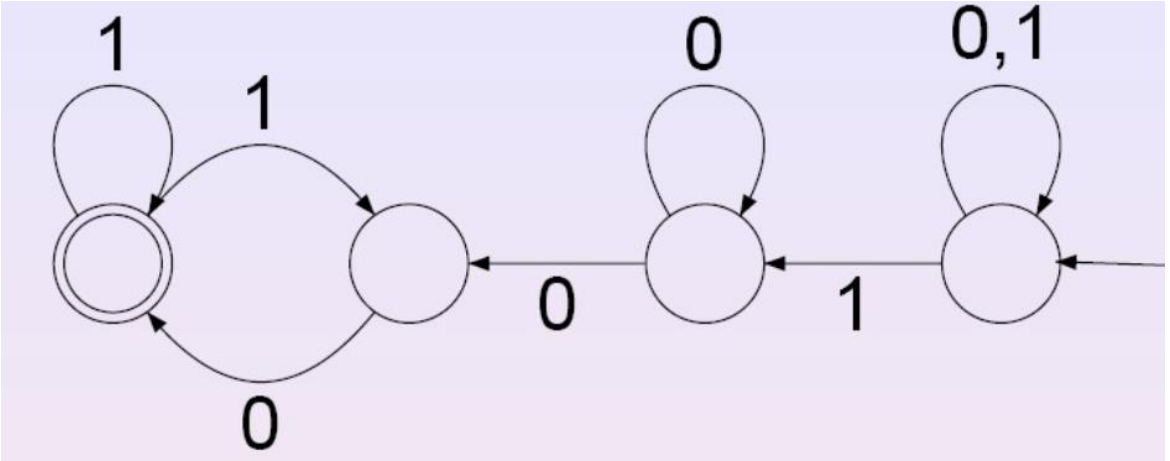
Formal languages and automata

Nondeterministic Finite Automata (NFA)

https://t.me/fla_uog

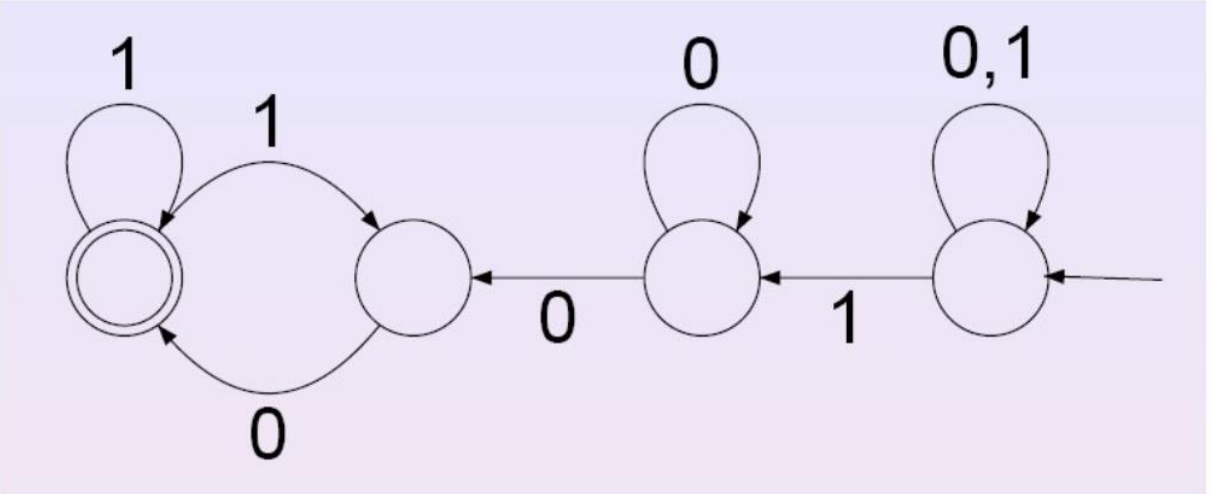
mh.olyaee@gmail.com

Example



- What happens with input 100?
  - There are multiple transitions from a state labeled with the same symbol.
  - State transitions are not deterministic any more: the next state is not uniquely determined by the current state and the current input. $\rightarrow$ Nondeterminism

