Finite State Machines (Automatons)

Computation and Discrete Structures III

Juan Marcos Caicedo Mejía – jmcaicedo@icesi.edu.co

Department of Computing and Intelligent Systems
Barberi Faculty of Engineering, Design and Applied Sciences
ICESI University

Material adapted from Professor Oscar Bedoya

2026

Contents

Contenido

Languages, Machine Types and Grammars

Type	Languages	Machine Type	Grammar Rules
0	Recursively enumerable	Turing machine	Unrestricted
1	Context-sensitive	Linear bounded automaton	$\alpha \to \beta, \quad \lvert\alpha\rvert \leq \lvert\beta\rvert$
2	Context-free	Pushdown automaton	$A \to \gamma$
3	Regular	Finite automaton	$A \to aB$ $A \to a$

Regular languages

A **finite automaton** can be constructed for each of these languages:

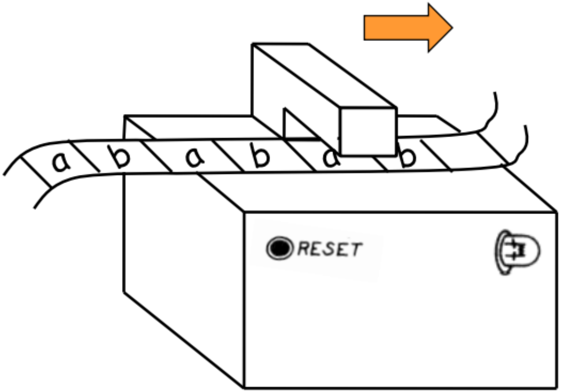
- $\{a\}^*$
- $\{a\}^* \cup \{b\}^*$
- $\{a\}^* \cdot \{b\}^*$
- $\{a, bc\}^*$
- $\{a\} \cdot \{b, c, ab\}$
- $\{(ab)^n \mid n \geq 0\}$
- $\{a^n b^m \mid n \geq 0, m \geq 0\}$
- $\{a^l b^m c^n \mid l \geq 0, m \geq 0, n \geq 0\}$

Non-regular languages

No finite automaton can be constructed for any of these languages:

- $\{a^n b^n \mid n \geq 0\}$, not regular
- $\{a^n b^{2n} \mid n \geq 0\}$, not regular
- $\{wcw \mid w \in \{a, b\}^*\}$, not regular

Finite automaton



A finite automaton can be designed to **accept**, for example, the language $\{ab\}^* = \{\epsilon, ab, abab, ababab, ...\}$

Regular languages

Consider the regular language L represented by:

$$c^*(a \cup bc^*)^*$$

- Does $w_1 = abc^5ab$ belong to L?
- Does $w_2 = cabac^3bc$ belong to L?

Regular languages

Consider the regular language L represented by:

$$c^*(a \cup bc^*)^*$$

- Does $w_1 = abc^5ab$ belong to L?
- Does $w_2 = cabac^3bc$ belong to L?

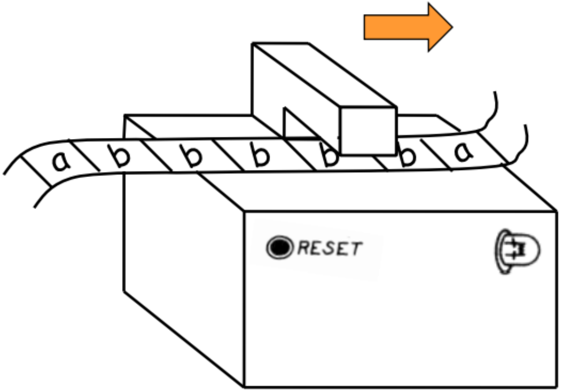
To determine if a string w is generated by a language L, a **finite automaton** can be created

Finite automata

Finite automata

- Black box that accepts tape data as input
- A light bulb represents the output; when input is accepted by the automaton, it lights up
- Reset button
- Machine operation consists of a set of internal states

Finite automata



The automaton's head can only **read** (cannot write) and always moves to the **right**

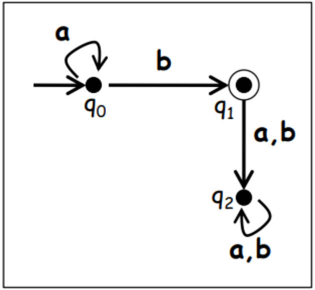
Finite automata

Consider an automaton that accepts strings in $\{a, b\}^*$ that have a single b at the end of the string

Finite automata

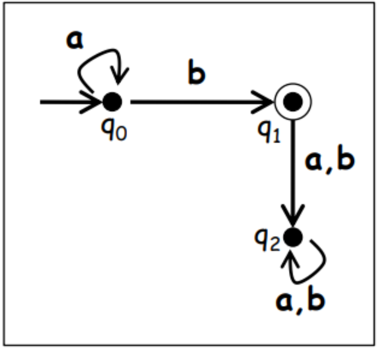
Finite automata can be represented by a directed graph known as the **transition diagram**

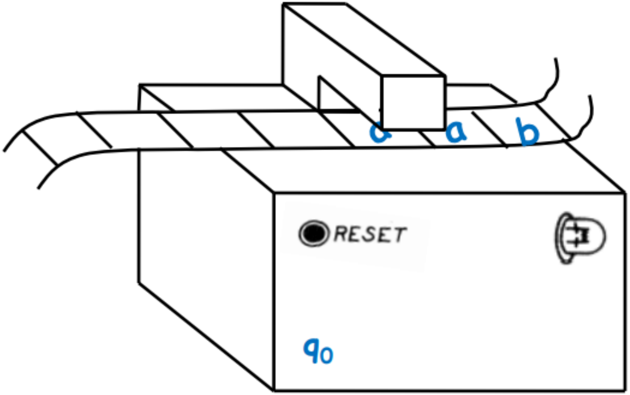
- Nodes (**states**)
  - Initial state
  - Accepting state
- Edges (**transitions**)



Finite automata

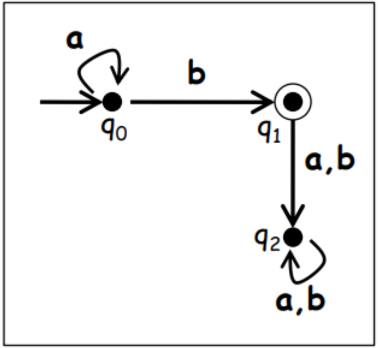
Each **step** in the automaton depends on: (symbolRead, currentState).

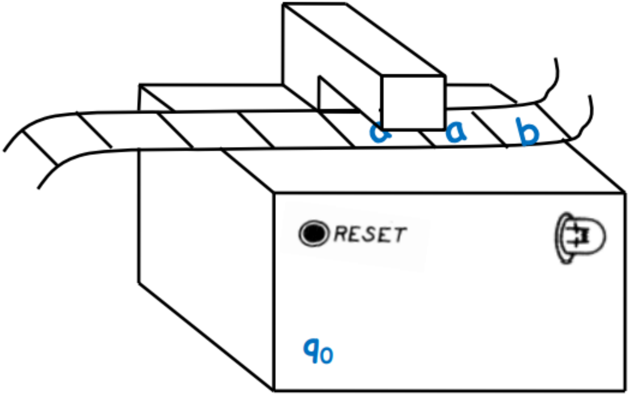




Finite automata

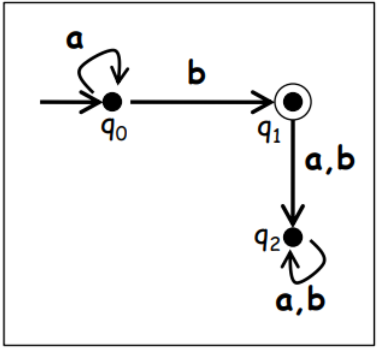
Follow the computation for the string *aab*

