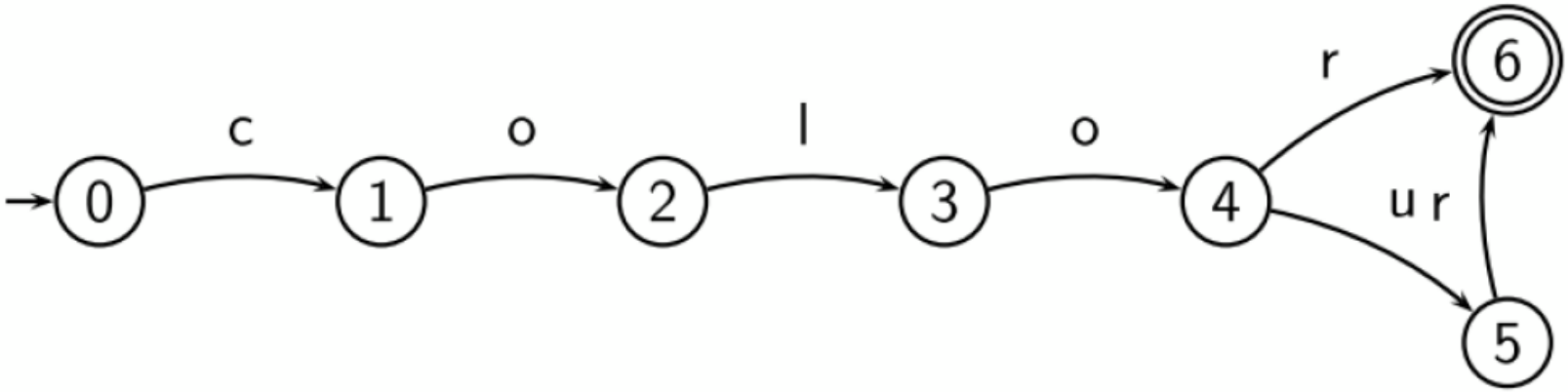
Finite State Automaton (FSA)



What is FSA?

- A kind of directed graph
- Nodes are called states, edges are labeled with symbols (possibly empty ϵ)

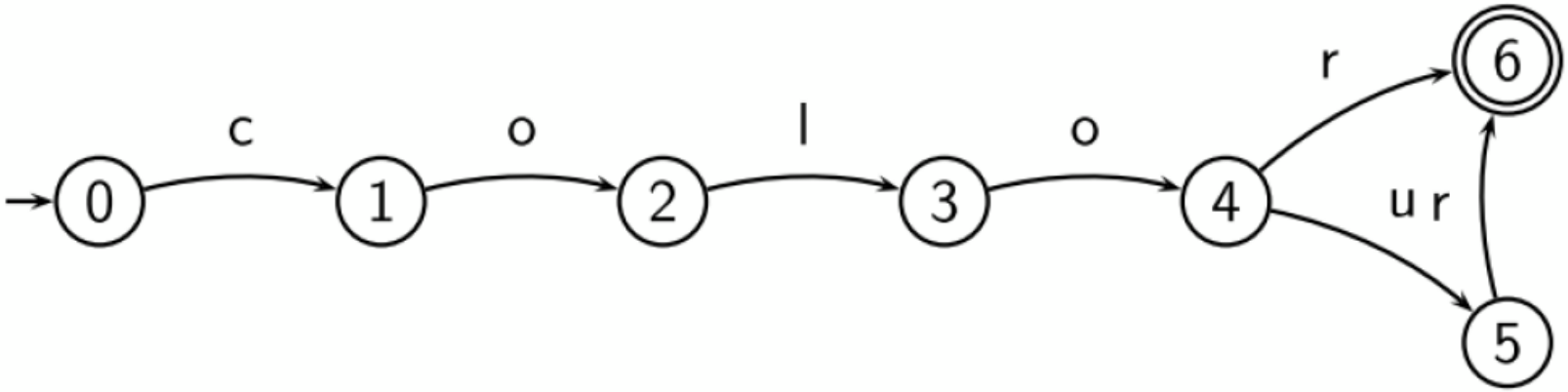
Finite State Automaton (FSA)



What is FSA?

