Natural Language Processing

Regular Expressions and FSA

FSA is non-examinable.
For your information only.

Dr. Sun Aixin

FSA is non-examinable

Regular Expression and Finite state automata (FSA)

- Regular expressions
  - Regex is compact textual strings (e.g., "/[tT]he/"), which is perfect for specifying patterns in programs or command-lines
  - Regex can be implemented as an FSA (Finite State Automata)

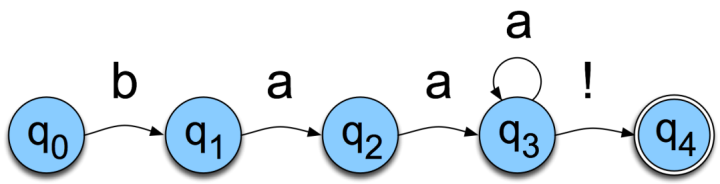
- Finite state automata
  - Graphs: **Nodes** are states and **edges** are transitions among states
  - FSA has a wide range of uses

- FSA vs Regex
  - FSA can be described with a regular expression
  - Regular expression is a textual way of specifying the structure of FSA

FSA as Graphs

FSA is non-examinable

- Example regular expression: /baa+!/
- Corresponding FSA

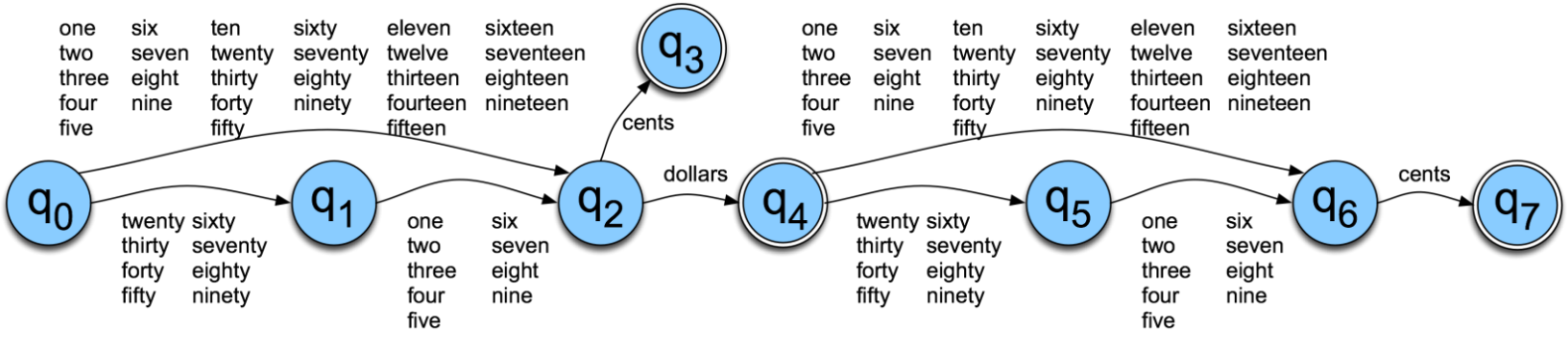


It has 5 states (q_0 to q_4)
b, **a**, and **!** are in its alphabet
q_0 is the start state
q_4 is an accept state
It has 5 transitions

FSA

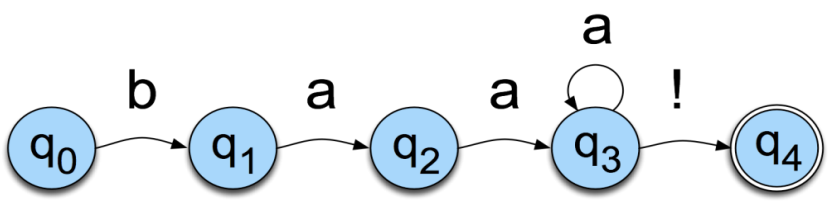
- A set of states: Q
- A finite alphabet: Σ
- A start state
- A set of accept/final states
- A transition function that maps $Q \times \Sigma \rightarrow Q$

Don't take term **alphabet** word too narrowly;
it just means we need **a finite set of symbols** in the input.