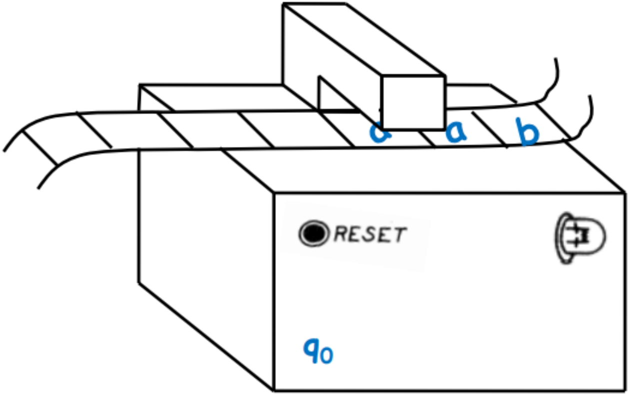




Finite automata

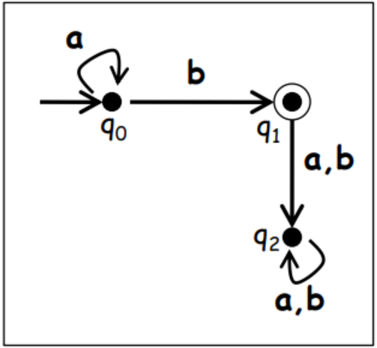
$$(q_0, a) \Rightarrow q_0$$

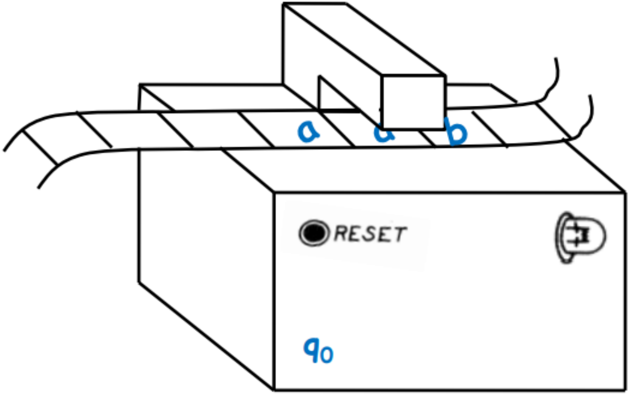




Finite automata

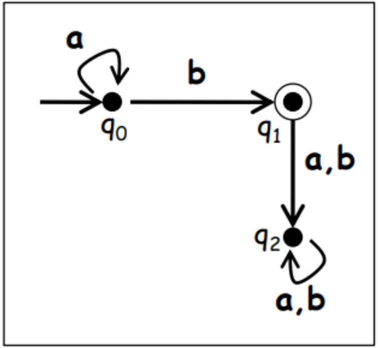
$$(q_0, a) \Rightarrow q_0$$

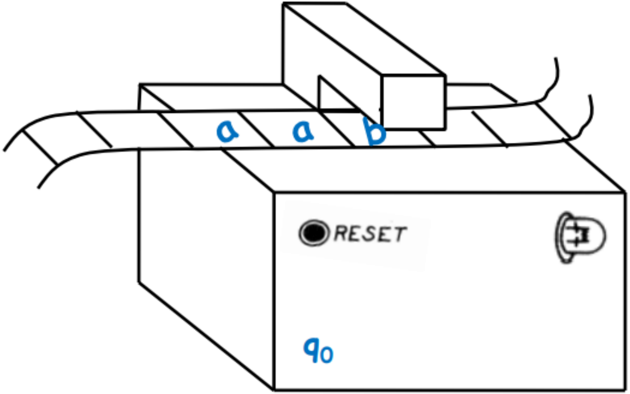




Finite automata

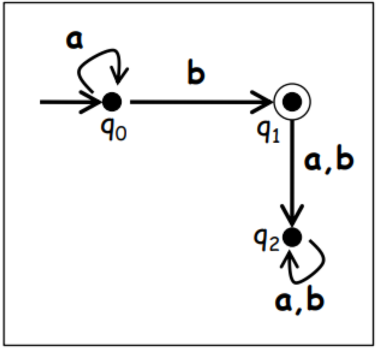
$$(q_0, a) \Rightarrow q_0$$

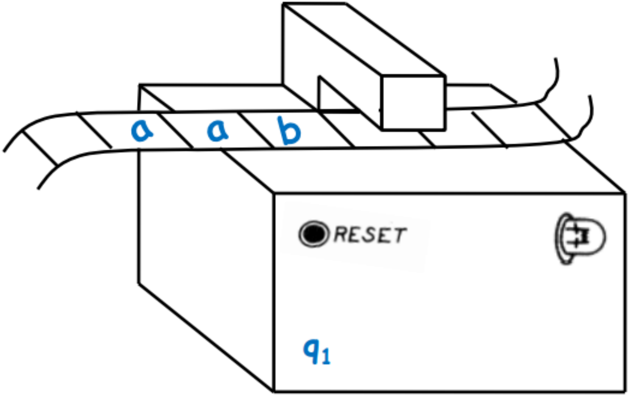




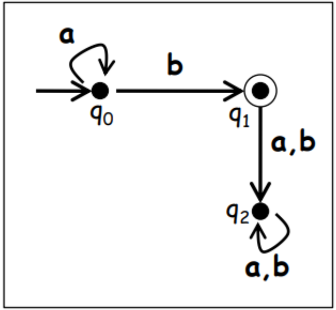
Finite automata

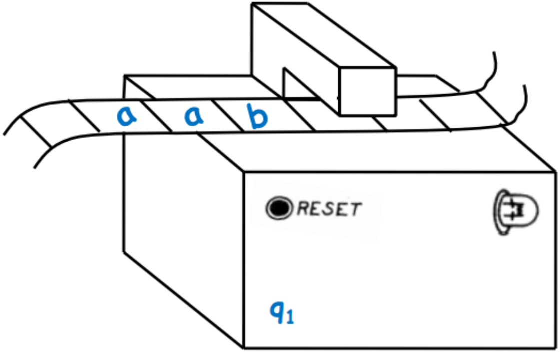
$$(q_0, b) \Rightarrow q_1$$





$$(q_0, b) \Rightarrow q_1$$

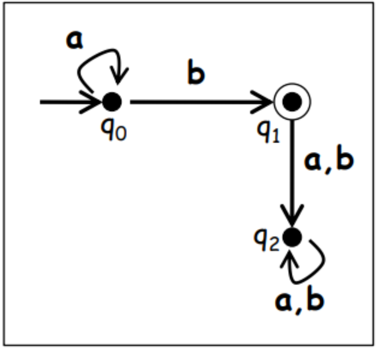


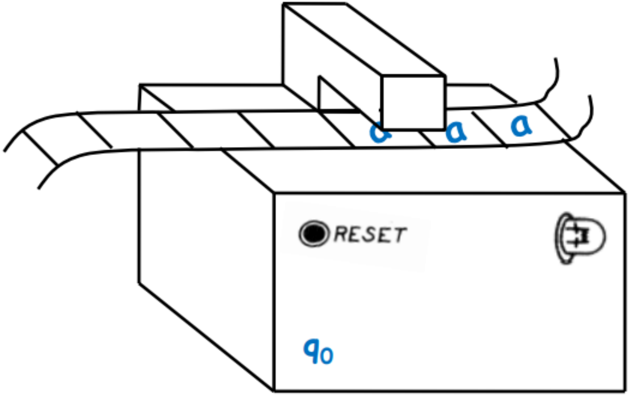


Since all symbols on the tape are consumed and q1 is an accepting state, the automaton recognizes **aab**

Finite automata

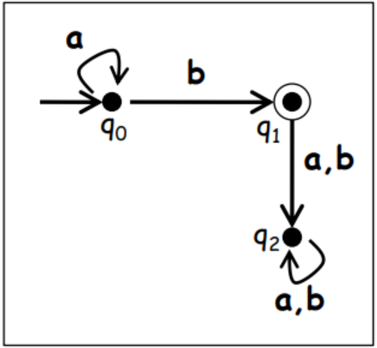
Indicate if the string *aaa* is accepted:

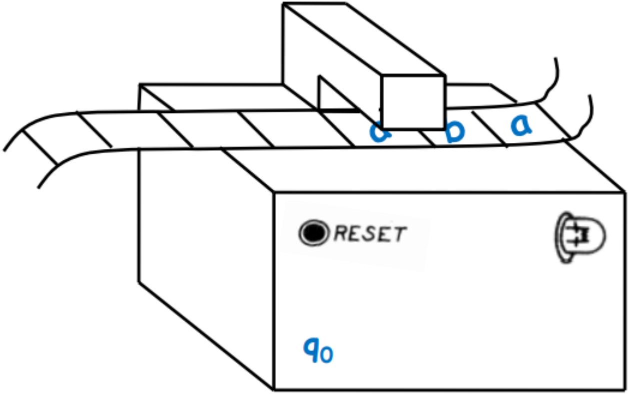




Finite automata

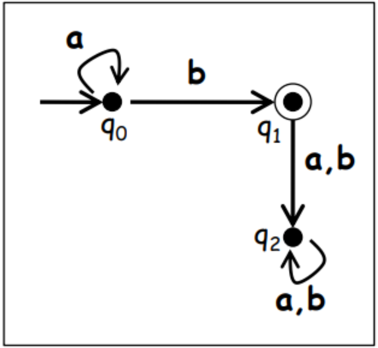
Indicate if the string *aba* is accepted:

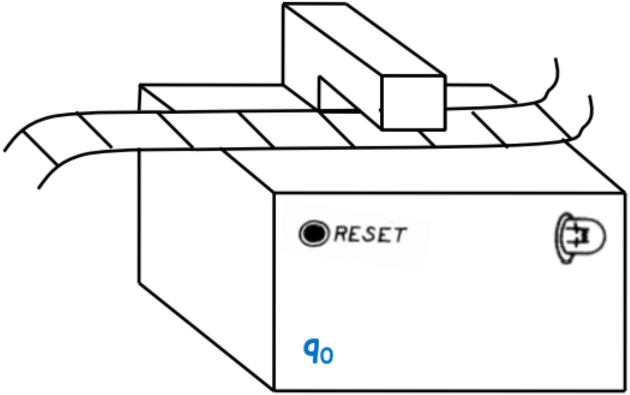




Finite automata

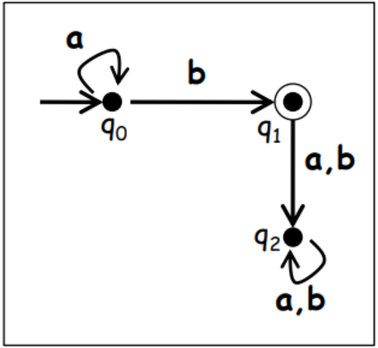
Indicate if the empty string ϵ is accepted:

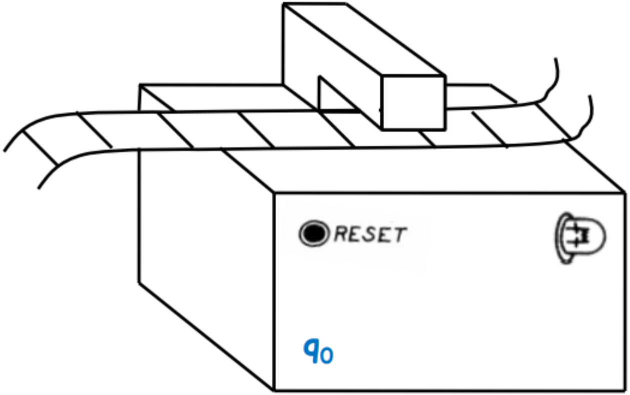




Finite automata

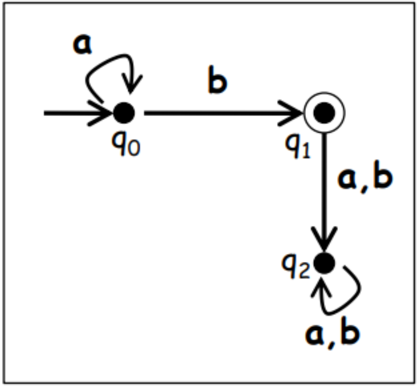
Indicate a regular expression for the automaton





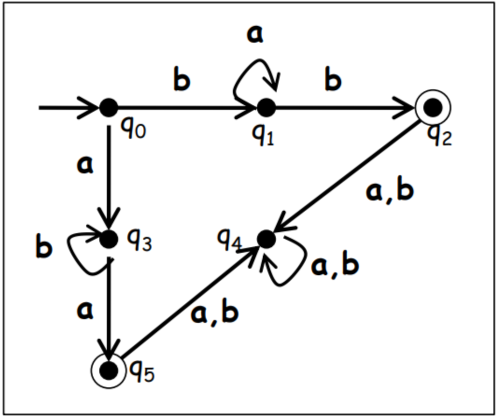
Finite automata

A regular expression for the automaton is a^*b, meaning it accepts strings that belong to that language.