- A kind of directed graph
- Nodes are called states, edges are labeled with symbols (possibly empty ϵ)
- Start state and accepting states

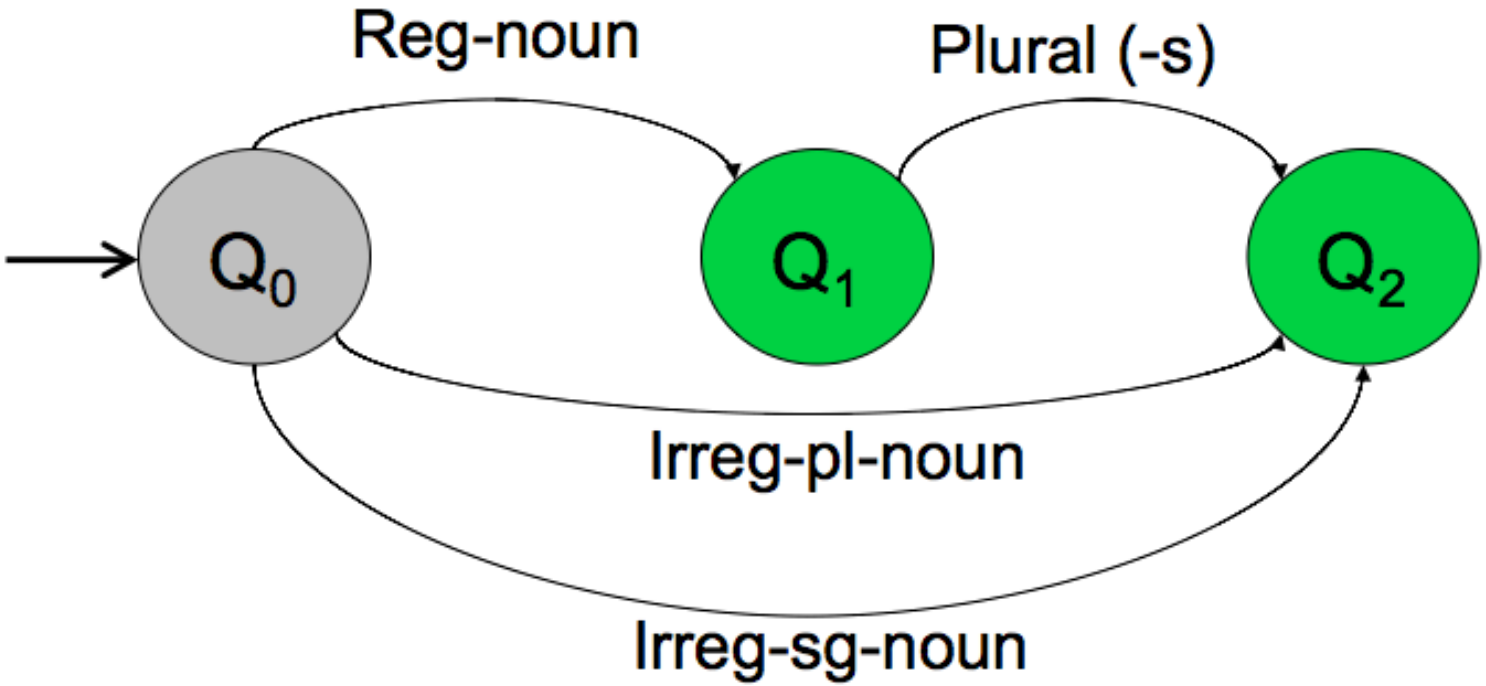
Finite State Automaton (FSA)



