Unit 2
Word and Morphology
Finite State Machine, Morphology, Word Construction

Natural Language Processing (NLP)

MDS 555

Objective

- Regular Expressions
- Formal Language Processing
  - Finite state machine
  - Finite state transducers
- Further Study
  - Chapter 2 , 3 of Text book

Regular Expressions (RE)

- First developed by Kleene (1956)
- language for specifying text search strings
- The Regular expression languages used for searching texts in UNIX (vi, Perl, Emacs, grep)
- RE features exist in the various Web search engines.

Regular Expressions (RE)

- A string is a sequence of symbols
  - for the purpose of most text based search techniques, a string is any sequence of alphanumeric characters (letters, numbers, spaces, tabs, and punctuation).
  - Regular expression search requires a pattern that we want to search for, and a corpus

Regular Expressions (RE)

- A string is a sequence of symbols
  - for the purpose of most text based search techniques, a string is any sequence of alphanumeric characters (letters, numbers, spaces, tabs, and punctuation).
  - Regular expression search requires a pattern that we want to search for, and a corpus

RE: Basic Patterns

- Regular expressions are case sensitive

RE	Example Patterns Matched
/woodchucks/	"interesting links to woodchucks and lemurs"
/a/	"Mary Ann stopped by Mona's"
/Claire␣says,/	" "Dagmar, my gift please," Claire says,"
/DOROTHY/	"SURRENDER DOROTHY"
/!/	"You've left the burglar behind again!" said Nori

- Disjunction

RE	Match	Example Patterns
/[wW]oodchuck/	Woodchuck or woodchuck	"Woodchuck"
/[abc]/	'a', 'b', *or* 'c'	"In uomini, in soldati"
/[1234567890]/	any digit	"plenty of 7 to 5"

RE: Basic Patterns

- RE: Basic Patterns

RE	Match	Example Patterns Matched
/[A-Z]/	an uppercase letter	"we should call it 'Drenched Blossoms'"
/[a-z]/	a lowercase letter	"my beans were impatient to be hoed!"
/[0-9]/	a single digit	"Chapter 1: Down the Rabbit Hole"

- caret ^ for negation

RE	Match (single characters)	Example Patterns Matched
[^A-Z]	not an uppercase letter	"Oyfn pripetchik"
[^Ss]	neither 'S' nor 's'	"I have no exquisite reason for't"
[^\.]	not a period	"our resident Djinn"
[e^]	either 'e' or '^'	"look up ^ now"
a^b	the pattern 'a^b'	"look up a^b now"

the period (/./), a wildcard expression

- One very important special character is the period (/./), a wildcard expression that matches any single character (except a carriage return):

RE	Match	Example Patterns
`/beg.n/`	any character between *beg* and *n*	begin, beg’n, begun

the period (/./), a wildcard expression

- One very important special character is the period (/./), a wildcard expression that matches any single character (except a carriage return):

RE	Match	Example Patterns
`/beg.n/`	any character between *beg* and *n*	begin, beg'n, begun

Finite State Machine

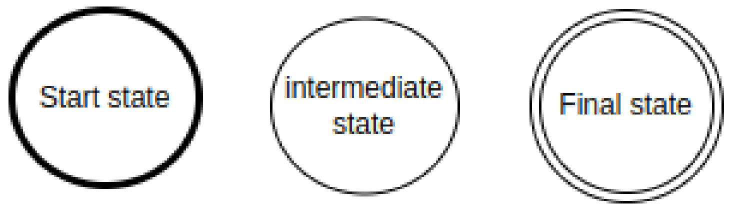
- Finite State Automata
- It is a computation model that can be implemented with hardware or software and can be used to simulate sequential logic and some computer programs.
- It has fixed set of possible states, a set of inputs that change the state and set of possible outputs.

Finite State Machine

- Finite state automata generate regular languages.
- Finite state machines can be used to model problems in many fields including
  - mathematics, artificial intelligence, games, and linguistics.