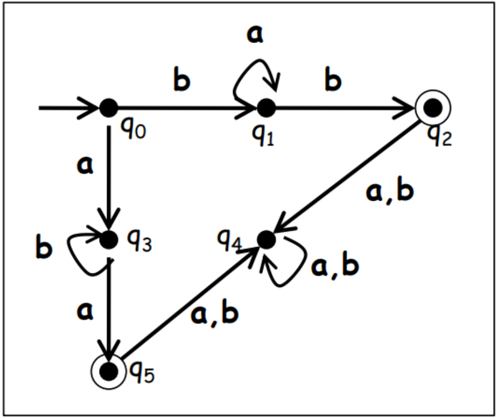
Finite automata

From the automaton in the image, indicate if the following strings are accepted: ϵ, *ab*, *bab*, *ba*^{4}*b*, *ab*^{3}*a*.



Finite automata

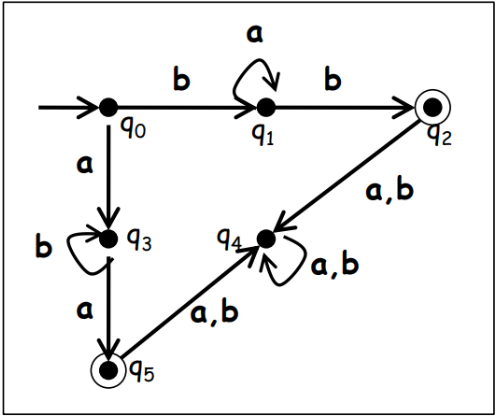
Indicate a regular expression that represents the language accepted by the automaton:



Finite automata

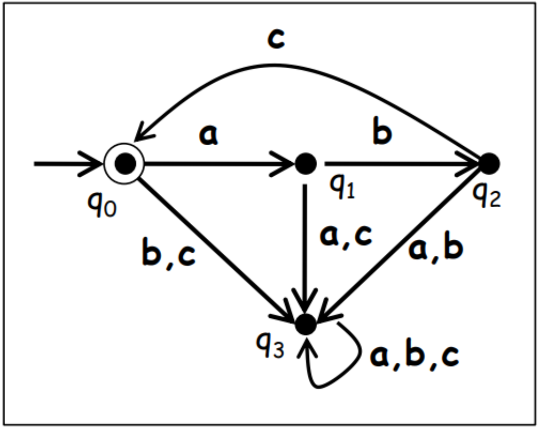
A regular expression that represents the language accepted by the automaton is:

$$(ab^*a) \cup (ba^*b)$$



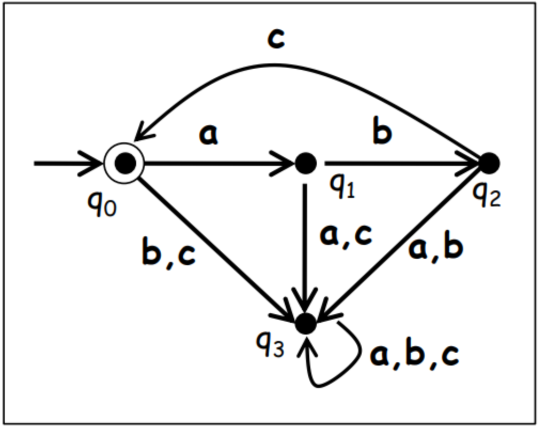
Finite automata

From the automaton in the image, indicate if the following strings are accepted: ϵ, *abc*, $(abc)^2$, *aabc*, *aba*, *abca*



Finite automata

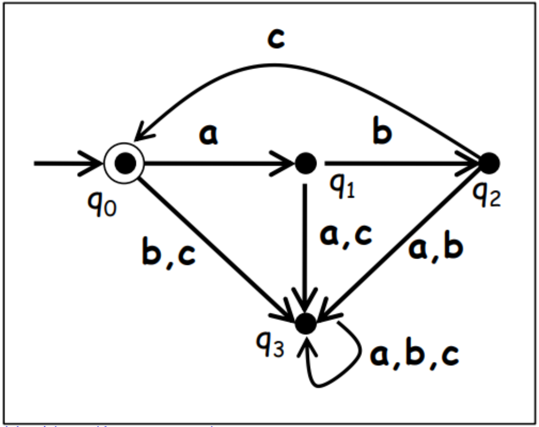
Indicate a regular expression that represents the language accepted by the automaton:



Finite automata

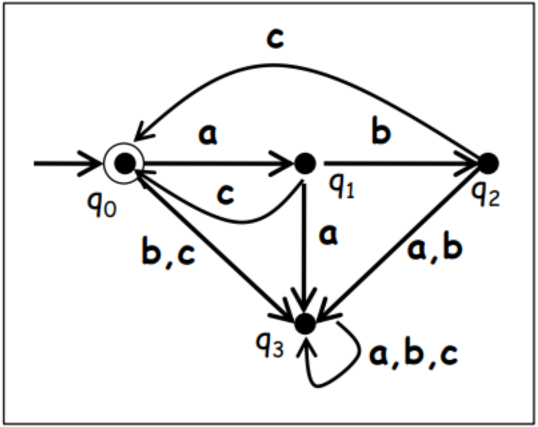
A regular expression that represents the language accepted by the automaton is:

$$(abc)^*$$



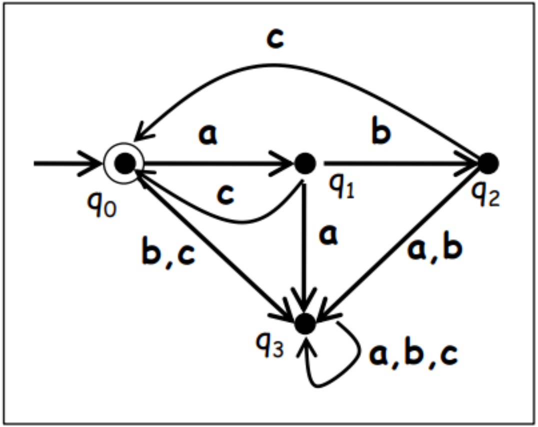
Finite automata

From the automaton in the image, indicate if the following strings are accepted: ϵ, abc, $abcac$, $(ac)^{10}$, $a^2b^2c^2$, $(abc)^2$, $(abc)^2(ac)^3$



Finite automata

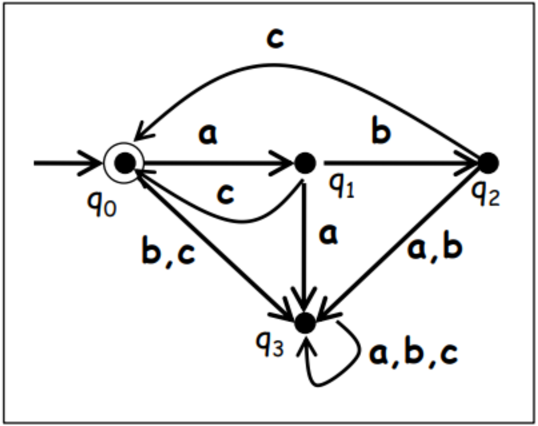
Indicate a regular expression that represents the language accepted by the automaton:



Finite automata

A regular expression that represents the language accepted by the automaton is:

$$(abc \cup ac)^*$$

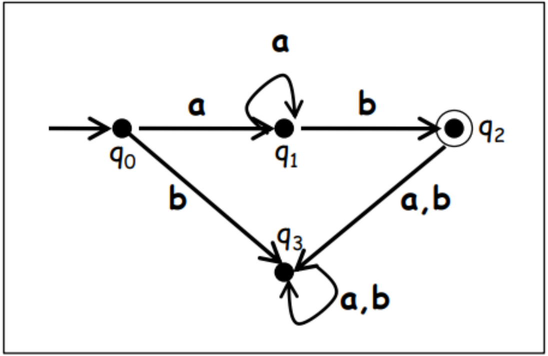


Finite automata

Design a finite automaton that accepts a^+b

Finite automata

Design a finite automaton that accepts a^+b



Regular expression: a^+b

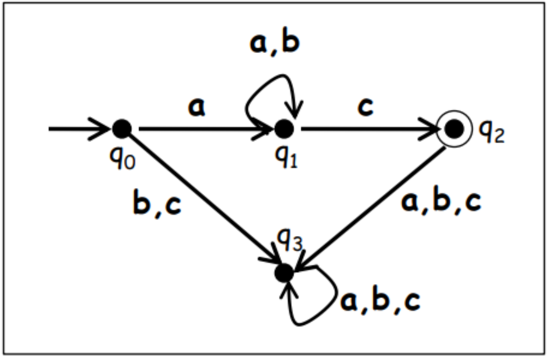
Language: $\{ab, aab, aaab, \ldots\}$

Finite automata

Design a finite automaton that accepts $a(a \cup b)^*c$

Finite automata

Design a finite automaton that accepts $a(a \cup b)^*c$



Regular expression: $a(a \cup b)^*c$

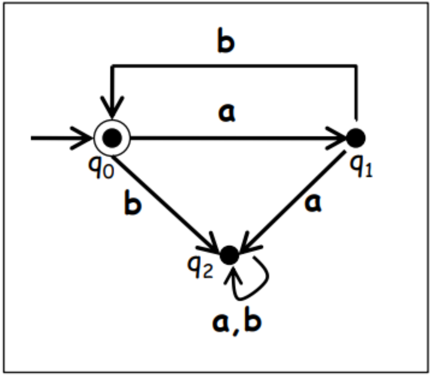
Language: $\{ac, aac, abc, aabc, \ldots\}$