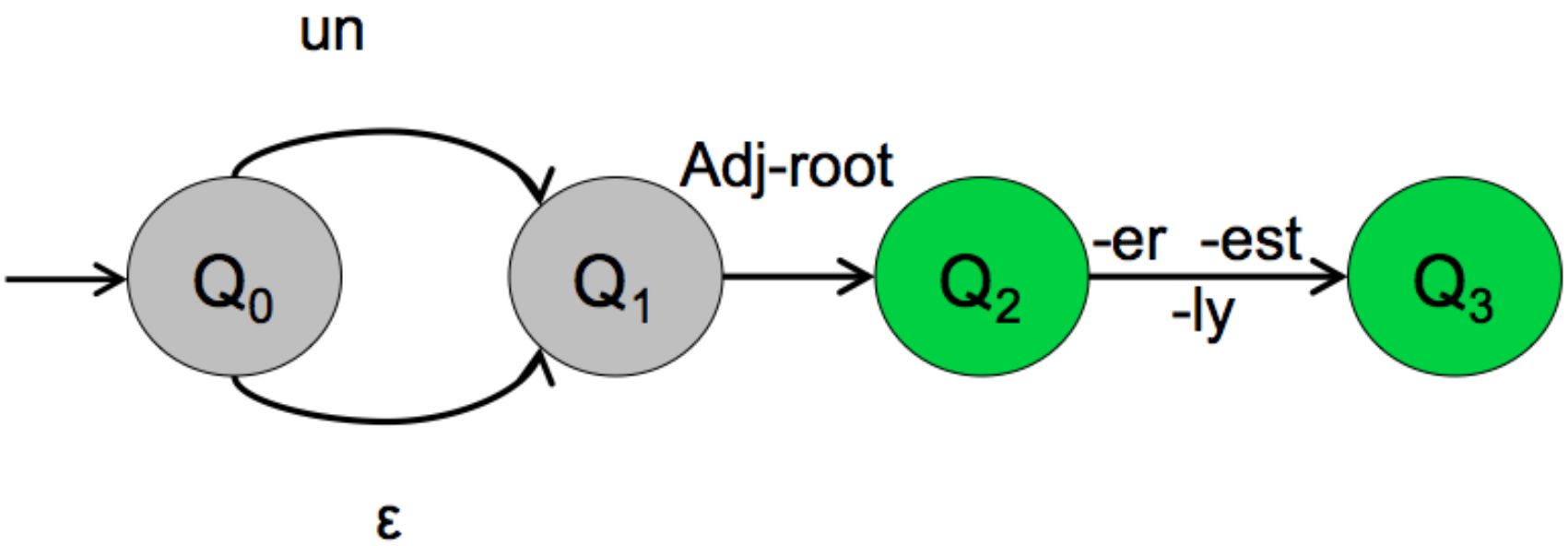
What is FSA?

- A kind of directed graph
- Nodes are called states, edges are labeled with symbols (possibly empty ϵ)
- Start state and accepting states
- Recognizes regular languages, i.e., languages specified by regular expressions

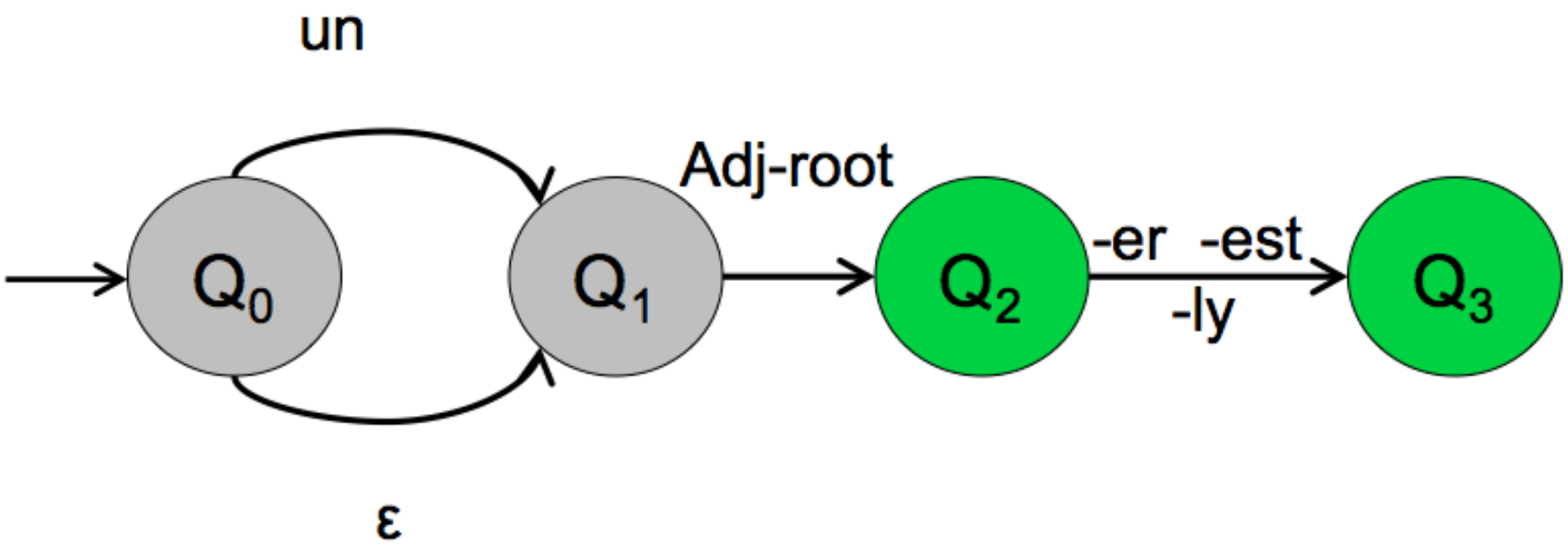
FSA for nominal inflection in English



FSA for English Adjectives



FSA for English Adjectives



Word modeled

happy, happier, happiest, real, unreal, cool, coolly, clear, clearly, unclear, unclearly, ...