Example

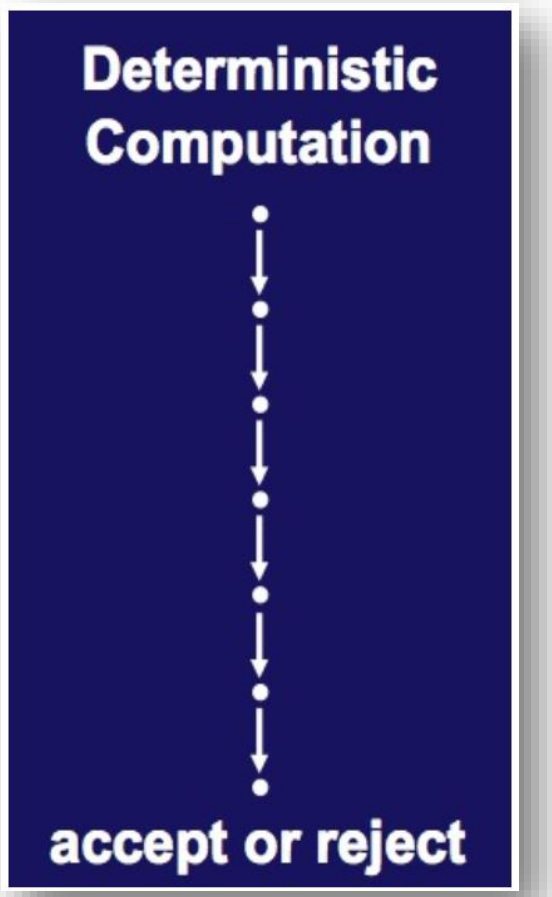


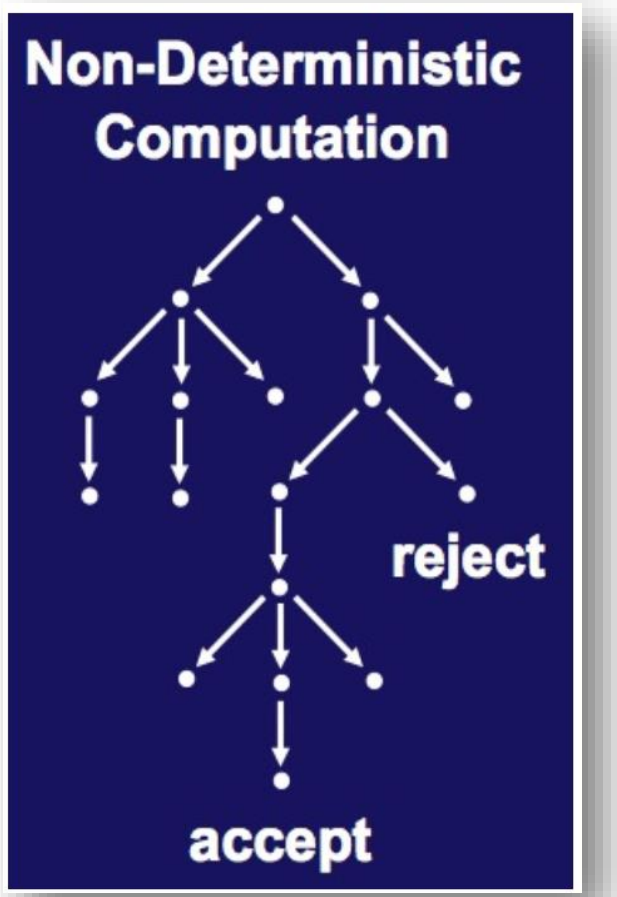
- We will say that this machine accepts a string if there is some path that reaches an accept state from a start state.

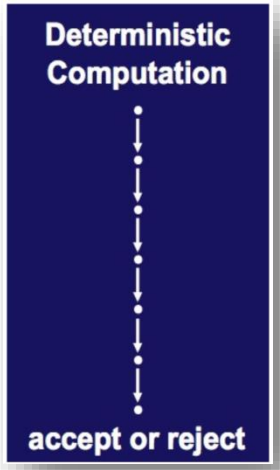
How dose Nondeterminism works?

- When a nondeterministic finite state automaton (NFA) reads an input symbol and there are multiple transitions labeled with that symbol
  - It splits into multiple copies of itself, and
  - follows all possibilities in parallel.