State transition diagram

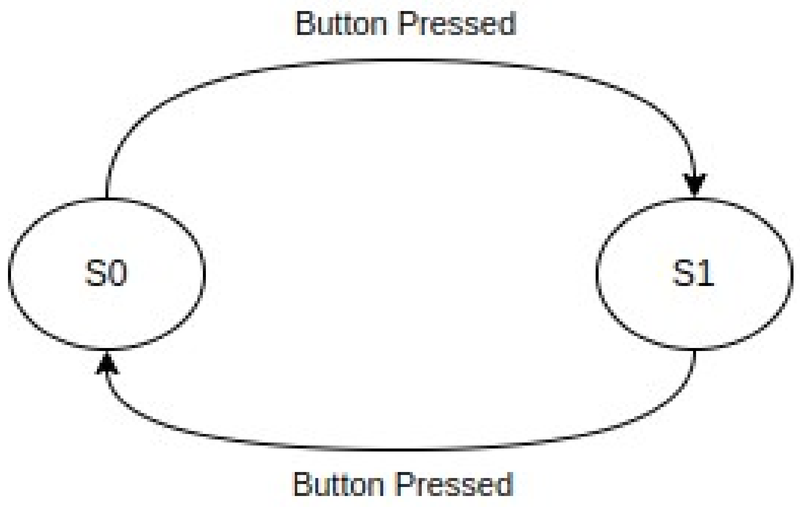
- States
  - Start State – Circle with bold border
  - State – intermediate states
  - Final State – double border circle
- Transition is shows by arrow



State transition diagram

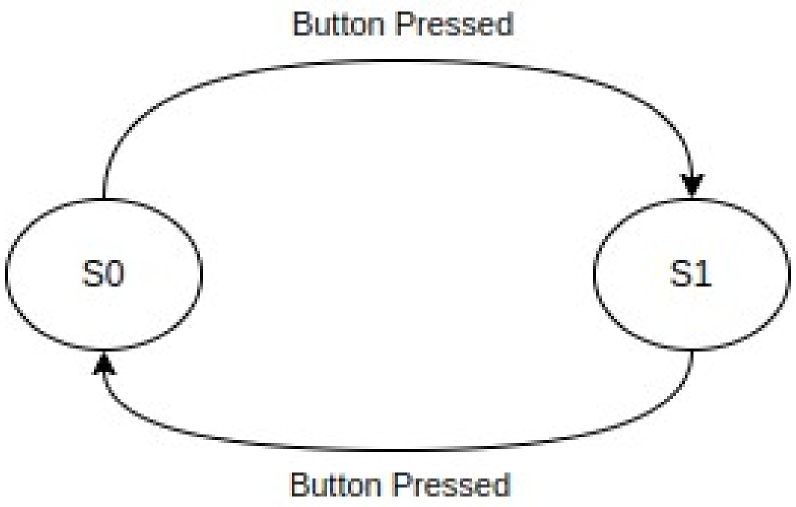
- Lets consider pen as a machine

  PEN: has push button on top and writing NIB on the bottom
  - S0 : NIB Retracted
  - S1 : NIB Extended



State transition table

- Table with
  - All possible input
  - Current State
  - Output or Next state after input is applied



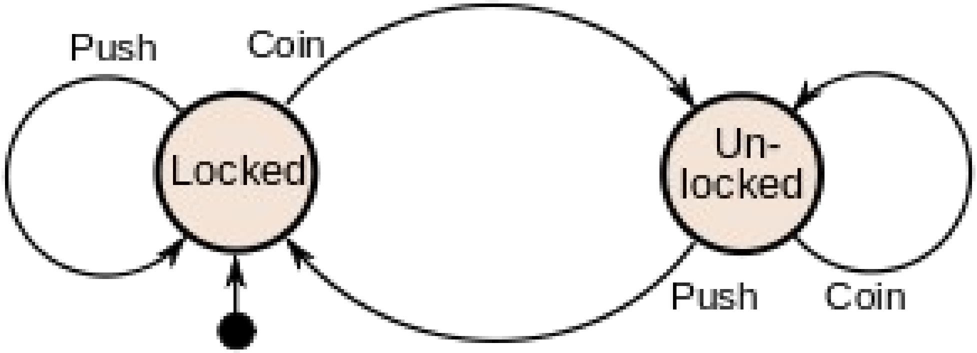
Input	Current State	Next State
Button pressed	NIB retracted	NIB extended
Button pressed	NIB extended	NIB retracted