Morphotactics

- The last two examples model some parts of the English morphotactics
- But what about the information about regular and irregular roots?

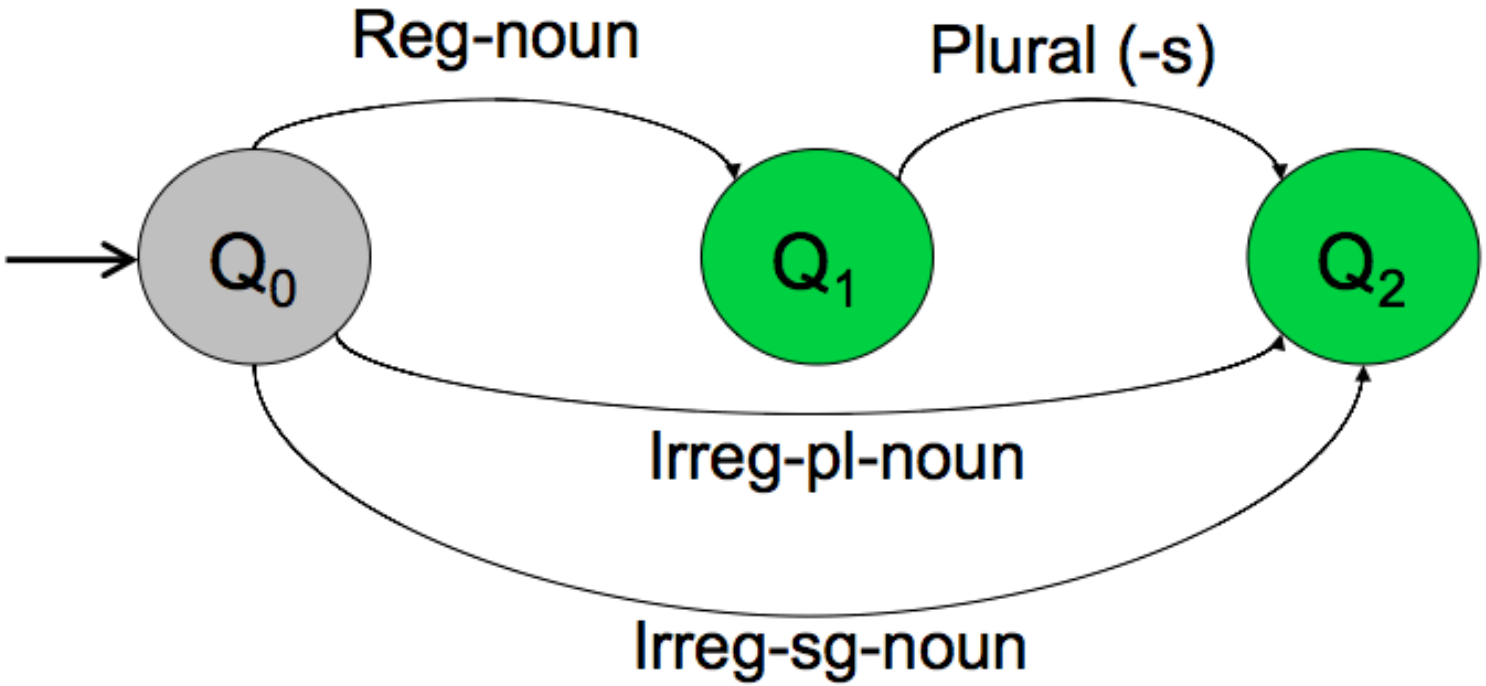
Morphotactics

- The last two examples model some parts of the English morphotactics
- But what about the information about regular and irregular roots?