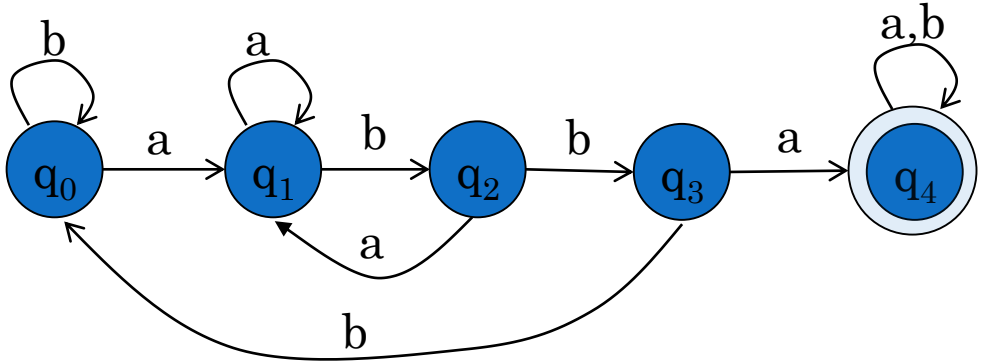
Deterministic vs Nondeterministic



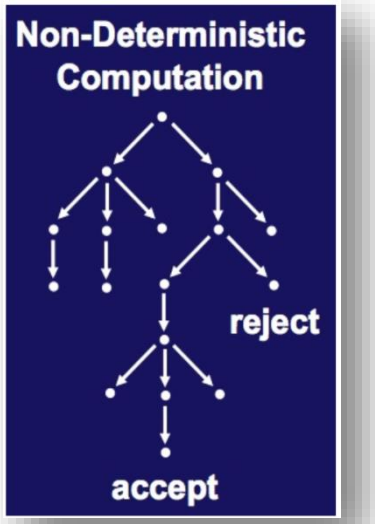




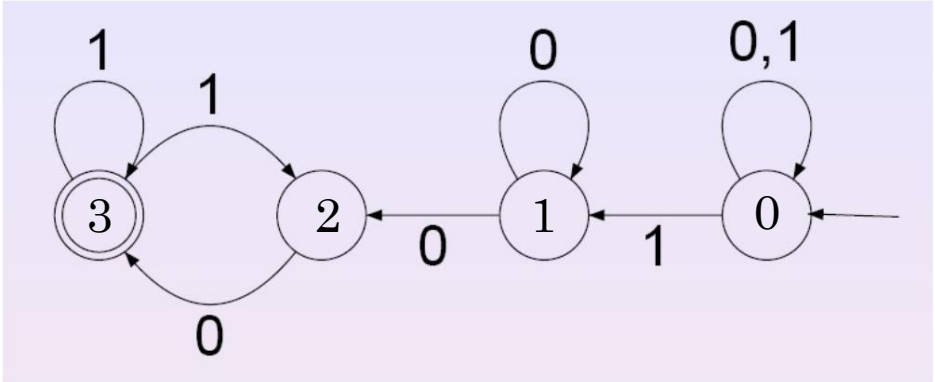
DFA



$\delta^*(q_0,\text{abbb}): q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_3 \rightarrow q_0$



NFA



- We will say that this machine accepts a string if there is some path that reaches an accept state from a start state.

$\delta^*(q_0,1001)$: $q_0 \to q_0 \to q_0 \to q_0 \to q_0$

$q_0 \to q_1 \to q_1 \to q_1 \to ?$

$q_0 \to q_1 \to q_2 \to q_3 \to q_2$

$q_0 \to q_1 \to q_2 \to q_3 \to q_3$

How dose Nondeterminism works?

- When a nondeterministic finite state automaton (NFA) reads an input symbol and there are multiple transitions labeled with that symbol
  - It splits into multiple copies of itself, and
  - follows all possibilities in parallel.
- Each copy of the machine takes one of the possible ways to proceed and continues as before.
- If there are subsequent choices, the machine splits again.
  - We have an unending supply of these machines that we can boot at any point to any state!

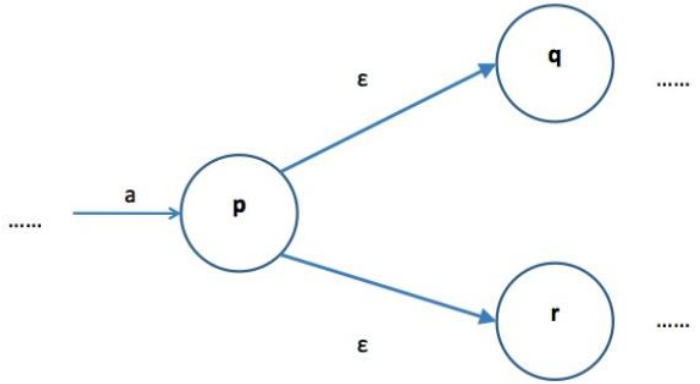
DFAs and NFAs; Other differences

- A state need not have a transition with every symbol in Σ
  - No transition with the next input symbol? $\Rightarrow$ that copy of the machine dies, along with the branch of computation associated with it.
  - If any copy of the machine is in a final state at the end of the input, the NFA accepts the input string.
- NFAs can have transitions labeled with ϵ – the empty string.

δ(p,….cbb…)→?