Finite automata

Design a finite automaton that accepts $(ab)^*$

Finite automata

Design a finite automaton that accepts $(ab)^*$



Regular expression: $(ab)^*$

Language: $\{\epsilon, ab, abab, ababab, ...\}$

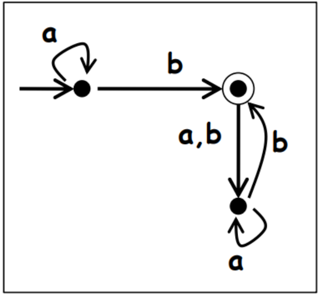
Finite automata

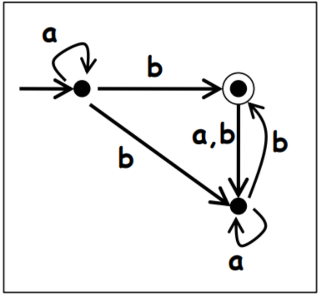
Kleene's Theorem

A language is regular if and only if it is accepted by a finite automaton

Finite automata

Finite automata are divided into deterministic finite automata (**DFA**) and non-deterministic finite automata (**NFA**)





- Given a state *q* and a symbol *x*, there is a single transition edge. **(DFA)**
- Given a state *q* and a symbol *x*, there are multiple possible transitions (and could be also, none). **(NFA)**

Contenido

Deterministic finite automata (DFA)

A DFA is a collection of five elements:

- An alphabet Σ
- A finite collection of states Q
- An initial state q_0
- A finite collection of accepting states T
- A function $\delta : Q \times \Sigma \Rightarrow Q$ that determines the **unique** next state for the pair (q_i, σ) corresponding to the current state q_i and input σ

Deterministic finite automata (DFA)

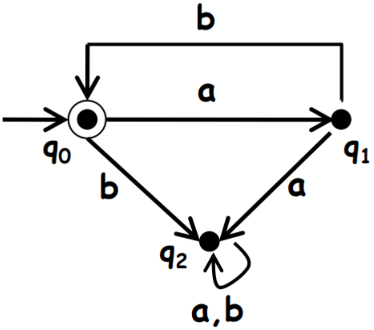
A DFA is a collection of five elements:

- An alphabet Σ
- A finite collection of states Q
- An initial state q_0
- A finite collection of accepting states T
- A function $\delta : Q \times \Sigma \Rightarrow Q$ that determines the **unique** next state for the pair (q_i, σ) corresponding to the current state q_i and input σ

δ must be a **function** for determinism to exist

Deterministic finite automata (DFA)

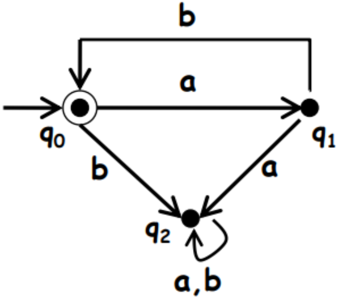
- An alphabet Σ
- A finite collection of states Q
- An initial state q_0
- A finite collection of accepting states T
- A function $\delta : Q \times \Sigma \Rightarrow Q$



Deterministic finite automata (DFA)

- Σ
- Q
- Initial state
- T
- $\delta : Q \times \Sigma \Rightarrow Q$

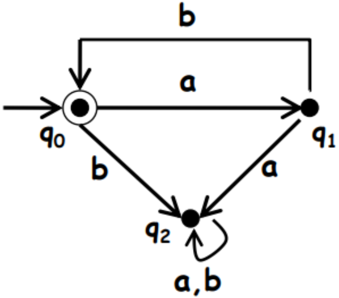
δ	a	b
q_0		
q_1		
q_2		



Deterministic finite automata (DFA)

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2\}$
- Initial state q_0
- $T = \{q_0\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

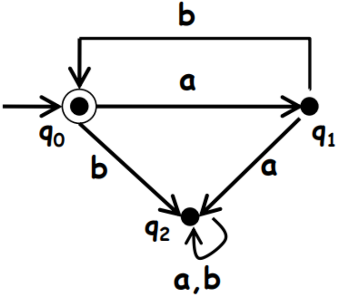
δ	a	b
q_0		
q_1		
q_2		



Deterministic finite automata (DFA)

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2\}$
- Initial state q_0
- $T = \{q_0\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

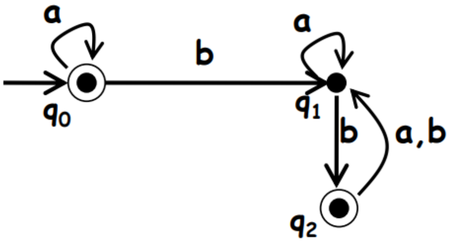
δ	a	b
q_0	q_1	q_2
q_1	q_2	q_0
q_2	q_2	q_2



Deterministic finite automata (DFA)

- Σ
- Q
- Initial state
- T
- $\delta : Q \times \Sigma \Rightarrow Q$

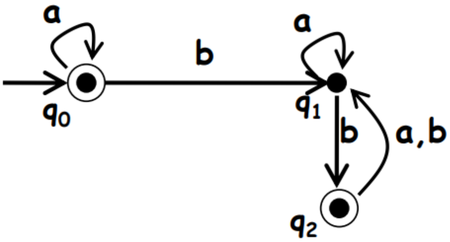
δ	a	b
q_0		
q_1		
q_2		



Deterministic finite automata (DFA)

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2\}$
- Initial state q_0
- $T = \{q_0, q_2\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

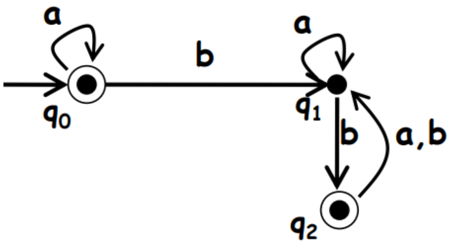
δ	a	b
q_0		
q_1		
q_2		



Deterministic finite automata (DFA)

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2\}$
- Initial state q_0
- $T = \{q_0, q_2\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_2
q_2	q_1	q_1



Deterministic finite automata (DFA)

Show the transition diagram for the automaton:

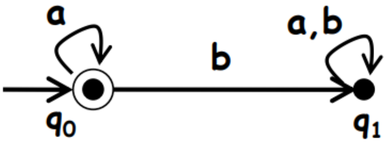
- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$
- Initial state q_0
- $T = \{q_0\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_1

Deterministic finite automata (DFA)

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$
- Initial state q_0
- $T = \{q_0\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_1

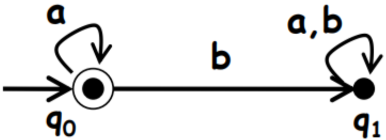


Deterministic finite automata (DFA)

Indicate the accepted language

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$
- Initial state q_0
- $T = \{q_0\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_1

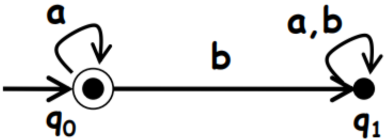


Deterministic finite automata (DFA)

The language accepted by the automaton is a^*

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$
- Initial state q_0
- $T = \{q_0\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_0	q_1
q_1	q_1	q_1



Deterministic finite automata (DFA)

Show the transition diagram for the automaton:

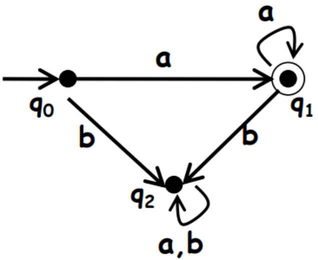
- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2\}$
- Initial state q_0
- $T = \{q_1\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_2
q_1	q_1	q_2
q_2	q_2	q_2

Deterministic finite automata (DFA)

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2\}$
- Initial state q_0
- $T = \{q_1\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_2
q_1	q_1	q_2
q_2	q_2	q_2

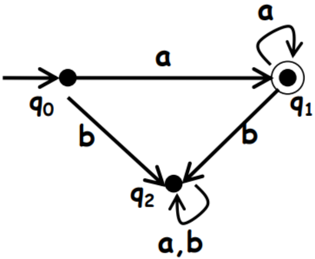


Deterministic finite automata (DFA)

Indicate the accepted language

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2\}$
- Initial state q_0
- $T = \{q_1\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_2
q_1	q_1	q_2
q_2	q_2	q_2

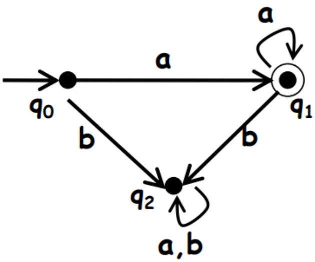


Deterministic finite automata (DFA)

The language accepted by the automaton is a^+

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2\}$
- Initial state q_0
- $T = \{q_1\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_2
q_1	q_1	q_2
q_2	q_2	q_2



Deterministic finite automata (DFA)

Show the transition diagram for the automaton:

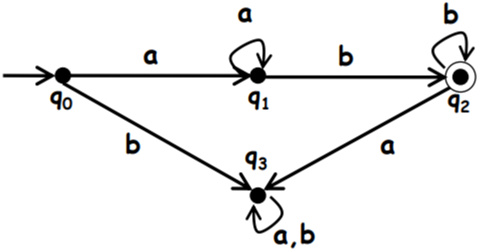
- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2, q_3\}$
- Initial state q_0
- $T = \{q_2\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_3
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_3	q_3

Deterministic finite automata (DFA)

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2, q_3\}$
- Initial state q_0
- $T = \{q_2\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_3
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_3	q_3

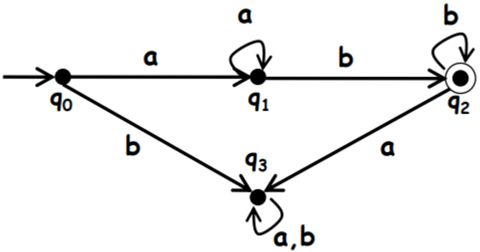


Deterministic finite automata (DFA)

Indicate the accepted language

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2, q_3\}$
- Initial state q_0
- $T = \{q_2\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_3
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_3	q_3

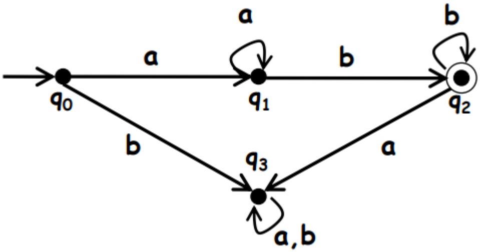


Deterministic finite automata (DFA)

The language accepted by the automaton is a^+b^+

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2, q_3\}$
- Initial state q_0
- $T = \{q_2\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_3
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_3	q_3



Deterministic finite automata (DFA)

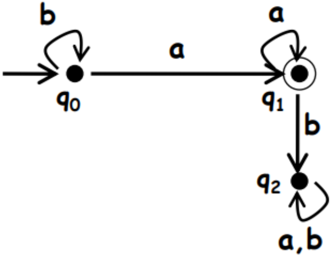
Design a DFA over $\Sigma = \{a, b\}$ that recognizes b^*a^+

- Show the transition diagram
- Express the automaton formally

Deterministic finite automata (DFA)

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2\}$
- Initial state q_0
- $T = \{q_1\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_0
q_1	q_1	q_2
q_2	q_2	q_2



Deterministic finite automata (DFA)

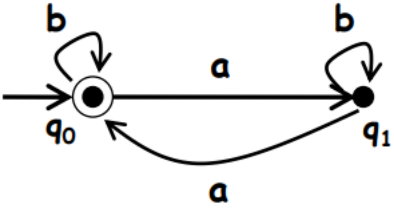
Design a DFA over $\Sigma = \{a, b\}$ that recognizes the language of all words containing an even number of a's. Strings with zero a's are accepted.