FSM – Example (Turnstile)

- Inserting a coin into a turnstile will unlock it, and after the turnstile has been pushed, it locks again. Inserting a coin into an unlocked turnstile, or pushing against a locked turnstile will not change its state

FSM – Example (Turnstile)

- Inserting a coin into a turnstile will unlock it, and after the turnstile has been pushed, it locks again. Inserting a coin into an unlocked turnstile, or pushing against a locked turnstile will not change its state



Deterministic Finite State Machine (DFA):

- In a DFA, each state has a well-defined transition for every possible input.
- The transition from one state to another is uniquely determined by the current state and the input.
- DFAs are often used in scenarios where the system's behavior is straightforward and deterministic.

DFA : Formal definition

A deterministic finite automaton (DFA) is described by a five-element tuple: $(Q, \Sigma, \delta, q0, F)$

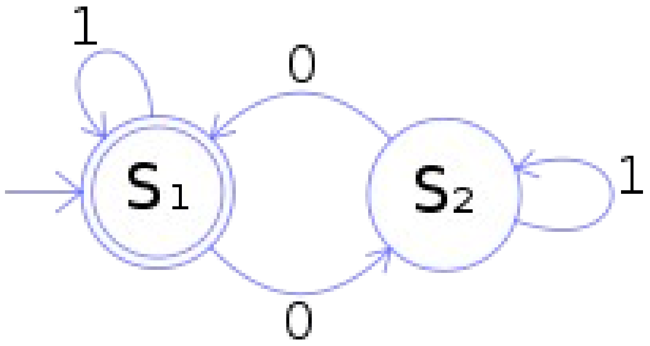
- Q = a finite set of states
- Σ = a finite, nonempty input alphabet
- δ = a series of transition functions
- $q0$ = the starting state
- F = the set of accepting states

There must be exactly one transition function for every input symbol in Σ from each state.

DFA

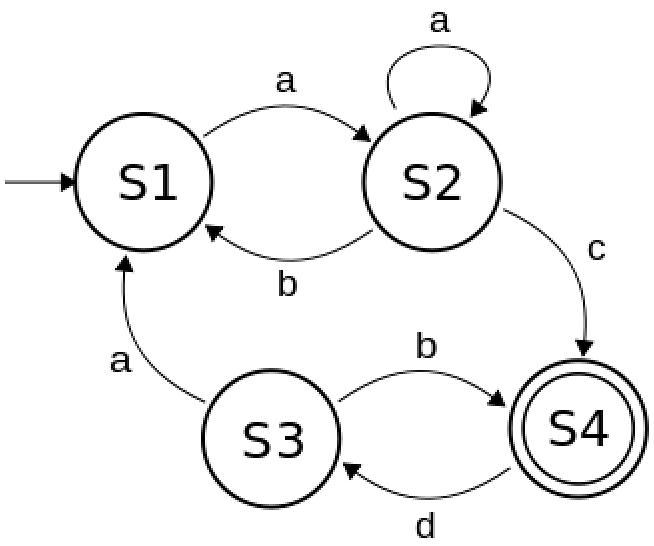
- $Q=\{s1,s2\}$
- $\Sigma=\{0,1\}$
- $q0=s1$
- $F=s1$
- The following table describes δ:

current state	input symbol	new state
s_1	1	s_1
s_1	0	s_2
s_2	1	s_2
s_2	0	s_1



DFA

- What string cannot be generated by the finite state machine below?
  - abacdaac
  - abac
  - aaaaac
  - aaaacd