Yet Another View

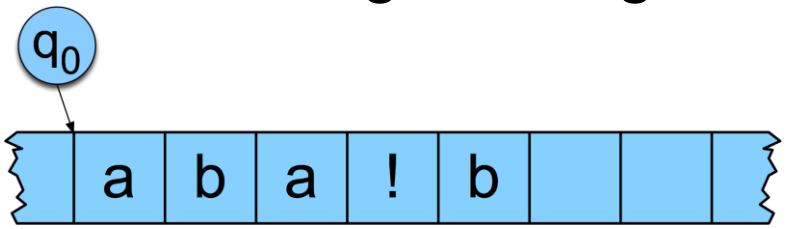
- An FSA can ultimately be represented as a **table**



If currently in state q_1 and an input is ***a***, then go to state q_2

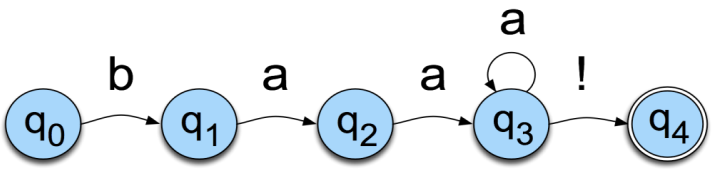
	Input		
State	b	a	!
q_0	1		
q_1		2	
q_2		3	
q_3		3	4
q_4:			

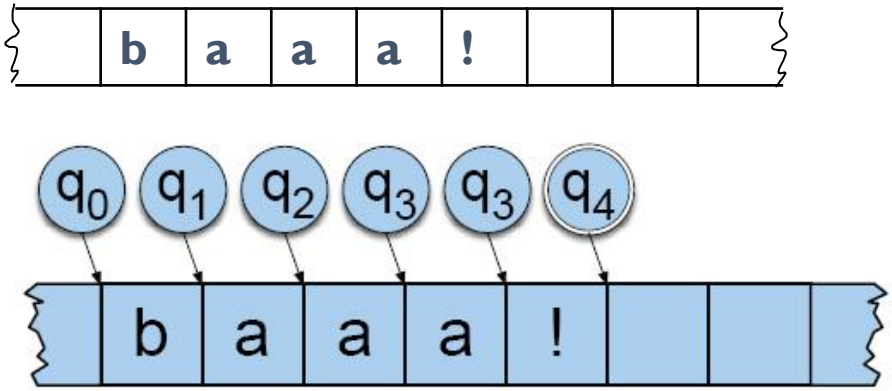
- **Recognition** is the process of determining if a string should be accepted by a machine



Recognition (D-Recognize)

- D-Recognize (in a tap-view)
  - Starting the process from the **start state**
  - Examining the **current input**
  - Consulting **the table**
  - Going **to a new state** and updating the tape pointer.
  - Until you run **out of tape.** Accept?

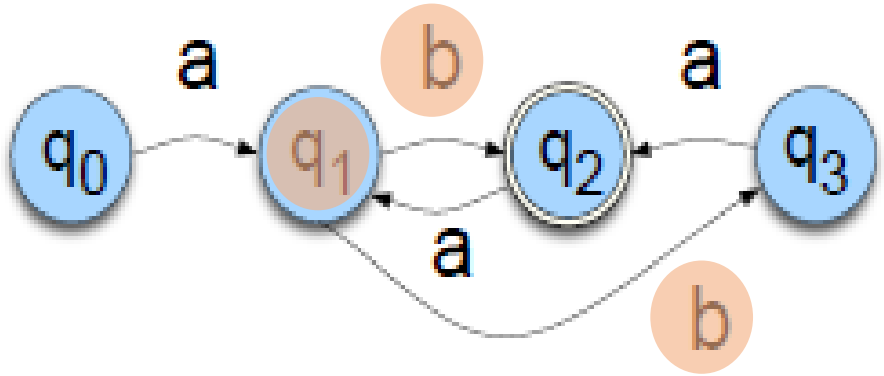




	Input		
State	b	a	!
q_0	1		
q_1		2	
q_2		3	
q_3		3	4
q_4:			