- Show the transition diagram
- Express the automaton formally

Deterministic finite automata (DFA)

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$
- Initial state q_0
- $T = \{q_0\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_0
q_1	q_0	q_1



Deterministic finite automata (DFA)

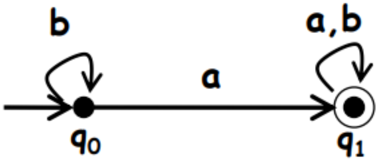
Design a DFA over $\Sigma = \{a, b\}$ that recognizes the language of all words that have at least one a

- Show the transition diagram
- Express the automaton formally
- Indicate the regular expression

Deterministic finite automata (DFA)

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$
- Initial state q_0
- $T = \{q_1\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_0
q_1	q_1	q_1

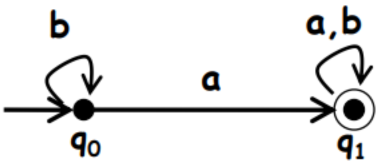


Deterministic finite automata (DFA)

Indicate the accepted language

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$
- Initial state q_0
- $T = \{q_1\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_0
q_1	q_1	q_1

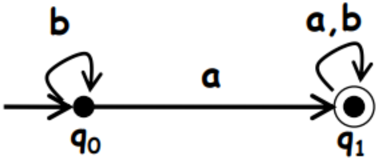


Deterministic finite automata (DFA)

The automaton accepts: $b^*a(a \cup b)^*$

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$
- Initial state q_0
- $T = \{q_1\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_0
q_1	q_1	q_1



Deterministic finite automata (DFA)

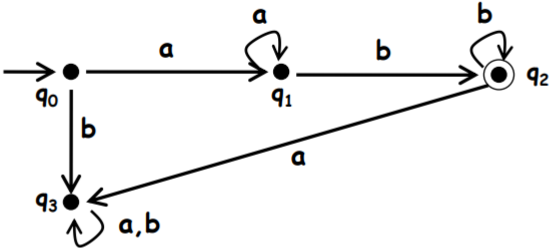
Design a DFA over $\Sigma = \{a, b\}$ that recognizes a^+b^+

- Show the transition diagram
- Express the automaton formally

Deterministic finite automata (DFA)

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2, q_3\}$
- Initial state q_0
- $T = \{q_2\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_3
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_3	q_3



Deterministic finite automata (DFA)

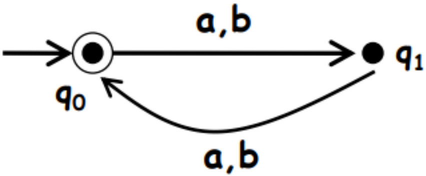
Design a DFA over $\Sigma = \{a, b\}$ that recognizes the language of all strings that have an even number of symbols (including the empty string).

- Show the transition diagram
- Express the automaton formally
- Indicate the regular expression

Deterministic finite automata (DFA)

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$
- Initial state q_0
- $T = \{q_0\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_1
q_1	q_0	q_0

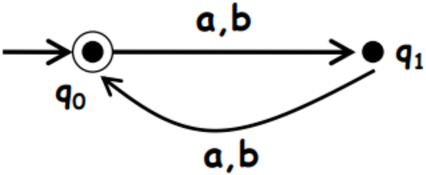


Deterministic finite automata (DFA)

The automaton accepts: $((a \cup b)(a \cup b))^*$ or equivalently, $(aa \cup ab \cup ba \cup bb)^*$

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$
- Initial state q_0
- $T = \{q_0\}$
- $\delta : Q \times \Sigma \Rightarrow Q$

δ	a	b
q_0	q_1	q_1
q_1	q_0	q_0



Deterministic finite automata (DFA)

* Design a DFA over $\Sigma = \{a, b\}$ that recognizes the language of all strings that begin and end with the same symbol in $\{a, b\}^*$

- Show the transition diagram
- Express the automaton formally
- Indicate the regular expression

Contenido

Non-deterministic finite automata (NFA)

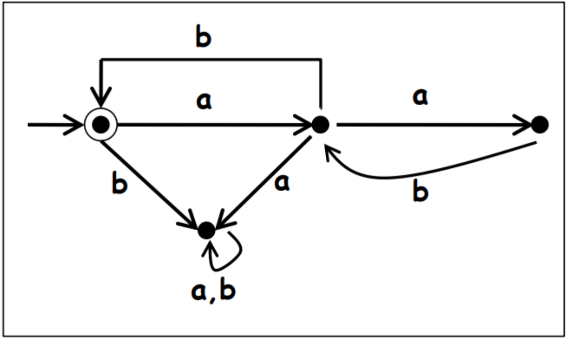
Non-deterministic finite automata

If zero, two or more transitions are allowed from some state using the same input symbol, the **finite automaton is non-deterministic**

Non-deterministic finite automata (NFA)

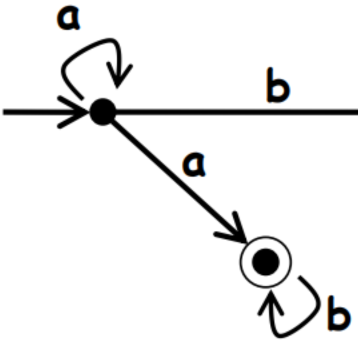
Non-deterministic finite automata

If zero, two or more transitions are allowed from some state using the same input symbol, the **finite automaton is non-deterministic**



Non-deterministic finite automata (NFA)

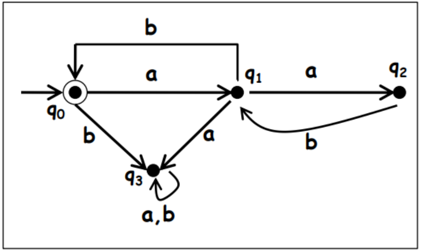
NFAs are used because they can be simpler than DFAs.

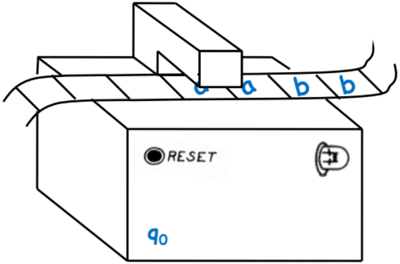


NFA that accepts $a^*b \cup a^+b^*$

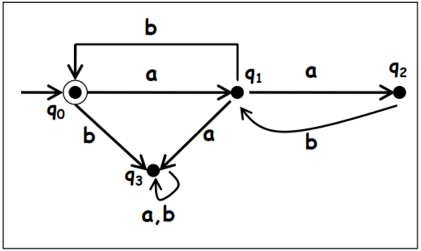
Deterministic finite automata (DFA)

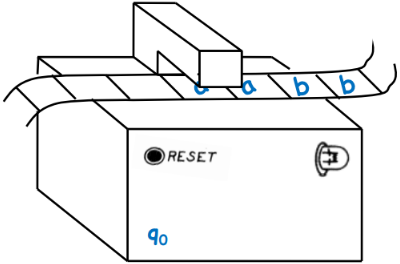
Does the finite automaton accept or reject the string *aabb*?





In an NFA, it can be assumed that if there exists a path in the transition diagram that ends in an accepting state, the automaton finds it





Non-deterministic finite automata (NFA)

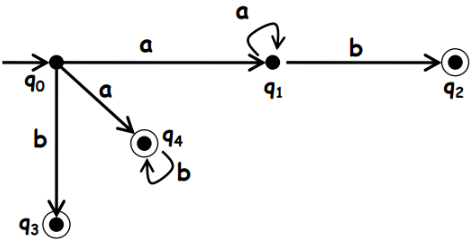
An NFA is a collection of five elements:

- An alphabet Σ
- A finite collection of states Q
- An initial state q_0
- A finite collection of accepting states T
- A relation Δ over $(Q \times \Sigma) \Rightarrow 2^Q$ called **transition relation**. 2^Q is the power set of Q (subsets of Q)

Non-deterministic finite automata (NFA)

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- Initial state q_0
- $T = \{q_2, q_3, q_4\}$
- $\Delta : Q \times \Sigma \Rightarrow 2^Q$

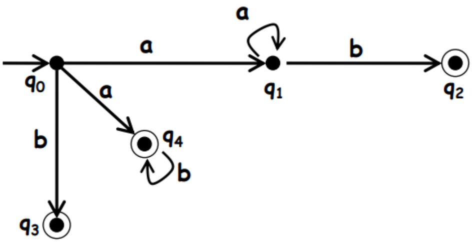
Δ	a	b
q_0		
q_1		
q_2		
q_3		
q_4		



Non-deterministic finite automata (NFA)

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- Initial state q_0
- $T = \{q_2, q_3, q_4\}$
- $\Delta : Q \times \Sigma \Rightarrow 2^Q$

Δ	a	b
q_0	$\{q_1, q_4\}$	$\{q_3\}$
q_1	$\{q_1\}$	$\{q_2\}$
q_2	$\varnothing$	$\varnothing$
q_3	$\varnothing$	$\varnothing$
q_4	$\varnothing$	$\{q_4\}$

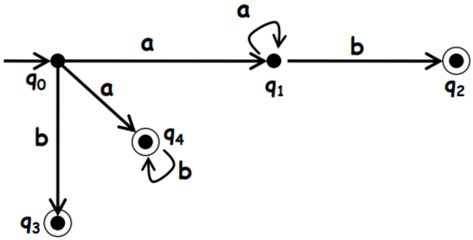


Non-deterministic finite automata (NFA)

Does the string *baa* get accepted or rejected by the automaton?