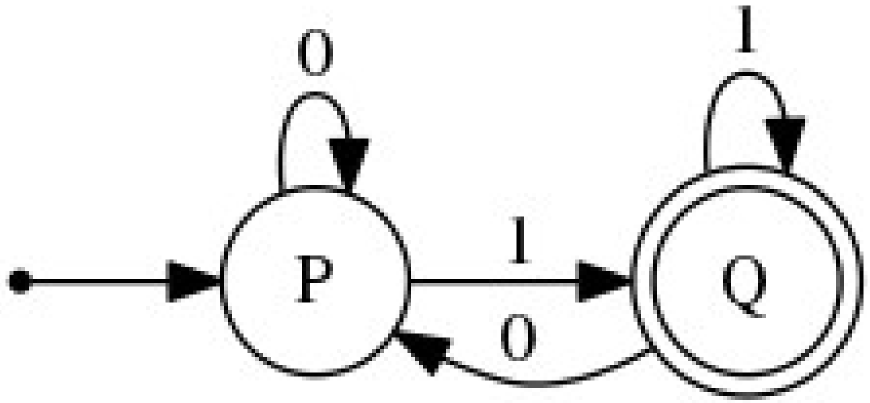


DFA

- Draw a diagram for a DFA that recognizes the following language:
  - The language of all strings that end with a 1.

DFA

- Draw a diagram for a DFA that recognizes the following language:
  - The language of all strings that end with a 1.



Non-Deterministic Finite State Machine (NFA)

- In an NFA, there can be multiple possible transitions for a given input in a given state.
- NFAs are used when the system's behavior is more complex and might have multiple valid paths.

NFA – Formal Defination

Similar to a DFA, a nondeterministic finite automaton (NDFA or NFA) is described by a five-element tuple: (Q, Σ, δ, q0, F)

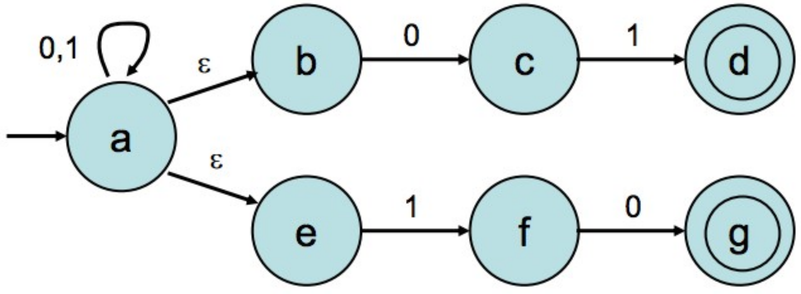
Q = a finite set of states

Σ = a finite, nonempty input alphabet

δ = a series of transition functions

q0 = the starting state

F = the set of accepting states

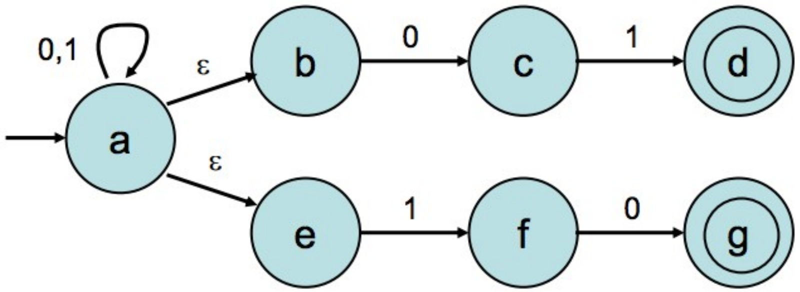


NFA – Formal Defination

- Unlike DFAs, NDFAs are not required to have transition functions for every symbol in Σ, and there can be multiple transition functions in the same state for the same symbol.
- Additionally, NDFAs can use null transitions, which are indicated by ϵ.
- Null transitions allow the machine to jump from one state to another without having to read a symbol.
- An NDFA accepts a string x if there exists a path that is compatible with that string that ends in an accept state.