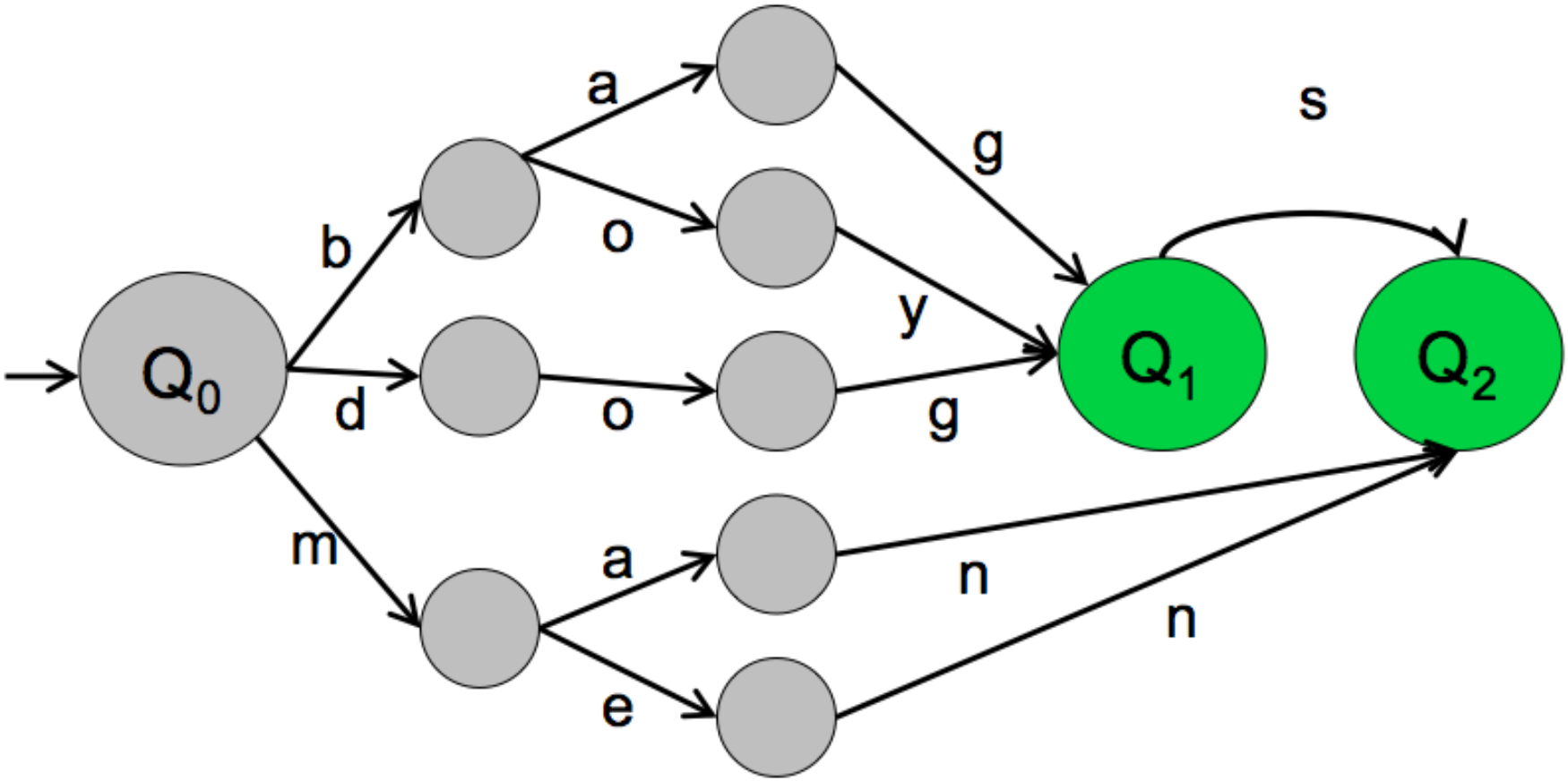
Lexicon

Can we include the lexicon in the FSA?

FSA for nominal inflection in English



After adding a mini-lexicon



Some properties of FSAs: Elegance

- Recognizing problem can be solved in linear time (independent of the size of the automaton)
- There is an algorithm to transform each automaton into a unique equivalent automaton with the least number of states
- An FSA is deterministic iff it has no empty (ϵ) transition and for each state and each symbol, there is at most one applicable transition
- Every non-deterministic automaton can be transformed into a deterministic one

But ...

FSAs are language recognizers/generators.

But ...

FSAs are language recognizers/generators.