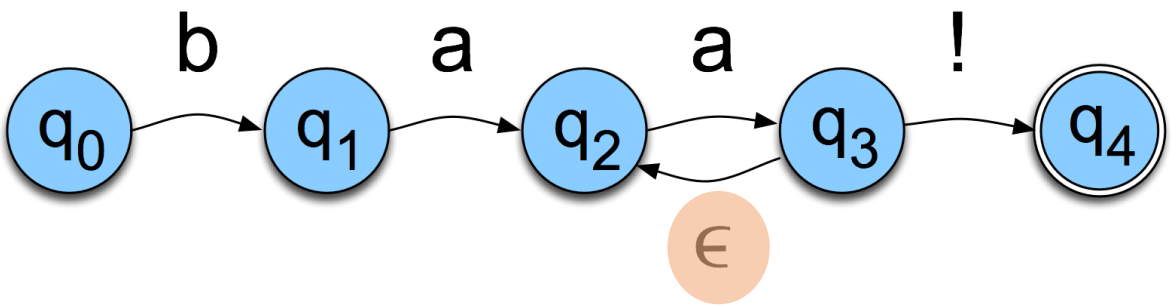
Deterministic FSA vs Non-Deterministic FSA

- If currently at a state, given an input
  - Deterministic FSA: There is only one next state to move to
  - Non-deterministic FSA: There are more than one possible states to move to



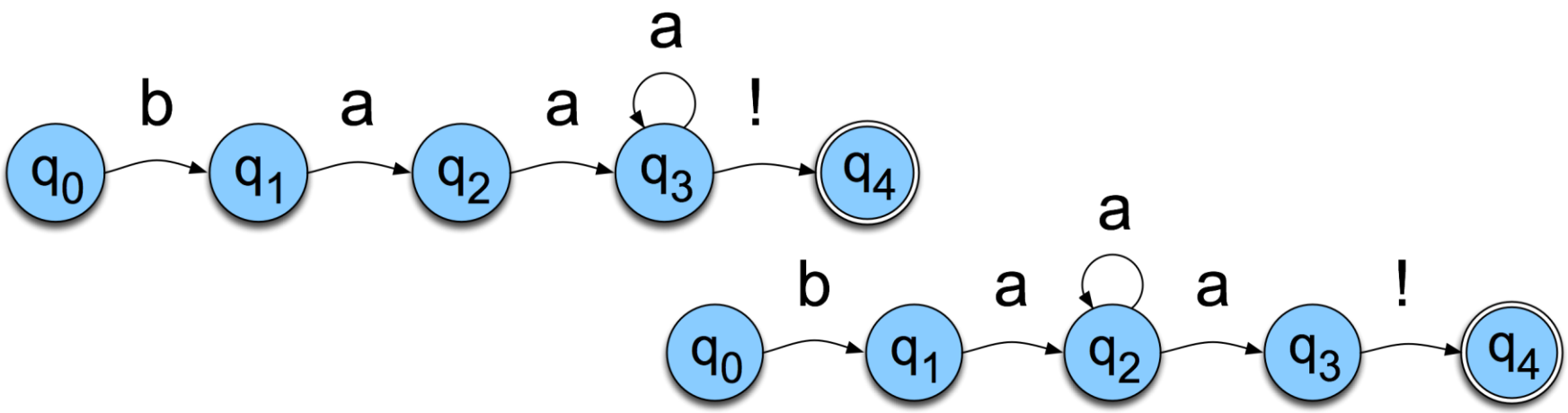
Deterministic FSA vs Non-Deterministic FSA

- There exists other form of Non-deterministic FSA
  - Epsilon transitions
  - These transitions do not examine or advance the tape during recognition