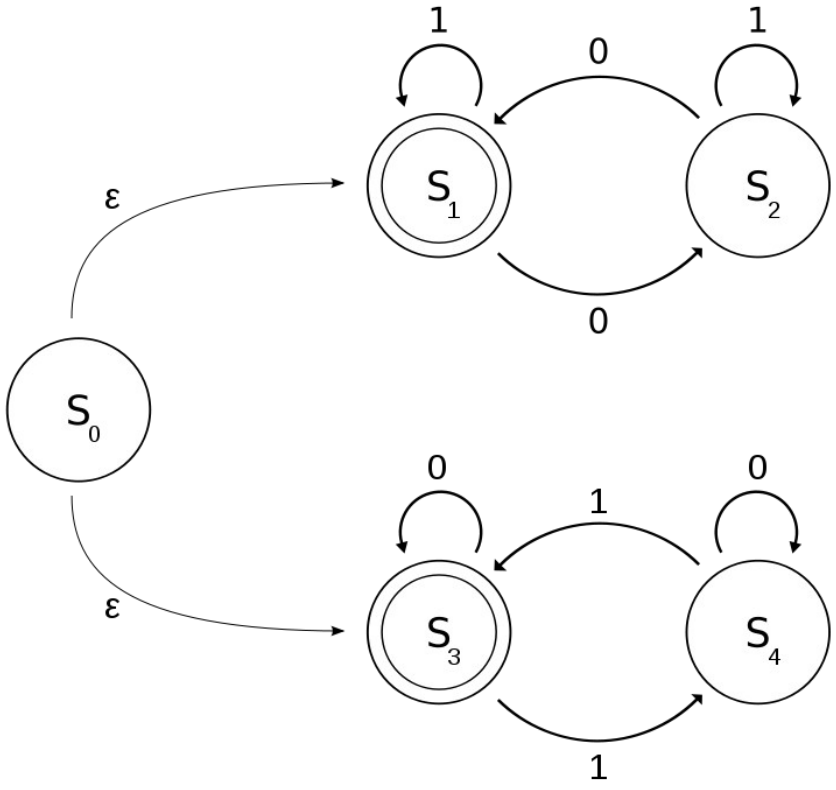
NDFA

- The NDFA that recognizes strings that end in “10” and strings that end in “01.”



NDFA

- Which string cannot be generated by the finite state machine below?
  - 1
  - 01001
  - 1011101
  - 1000
  - 0