We need transducers to build Morphological Analyzers

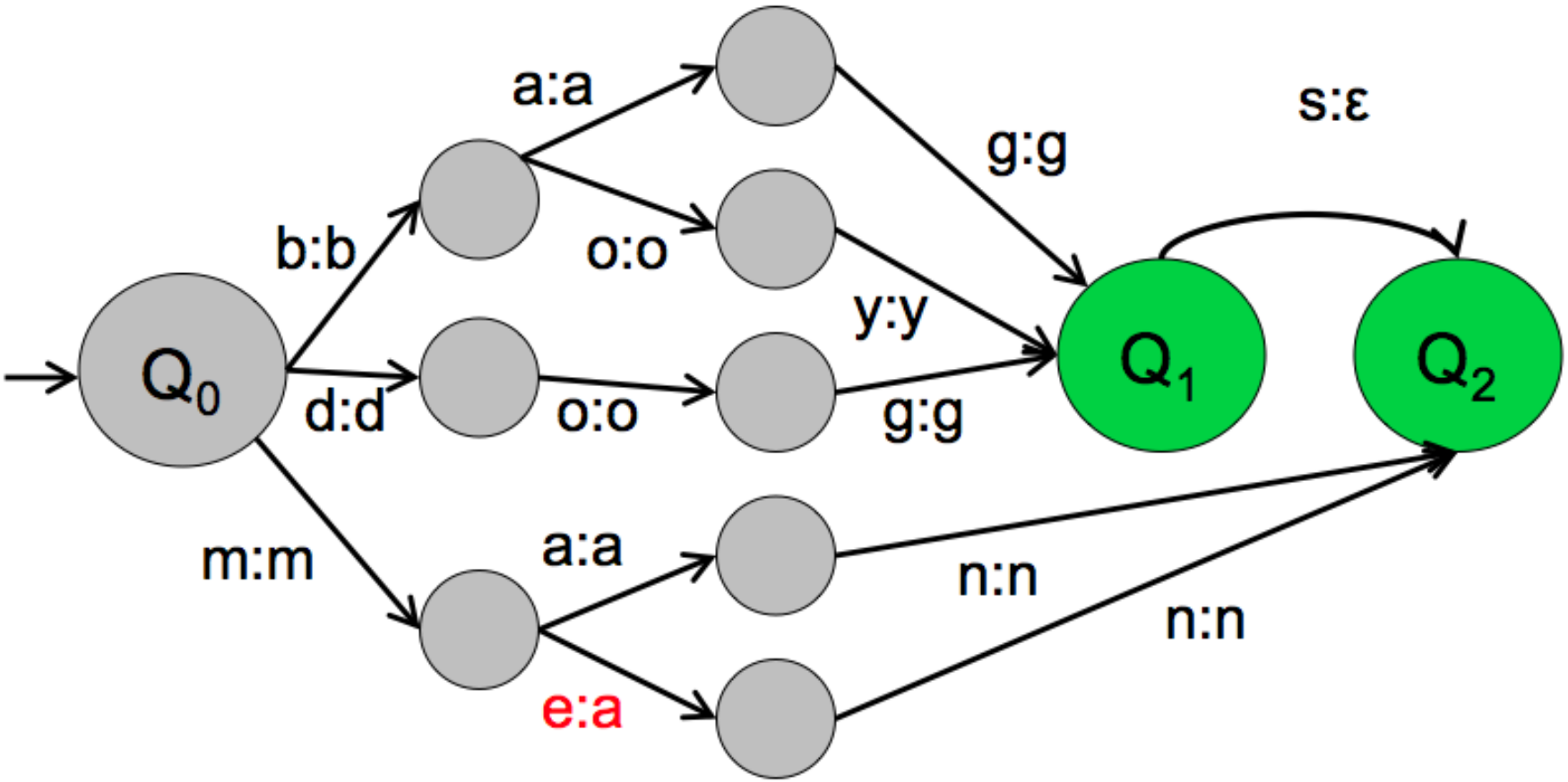
But ...

FSAs are language recognizers/generators.
We need transducers to build Morphological Analyzers

Finite State Transducers

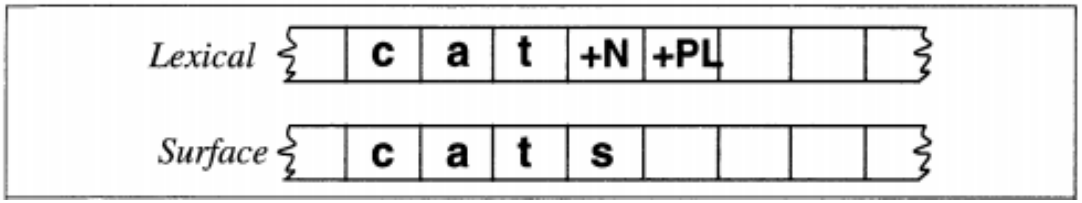
- Translate strings from one language to strings in another language
- Like FSA, but each edge is associated with two strings