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- Initial state q_0
- $T = \{q_2, q_3, q_4\}$
- $\Delta : Q \times \Sigma \Rightarrow 2^Q$

Δ	a	b
q_0	$\{q_1, q_4\}$	$\{q_3\}$
q_1	$\{q_1\}$	$\{q_2\}$
q_2	$\varnothing$	$\varnothing$
q_3	$\varnothing$	$\varnothing$
q_4	$\varnothing$	$\{q_4\}$

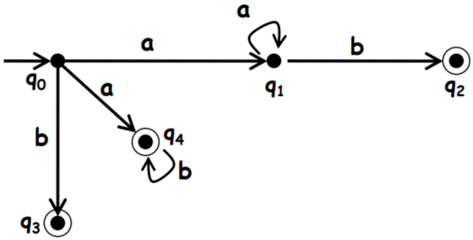


Non-deterministic finite automata (NFA)

If the string is not fully consumed, it is rejected

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- Initial state q_0
- $T = \{q_2, q_3, q_4\}$
- $\Delta : Q \times \Sigma \Rightarrow 2^Q$

Δ	a	b
q_0	$\{q_1, q_4\}$	$\{q_3\}$
q_1	$\{q_1\}$	$\{q_2\}$
q_2	$\varnothing$	$\varnothing$
q_3	$\varnothing$	$\varnothing$
q_4	$\varnothing$	$\{q_4\}$

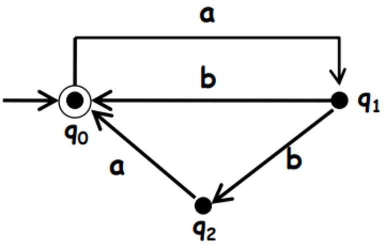


Non-deterministic finite automata (NFA)

Formally represent the NFA

- Σ
- Q
- Initial state
- T
- Δ

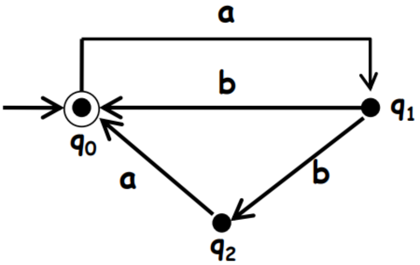
Δ	a	b
q_0		
q_1		
q_2		



Non-deterministic finite automata (NFA)

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2\}$
- Initial state q_0
- $T = \{q_0\}$
- $\Delta : Q \times \Sigma \Rightarrow 2^Q$

Δ	a	b
q_0	$\{q_1\}$	$\varnothing$
q_1	$\varnothing$	$\{q_0, q_2\}$
q_2	$\{q_0\}$	$\varnothing$

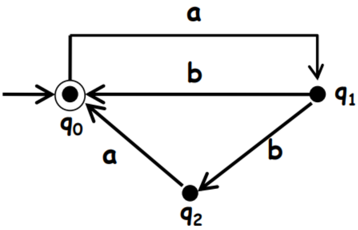


Non-deterministic finite automata (NFA)

NFA that accepts $(ab \cup aba)^*$

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2\}$
- Initial state q_0
- $T = \{q_0\}$
- $\Delta : Q \times \Sigma \Rightarrow 2^Q$

Δ	a	b
q_0	$\{q_1\}$	$\varnothing$
q_1	$\varnothing$	$\{q_0, q_2\}$
q_2	$\{q_0\}$	$\varnothing$



Non-deterministic finite automata (NFA)

Design the NFA specified below and indicate the accepted language:

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2\}$
- Initial state q_0
- $T = \{q_2\}$
- $\Delta : Q \times \Sigma \Rightarrow 2^Q$

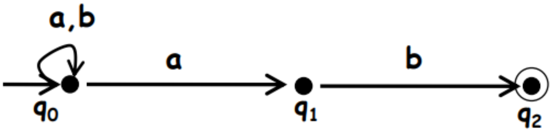
Δ	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_2\}$
q_2	$\varnothing$	$\varnothing$

Non-deterministic finite automata (NFA)

NFA that accepts strings ending in ab. $(a \cup b)^* ab$

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2\}$
- Initial state q_0
- $T = \{q_2\}$
- $\Delta : Q \times \Sigma \Rightarrow 2^Q$

Δ	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\varnothing$	$\{q_2\}$
q_2	$\varnothing$	$\varnothing$



Non-deterministic finite automata (NFA)

Design the NFA specified below and indicate the accepted language:

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- Initial state q_0
- $T = \{q_2, q_4\}$
- $\Delta : Q \times \Sigma \Rightarrow 2^Q$

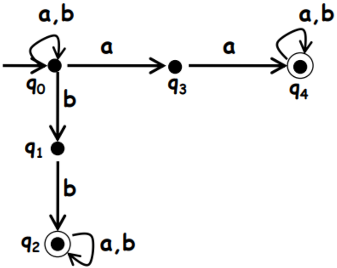
Δ	a	b
q_0	$\{q_0, q_3\}$	$\{q_0, q_1\}$
q_1	$\varnothing$	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$
q_3	$\{q_4\}$	$\varnothing$
q_4	$\{q_4\}$	$\{q_4\}$

Non-deterministic finite automata (NFA)

NFA that accepts $(a \cup b)^*(aa \cup bb)(a \cup b)^*$

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2, q_3, q_4\}$
- Initial state q_0
- $T = \{q_2, q_4\}$
- $\Delta : Q \times \Sigma \Rightarrow 2^Q$

Δ	a	b
q_0	$\{q_0, q_3\}$	$\{q_0, q_1\}$
q_1	$\varnothing$	$\{q_2\}$
q_2	$\{q_2\}$	$\{q_2\}$
q_3	$\{q_4\}$	$\varnothing$
q_4	$\{q_4\}$	$\{q_4\}$



Non-deterministic finite automata (NFA)

Design an NFA over $\Sigma = \{a, b\}$ that recognizes the language of all strings ending in b given by the regular expression $(a \cup b)^* b$

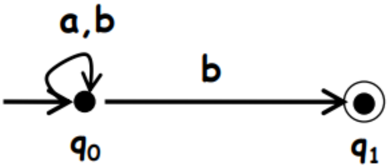
- Show the transition diagram
- Express the automaton formally

Non-deterministic finite automata (NFA)

NFA that accepts $(a \cup b)^*b$

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1\}$
- Initial state q_0
- $T = \{q_1\}$
- $\Delta : Q \times \Sigma \Rightarrow 2^Q$

Δ	a	b
q_0	$\{q_0\}$	$\{q_0, q_1\}$
q_1	$\varnothing$	$\varnothing$



Non-deterministic finite automata (NFA)

Design an NFA over $\Sigma = \{a, b\}$ that recognizes the language of all strings that have at least two consecutive a's given by the regular expression $(a \cup b)^* aa(a \cup b)^*$

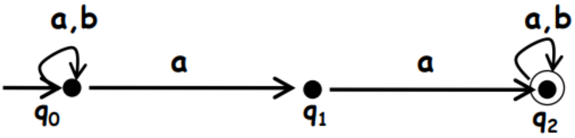
- Show the transition diagram
- Express the automaton formally

Non-deterministic finite automata (NFA)

NFA that accepts $(a \cup b)^* aa(a \cup b)^*$

- $\Sigma = \{a, b\}$
- $Q = \{q_0, q_1, q_2\}$
- Initial state q_0
- $T = \{q_2\}$
- $\Delta : Q \times \Sigma \Rightarrow 2^Q$

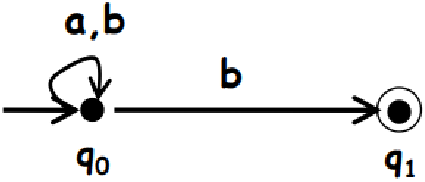
Δ	a	b
q_0	$\{q_0, q_1\}$	$\{q_0\}$
q_1	$\{q_2\}$	$\varnothing$
q_2	$\{q_2\}$	$\{q_2\}$



Contenido

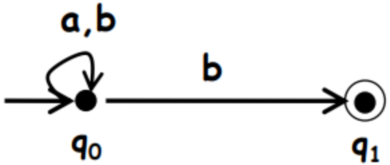
Equivalence between DFA and NFA

Consider the NFA that recognizes the language of words over $\Sigma = \{a, b\}$ that end in b