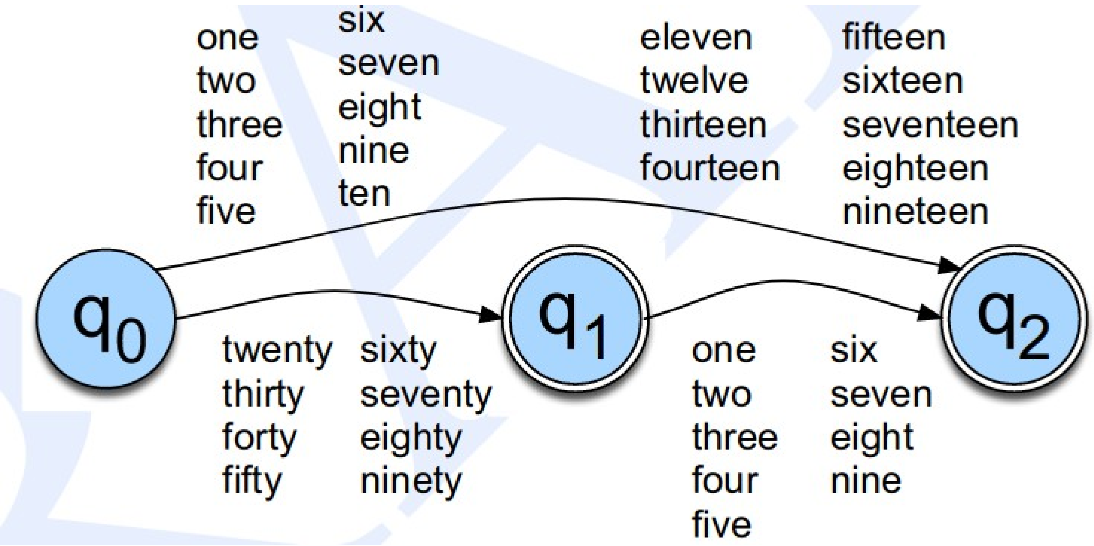


FSA - Example

- Make FSA for checking if given text represent english number or not?

FSA - Example

- Make FSA for checking if given text represent english number or not?



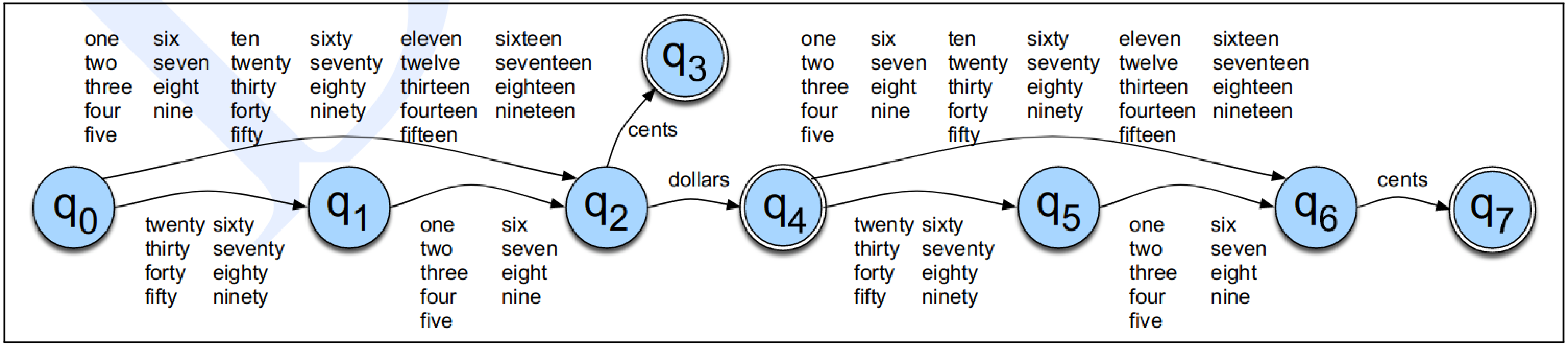
An FSA for the words for English numbers 1–99

FSA - Example

- Make FSA to check if “two dollars four cents” is valid language or not.

FSA - Example

- Make FSA to check if “two dollars four cents” is valid language or not.



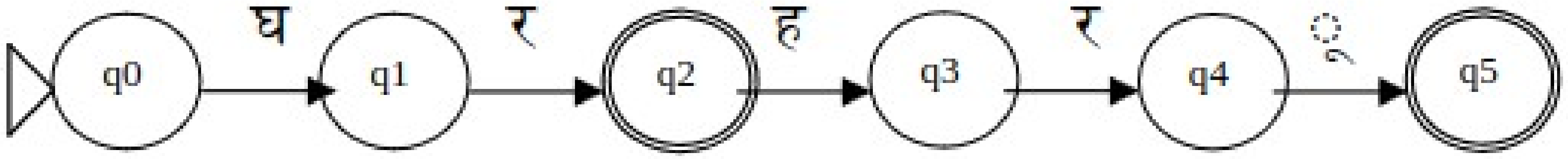
FSA for the simple dollars and cents

FSM – language processing

- Construct a FSM To validate the regular expression
  - Prefix/suffix detection
  - String end with “ing”

FSM – language processing

- For illustration, an automaton that accepts a string from the Nepali language घर ‘house’ and घरहरु ‘houses’ is visualized in Figure below



- This FSA accepts घर ‘house’ and घरहरु ‘houses’ because the inputs lead to final states. No other strings are accepted by this FSA.

