Deterministic Finite Automata

Formal definition of DFA: $M = (\boxed{Q}, \Sigma, \delta, q_0, \boxed{F})$

Finite set of states

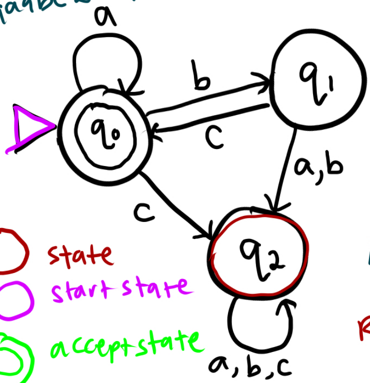
Alphabet

Transition function
2 inputs: a state and a symbol

Start state
$q_0 \in Q$

Set of accept states
$F \subseteq Q$

aaabcbcaaaa



$Q = \{q_0, q_1, q_2\}$

$\Sigma = \{a, b, c\}$

q_0

$F = \{q_0\}$

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

symbols / states

Accepted:
bc, a, bcbc, aaabcbcaaa

Rejected:
ca, bcbb, aaca

$\boxed{(a^* \cup bc)^*}$

A regular expression whose language is accepted by the machine: $\boxed{(a \cup bc)^*}$

state

start state

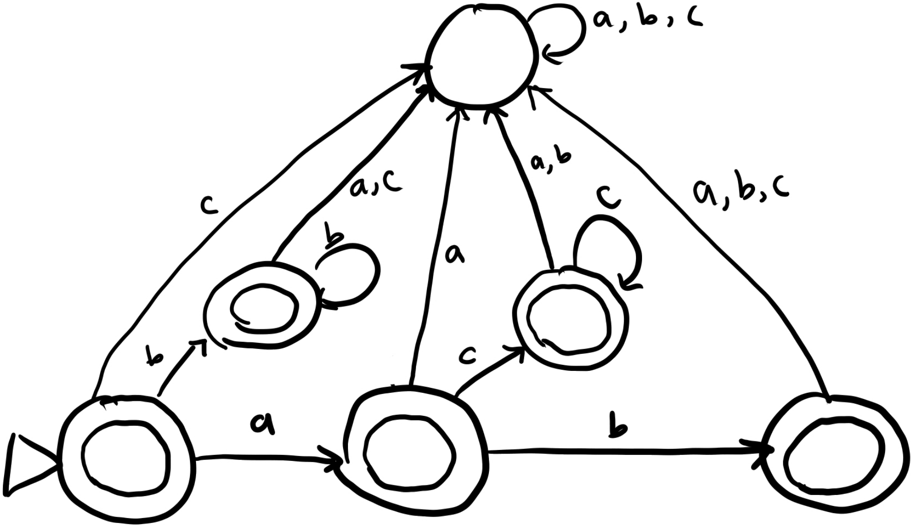
accept state

symbols transition

DFA that recognizes $L(ab \cup ac^* \cup b^*)$:

Examples of strings in the language: ab, a, b, ε, acc

Examples of strings not in the language: abc, cbab, c, acb



Nondeterministic Finite Automata

Formal definition of NFA: $M = (\boxed{Q}, \Sigma, \delta, q_0, \boxed{F})$

Finite set of states

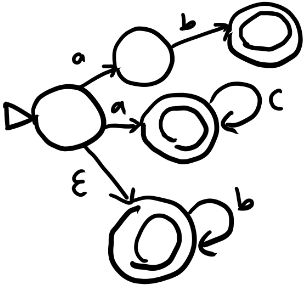
Alphabet

Transition function
2 inputs: state and symbol
output: set of states

Start state

Set of accept states

NFA that recognizes $L(ab \cup ac^* \cup b^*)$



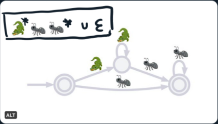
Key Differences from DFA:

- each state does not have to have exactly 1 outgoing transition for each symbol
- spontaneous transitions labeled with ε
- multiple paths of computation

← Tweet

vaguely reassuring state machines
@happyautomata

🐊🐜



11:12 AM · Apr 10, 2022 · libfsm

2 Likes