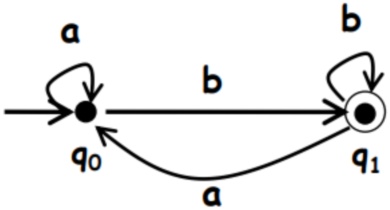


Equivalence between DFA and NFA

NFA that recognizes the language of words over $\Sigma = \{a, b\}$ that end in b



DFA that recognizes the same language

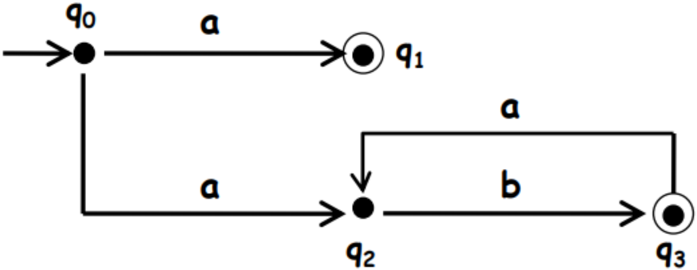


Equivalence between DFA and NFA

Every NFA M' has a DFA M such that $L(M') = L(M)$

Contenido

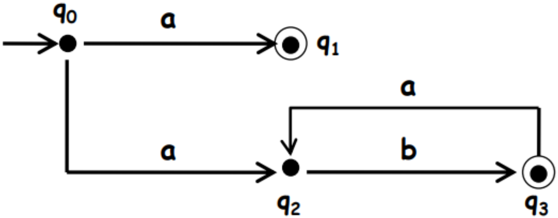
Method to convert an NFA to a DFA



NFA that accepts $a \cup (ab)^+$

Method to convert an NFA to a DFA

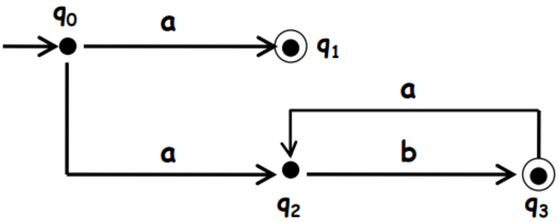
NFA that accepts $a \cup (ab)^+$



- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$

Method to convert an NFA to a DFA

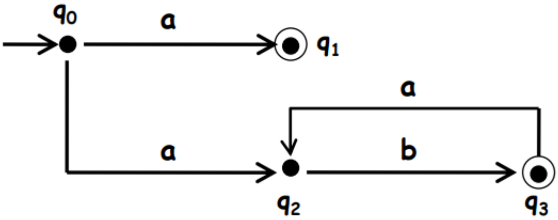
NFA that accepts $a \cup (ab)^+$



- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$
- $\Delta(\{q_1, q_2\}, a) = ?$
- $\Delta(\{q_1, q_2\}, b) = ?$

Method to convert an NFA to a DFA

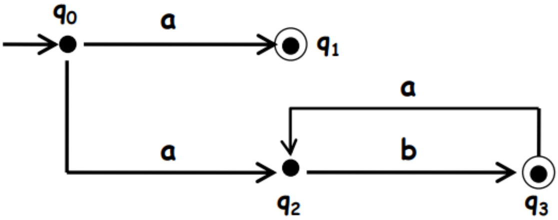
NFA that accepts $a \cup (ab)^+$



- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$
- $\Delta(\{q_1, q_2\}, a) = \varnothing$
- $\Delta(\{q_1, q_2\}, b) = \{q_3\}$

Method to convert an NFA to a DFA

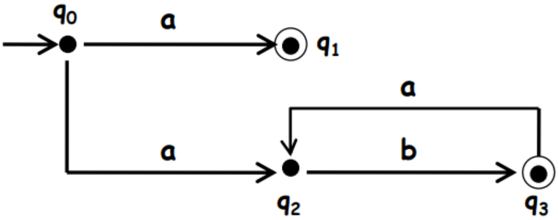
NFA that accepts $a \cup (ab)^+$



- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$
- $\Delta(\{q_1, q_2\}, a) = \varnothing$
- $\Delta(\{q_1, q_2\}, b) = \{q_3\}$
- $\Delta(\{q_3\}, a) = ?$
- $\Delta(\{q_3\}, b) = ?$

Method to convert an NFA to a DFA

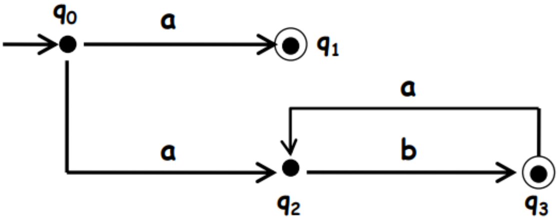
NFA that accepts $a \cup (ab)^+$



- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$
- $\Delta(\{q_1, q_2\}, a) = \varnothing$
- $\Delta(\{q_1, q_2\}, b) = \{q_3\}$
- $\Delta(\{q_3\}, a) = \{q_2\}$
- $\Delta(\{q_3\}, b) = \varnothing$

Method to convert an NFA to a DFA

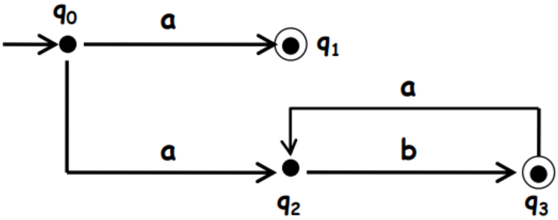
NFA that accepts $a \cup (ab)^+$



- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$
- $\Delta(\{q_1, q_2\}, a) = \varnothing$
- $\Delta(\{q_1, q_2\}, b) = \{q_3\}$
- $\Delta(\{q_3\}, a) = \{q_2\}$
- $\Delta(\{q_3\}, b) = \varnothing$
- $\Delta(\{q_2\}, a) = ?$
- $\Delta(\{q_2\}, b) = ?$

Method to convert an NFA to a DFA

NFA that accepts $a \cup (ab)^+$



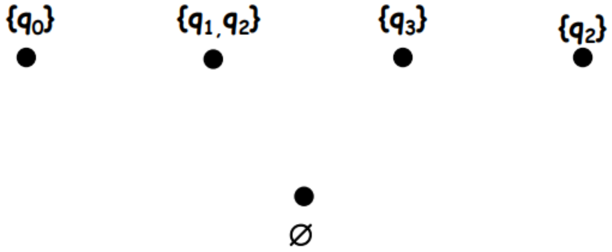
- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$
- $\Delta(\{q_1, q_2\}, a) = \varnothing$
- $\Delta(\{q_1, q_2\}, b) = \{q_3\}$
- $\Delta(\{q_3\}, a) = \{q_2\}$
- $\Delta(\{q_3\}, b) = \varnothing$
- $\Delta(\{q_2\}, a) = \varnothing$
- $\Delta(\{q_2\}, b) = \{q_3\}$

Method to convert an NFA to a DFA

- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$
- $\Delta(\{q_1, q_2\}, a) = \varnothing$
- $\Delta(\{q_1, q_2\}, b) = \{q_3\}$
- $\Delta(\{q_3\}, a) = \{q_2\}$
- $\Delta(\{q_3\}, b) = \varnothing$
- $\Delta(\{q_2\}, a) = \varnothing$
- $\Delta(\{q_2\}, b) = \{q_3\}$

Method to convert an NFA to a DFA

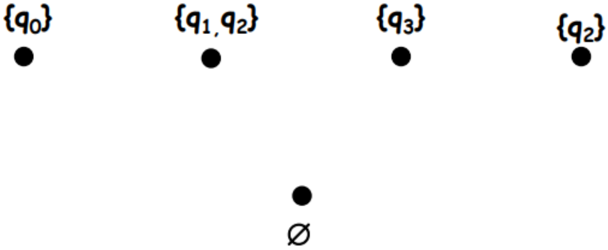
- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$
- $\Delta(\{q_1, q_2\}, a) = \varnothing$
- $\Delta(\{q_1, q_2\}, b) = \{q_3\}$
- $\Delta(\{q_3\}, a) = \{q_2\}$
- $\Delta(\{q_3\}, b) = \varnothing$
- $\Delta(\{q_2\}, a) = \varnothing$
- $\Delta(\{q_2\}, b) = \{q_3\}$



Method to convert an NFA to a DFA

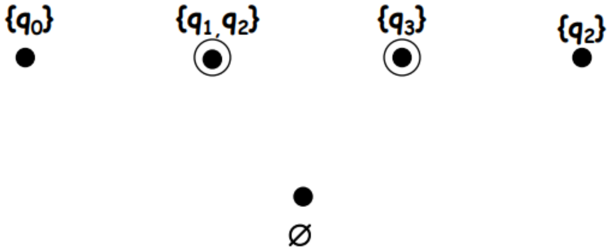
Any set containing an accepting state is marked as accepting

- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$
- $\Delta(\{q_1, q_2\}, a) = \varnothing$
- $\Delta(\{q_1, q_2\}, b) = \{q_3\}$
- $\Delta(\{q_3\}, a) = \{q_2\}$
- $\Delta(\{q_3\}, b) = \varnothing$
- $\Delta(\{q_2\}, a) = \varnothing$
- $\Delta(\{q_2\}, b) = \{q_3\}$



Method to convert an NFA to a DFA

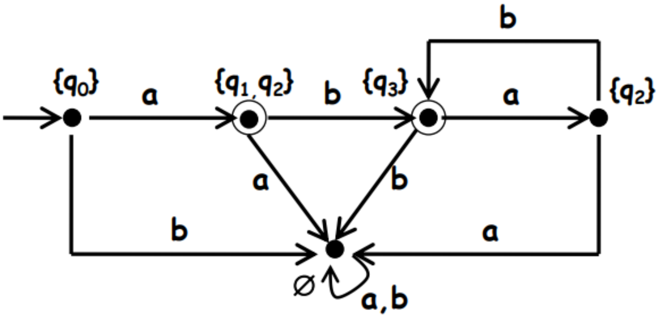
- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$
- $\Delta(\{q_1, q_2\}, a) = \varnothing$
- $\Delta(\{q_1, q_2\}, b) = \{q_3\}$
- $\Delta(\{q_3\}, a) = \{q_2\}$
- $\Delta(\{q_3\}, b) = \varnothing$
- $\Delta(\{q_2\}, a) = \varnothing$
- $\Delta(\{q_2\}, b) = \{q_3\}$



Method to convert an NFA to a DFA

The node labeled $\varnothing$ has transitions leading to itself

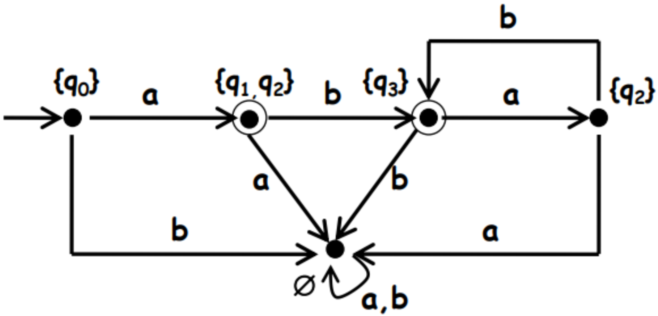
- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$
- $\Delta(\{q_1, q_2\}, a) = \varnothing$
- $\Delta(\{q_1, q_2\}, b) = \{q_3\}$
- $\Delta(\{q_3\}, a) = \{q_2\}$
- $\Delta(\{q_3\}, b) = \varnothing$
- $\Delta(\{q_2\}, a) = \varnothing$
- $\Delta(\{q_2\}, b) = \{q_3\}$