NLP using Finite State Automata / Machine

- FSA/FSM, are useful in classical, rule-based NLP tasks that involve pattern recognition, regular languages, or finite context constraints.
- FSAs are not expressive enough for full natural language understanding,
- Still powerful tools in specific types of NLP tasks, particularly those that don't require deep hierarchical or long-distance dependencies.

FSA Use - Morphological Analysis

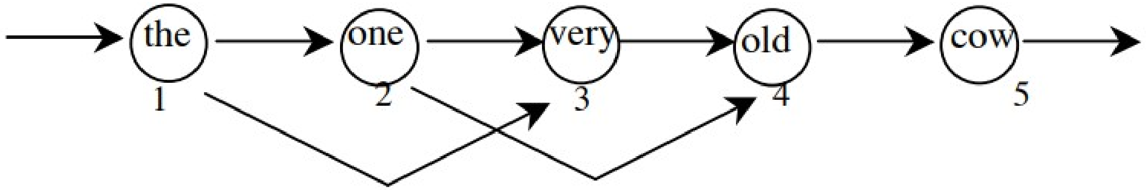
- Breaking words into morphemes (roots, suffixes, etc.).
- Finite State Transducers (FSTs, an extension of FSAs) are used for mapping surface forms to lemmas and morphological features.
- Example: walked → walk + PAST

Weighted Finite State Machines (WFSM)

- In many applications, we are not so interested in whether a sequence is a valid sentence in a language
- For instance, speech recognition engines typically use a model of word transition probabilities (e.g., how often a word w follows a word w') to drive the recognition process.
  - Adding such a model on top of straight acoustic recognition can double the word accuracy rate.
- We can model this data with a variant of an FSM called a weighted finite state machine (WFSM).
- A WFSM is just like an FSM expect that it has numbers on the transitions.

Weighted Finite State Machines (WFSM)

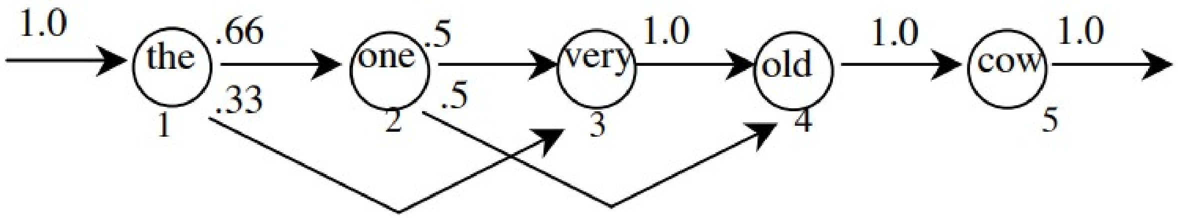
- We can generate
  - … the very old cow …..
  - … the one very old cow ….
  - … the one old cow ….



Which language is most likely?

Weighted Finite State Machines (WFSM)

- We can generate
  - … the very old cow …..
  - … the one very old cow ….
  - … the one old cow ….



P(The one old cow) = P(transition 1 to 2) * P(transition 2 to 3) * P(transition 3 to 5)

= 0.66 * 0.5 * 1.0 = 0.33

Markov Chain

- A deterministic probabilistic finite state machine, as previous one is, is often called a Markov Chain.
- It has the nice properties of DFSM in that we can very quickly determine whether a sequence is accepted by the machine.
- In addition, we can compute the probability of a sequence by simply multiplying the probabilities on each of the transitions together.

Thank you