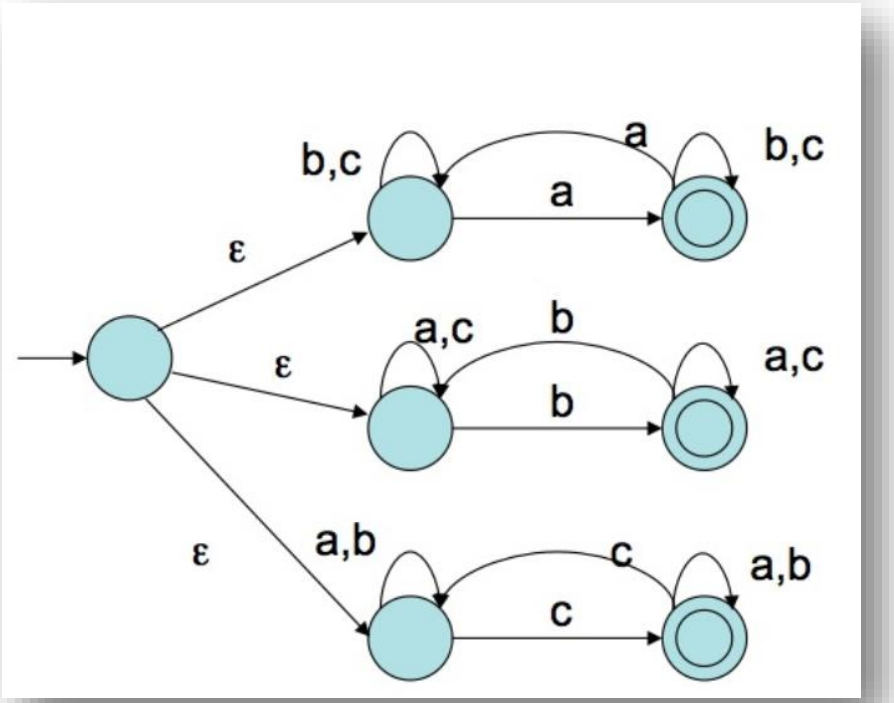
ε-transition

- If a transition with ϵ label is encountered, something similar happens:
  - The machine does not read the next input symbol.
  - It splits into multiple copies, one following each ϵ transition, and one staying at the current state.



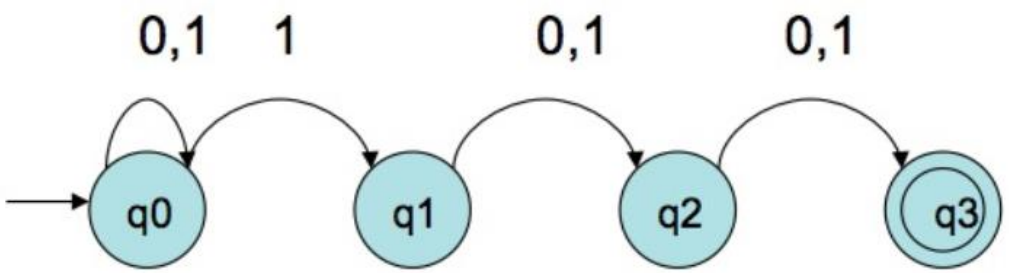
- What the NFA arrives at p (say after having read input *a*, it splits into 3 copies

NFA example



- Accepts all strings over $\Sigma = \{a, b, c\}$ with at least one of the symbols occuring an odd number of times.
- For example, the machine copy taking the upper ϵ transition guesses that there are an odd number of *a*'s and then tries to verify it.

NFA Example



- Accepts all strings over $\Sigma = \{0, 1\}$ where the 3rd symbol from the end is a 1.
  - How do you know that a symbol is the 3rd symbol from the end?
- The start state guesses every 1 is the 3rd from the end, and then the rest tries to verify that it is or it is not.
  - The machine dies if you reach the final state and you get one more symbol.

Example

- یک ماشین NFA طراحی کنید که رشته های حاوی aa را بپذیرد

Example

- یک NFA طراحی کنید که رشته های حاوی aa یا bb را بپذیرد

Formal definition

- A Nondeterministic Finite State Acceptor (NFA) is defined as the 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$ where
  - Q is a finite set of states
  - Σ is a finite set of symbols – the alphabet
  - $\delta : Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q$, is the next-state function
    - $2^Q = \{P | P \subseteq Q\}$
  - $q_0 \in Q$ is the (label of the) start state
  - $F \subseteq Q$ is the set of final (accepting) states
- δ maps states and inputs (including ϵ) to a set of possible next states

Acceptance

- What input strings have been accept by NFA?
- $L(m) = \{w \mid \delta^*(q_0, w) \cap F \neq \phi\}$

Power of NFA

- We know that DFAs accept regular languages.
- Are NFAs strictly more powerful than DFAs?
  - Are there languages that some NFA will accept but no DFA can accept?
- It turns out that NFAs and DFAs accept the same set of languages.
  - Q is finite $\Rightarrow |2^Q| = |\{P|P \subseteq Q\}| = 2^{|Q|}$ is also finite.

NFA and DFA are equivalent

THEOREM

Every NFA has an equivalent DFA.

PROOF IDEA

- Convert the NFA to an equivalent DFA that accepts the same language.
- If the NFA has k states, then there are 2^k possible subsets (still finite)
- The states of the DFA are labeled with subsets of the states of the NFA
- Thus the DFA can have up to 2^k states.

Transform NFA to DFA