Method to convert an NFA to a DFA

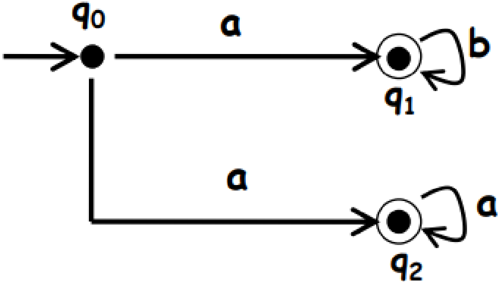
DFA that accepts $a \cup (ab)^+$

- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$
- $\Delta(\{q_1, q_2\}, a) = \varnothing$
- $\Delta(\{q_1, q_2\}, b) = \{q_3\}$
- $\Delta(\{q_3\}, a) = \{q_2\}$
- $\Delta(\{q_3\}, b) = \varnothing$
- $\Delta(\{q_2\}, a) = \varnothing$
- $\Delta(\{q_2\}, b) = \{q_3\}$



Method to convert an NFA to a DFA

Convert the following NFA to a DFA



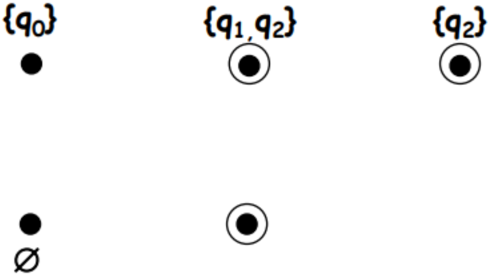
NFA that accepts $ab^* \cup a^+$

Method to convert an NFA to a DFA

- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$
- $\Delta(\{q_1, q_2\}, a) = \{q_2\}$
- $\Delta(\{q_1, q_2\}, b) = \{q_1\}$
- $\Delta(\{q_2\}, a) = \{q_2\}$
- $\Delta(\{q_2\}, b) = \varnothing$
- $\Delta(\{q_1\}, a) = \varnothing$
- $\Delta(\{q_1\}, b) = \{q_1\}$

Method to convert an NFA to a DFA

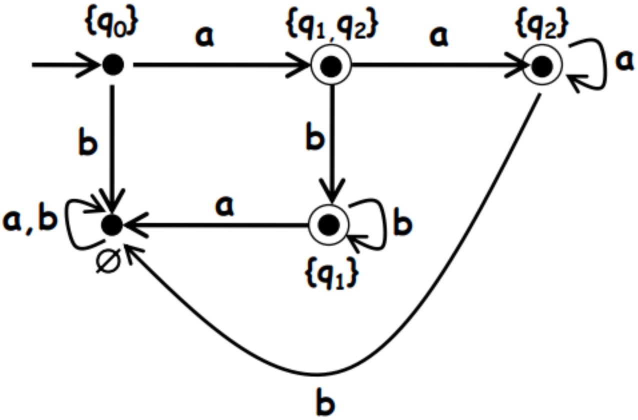
- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$
- $\Delta(\{q_1, q_2\}, a) = \{q_2\}$
- $\Delta(\{q_1, q_2\}, b) = \{q_1\}$
- $\Delta(\{q_2\}, a) = \{q_2\}$
- $\Delta(\{q_2\}, b) = \varnothing$
- $\Delta(\{q_1\}, a) = \varnothing$
- $\Delta(\{q_1\}, b) = \{q_1\}$



Method to convert an NFA to a DFA

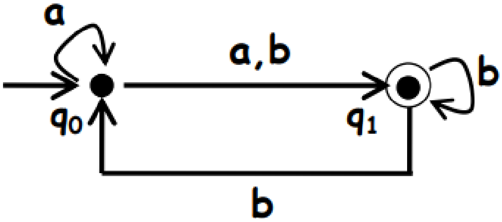
DFA that accepts $ab^* \cup a^+$

- $\Delta(q_0, a) = \{q_1, q_2\}$
- $\Delta(q_0, b) = \varnothing$
- $\Delta(\{q_1, q_2\}, a) = \{q_2\}$
- $\Delta(\{q_1, q_2\}, b) = \{q_1\}$
- $\Delta(\{q_2\}, a) = \{q_2\}$
- $\Delta(\{q_2\}, b) = \varnothing$
- $\Delta(\{q_1\}, a) = \varnothing$
- $\Delta(\{q_1\}, b) = \{q_1\}$



Method to convert an NFA to a DFA

Convert the following NFA to a DFA

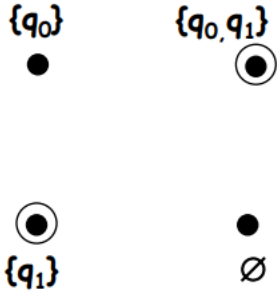


Method to convert an NFA to a DFA

- $\Delta(q_0, a) = \{q_0, q_1\}$
- $\Delta(q_0, b) = \{q_1\}$
- $\Delta(\{q_0, q_1\}, a) = \{q_0, q_1\}$
- $\Delta(\{q_0, q_1\}, b) = \{q_0, q_1\}$
- $\Delta(\{q_1\}, a) = \varnothing$
- $\Delta(\{q_1\}, b) = \{q_0, q_1\}$

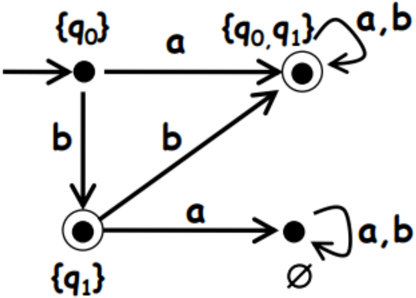
Method to convert an NFA to a DFA

- $\Delta(q_0, a) = \{q_0, q_1\}$
- $\Delta(q_0, b) = \{q_1\}$
- $\Delta(\{q_0, q_1\}, a) = \{q_0, q_1\}$
- $\Delta(\{q_0, q_1\}, b) = \{q_0, q_1\}$
- $\Delta(\{q_1\}, a) = \varnothing$
- $\Delta(\{q_1\}, b) = \{q_0, q_1\}$



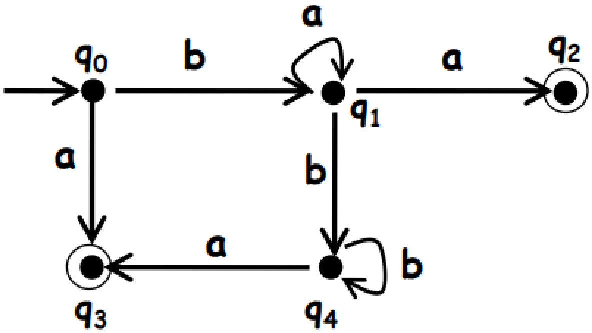
Method to convert an NFA to a DFA

- $\Delta(q_0, a) = \{q_0, q_1\}$
- $\Delta(q_0, b) = \{q_1\}$
- $\Delta(\{q_0, q_1\}, a) = \{q_0, q_1\}$
- $\Delta(\{q_0, q_1\}, b) = \{q_0, q_1\}$
- $\Delta(\{q_1\}, a) = \varnothing$
- $\Delta(\{q_1\}, b) = \{q_0, q_1\}$



Method to convert an NFA to a DFA

Convert the following NFA to a DFA

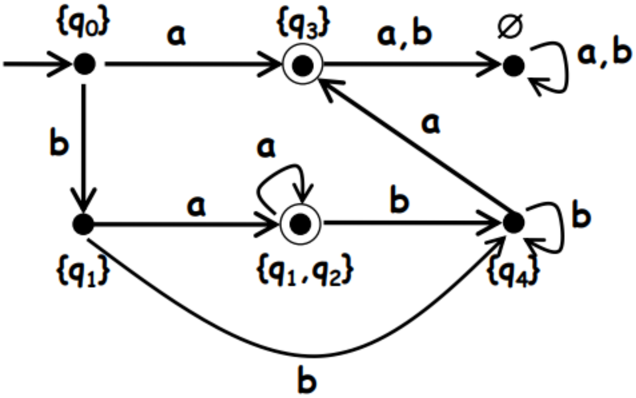


Method to convert an NFA to a DFA

- $\Delta(q_0, a) = \{q_3\}$
- $\Delta(q_0, b) = \{q_1\}$
- $\Delta(\{q_3\}, a) = \varnothing$
- $\Delta(\{q_3\}, b) = \varnothing$
- $\Delta(\{q_1\}, a) = \{q_1, q_2\}$
- $\Delta(\{q_1\}, b) = \{q_4\}$
- $\Delta(\{q_1, q_2\}, a) = \{q_1, q_2\}$
- $\Delta(\{q_1, q_2\}, b) = \{q_4\}$
- $\Delta(\{q_4\}, a) = \{q_3\}$
- $\Delta(\{q_4\}, b) = \{q_4\}$

Method to convert an NFA to a DFA

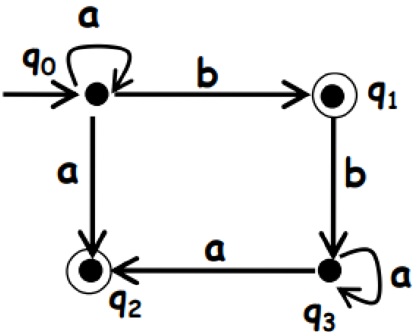
- $\Delta(q_0, a) = \{q_3\}$
- $\Delta(q_0, b) = \{q_1\}$
- $\Delta(\{q_3\}, a) = \varnothing$
- $\Delta(\{q_3\}, b) = \varnothing$
- $\Delta(\{q_1\}, a) = \{q_1, q_2\}$
- $\Delta(\{q_1\}, b) = \{q_4\}$
- $\Delta(\{q_1, q_2\}, a) = \{q_1, q_2\}$
- $\Delta(\{q_1, q_2\}, b) = \{q_4\}$
- $\Delta(\{q_4\}, a) = \{q_3\}$
- $\Delta(\{q_4\}, b) = \{q_4\}$



Regular Languages and Regular Expressions (Copy)

Method to convert an NFA to a DFA

* Convert the following NFA to a DFA



References

Kozen, D. C. (2007)
Automata and computability
Springer Science & Business Media. Lectures 3–5.