An example FST

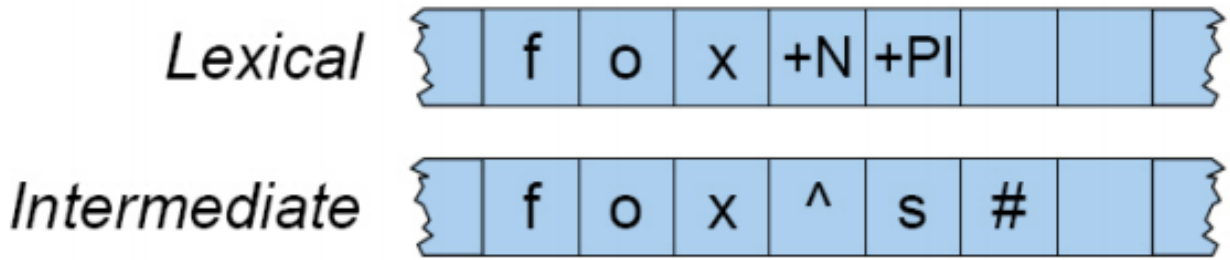


Two-level morphology

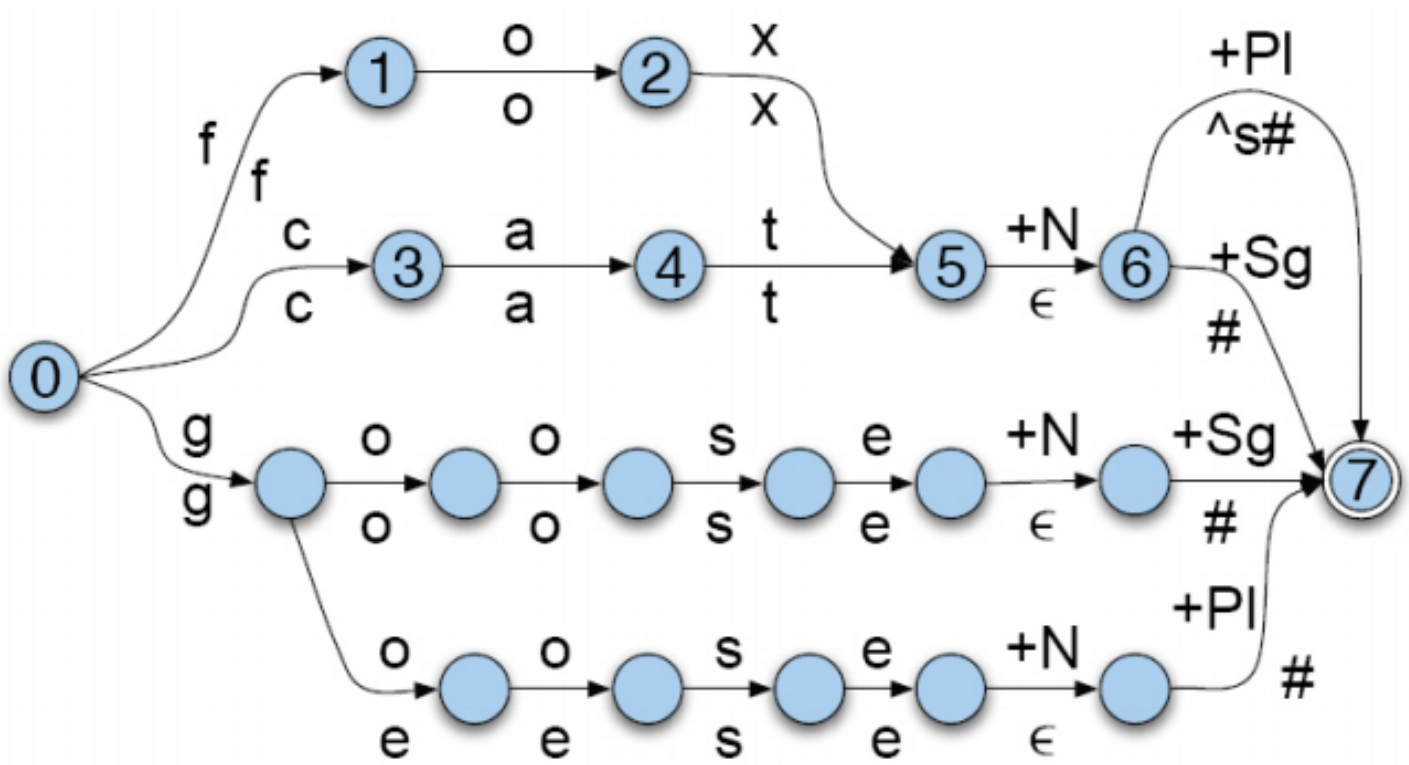
Given the input *cats*, we would like to output *cat+N+PL*, talling us that cat is a plural noun.
We do this via a version of **two-level morphology**, a correspondence between a lexical level (morphemes and features) to a surface level (actual spelling).