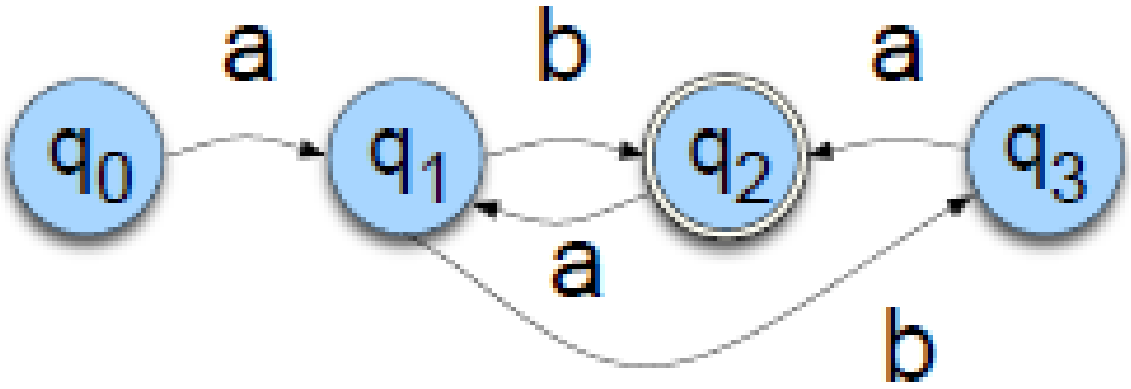
Deterministic FSA vs Non-Deterministic FSA

- Non-deterministic machines can be **converted** to deterministic through algorithm
- Non-deterministic machines **are not more powerful** than deterministic ones in terms of the languages they can and can't characterize
  - Not always obvious to users whether the regex that they've produced is deterministic or non-deterministic
  - Sometimes, non-determinism may look more natural (understandable)

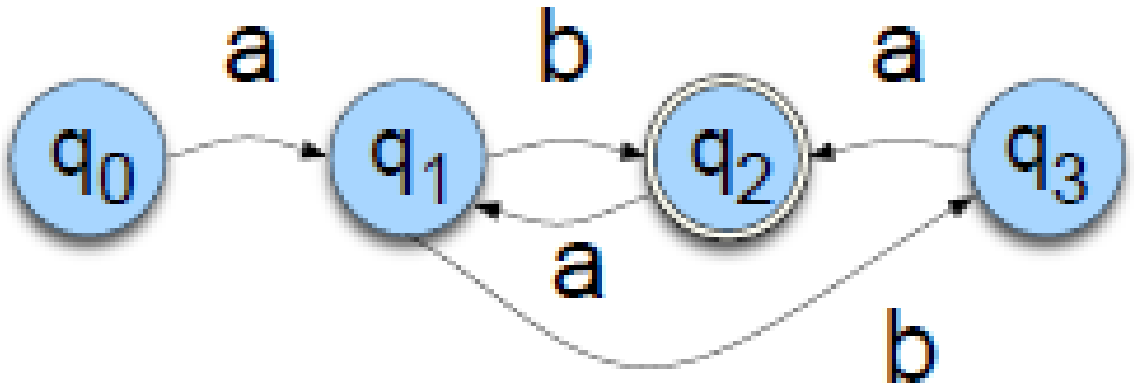


Additional Question on FSA

- Write a regular expression for the language accepted by the following NFSA.

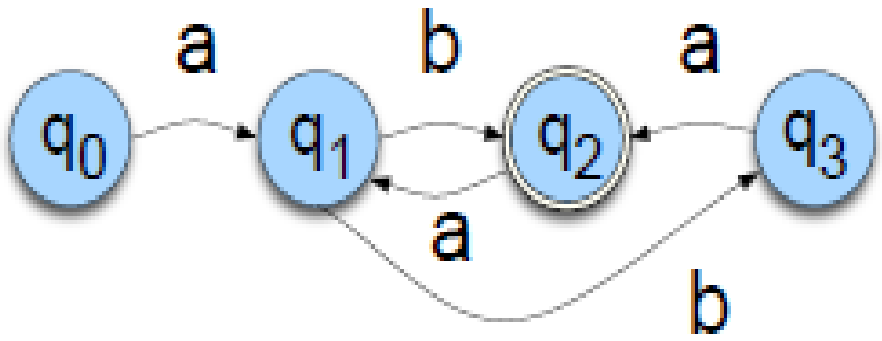


Hint: List the strings can be generated from the NFSA



- ab
- aba
- abab
- ababa
- ababab
- abaab
- ...

Answer



$q_0 ::= \varepsilon$

$q_1 ::= q_0a \mid q_2a$

$q_1 ::= a \mid q_2a$

$q_2 ::= q_1b \mid q_3a$

$q_3 ::= q_1b$

$q_2 ::= q_1b \mid q_1ba$

$q_2 ::= ab \mid q_2ab \mid aba \mid q_2aba$

$q_2 ::= [(ab)|(aba)]+$