Intermediate tape for Spelling change rules

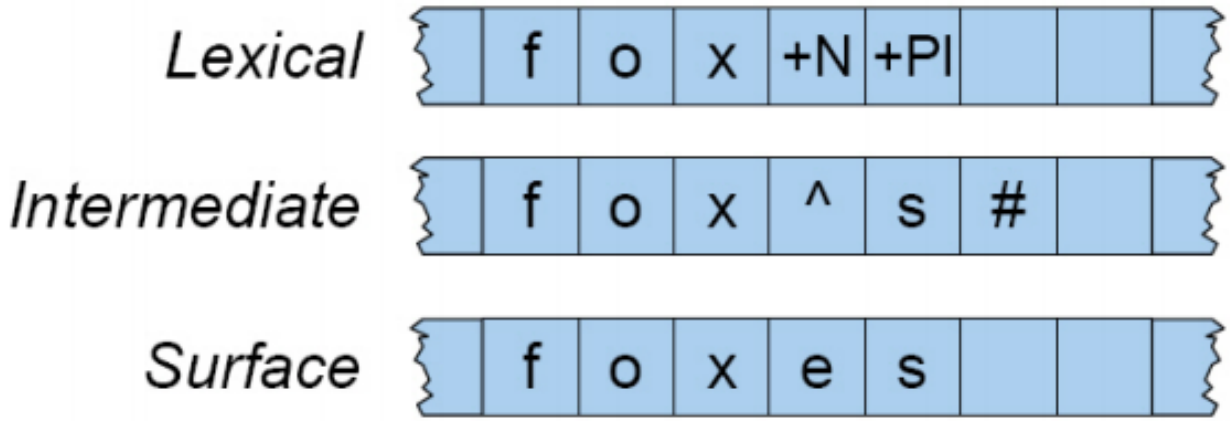


English Nominal Inflection FST



Spelling Handling

A spelling change rule would insert an e only in the appropriate environment.



Rule Handling

Rule Notation

$a \rightarrow b / c\_d$: "rewrite a as b when it occurs between c and d."

Morphological Analysis: Approaches

Two different ways to address phonological/graphemic variations

- Linguistic approach: A phonological component accompanying the simple concatenative process of attaching an ending
- Engineering approach: Phonological changes and irregularities are factored into endings and a higher number of paradigms

Different Approaches: Example from Czech

	woman	owl	draft	iceberg	vapor	fly
S1	žen-a	sov-a	skic-a	kr-a	pár-a	mouch-a
S2	žen-y	sov-y	skic-i	kr-y	pár-y	mouch-y
S3	žen-ě	sov-ě	skic-e	kř-e	pář-e	mouš-e
:						
P2	žen-0	sov-0	skic-0	ker-0	par-0	much-0

A linguistic approach

$$\textit{žen} + \begin{cases} a \\ y \\ ě \\ 0 \end{cases} \quad \textit{sov} + \begin{cases} a \\ y \\ ě \\ 0 \end{cases} \quad \textit{skic} + \begin{cases} a \\ y \\ ě \\ 0 \end{cases} \quad \textit{kr} + \begin{cases} a \\ y \\ ě \\ 0 \end{cases} \quad \textit{pár} + \begin{cases} a \\ y \\ ě \\ 0 \end{cases} \quad \textit{mouch} + \begin{cases} a \\ y \\ ě \\ 0 \end{cases}$$

An engineering approach

$$\textit{žen} + \begin{cases} a \\ y \\ ě \\ 0 \end{cases} \quad \textit{sov} + \begin{cases} a \\ y \\ ě \\ 0 \end{cases} \quad \textit{skic} + \begin{cases} a \\ i \\ e \\ 0 \end{cases} \quad k + \begin{cases} ra \\ ry \\ ře \\ er \end{cases} \quad p + \begin{cases} ára \\ áry \\ áře \\ ar \end{cases} \quad m + \begin{cases} oucha \\ ouchy \\ ouše \\ uch \end{cases}$$

Tools Available

- AT&T FSM Library and Lextools
  `http://www2.research.att.com/~fsmtools/fsm/`
- OpenFST (Google and NYU)
  `http://www.openfst.org/`