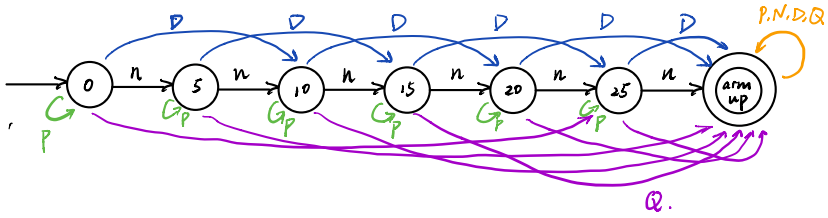
Finite State Machines

Motivating examples



Example 1: toll booth

30 ¢ toll. nickles, dimes, quarters accepted, pennies are accepted but ignored. no changes is given

S = {vertices labeled 0, 5, 15, 20, 25, arm up}.

Σ = {coin (i.e penny, nickle, dime, quarter)}.

s_0 = 0

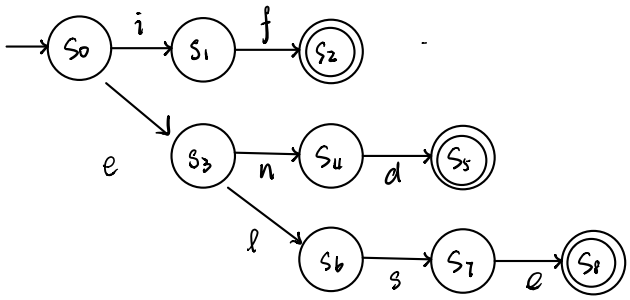
A = {arm up.}

δ = represented by graphical depiction.

	P	N	D	Q
0	0	5	10	25
5	5	10	15	arm up
10	10	15	20	arm up
15	15	20	25	arm up
20	20	25	arm up	arm up
25	25	arm up	arm up	arm up.
arm up	arm up	arm up	arm up	arm up

Example 2: language recognition

FSM that recognizes the string "if", "end", "else"



Consider input

"if" - accept

"input" - stuck.

"endif" - stuck.

"els" - not in accepting state. but entire input is consumed.

S = $\{s_0, s_1, s_2 \cdots, s_8\}$

Σ = $\{a, b, c \cdots, z\}$

s_0 = start state

A = $\{s_2, s_5, s_8\}$

L(*M*) = {if, end, else}

How a finite state machine works

Given a finite input string of characters

curr ← start state s_0
ch ← current input character (symbol)
repeat
  if ∃ edge out of curr labled with ch going to next edge.
    curr ← next
    ch ← next character of input
  otherwise
    stuck

until stuck or the entire input string is consumed.

String is **accepted** if entire string is consumed and be ended up in an accepted state.

otherwise string is **rejected**

Definitions

alphabet (Σ) = finite, non-empty set of elements called **symbols**

string over Σ = finite sequence of symbols from Σ

λ (or ε) = empty string = empty sequence of symbols.
|x| = length of string x.

language over Σ = set of strings over Σ ← not finite.

finite state machine *M* = (*S*, Σ, δ, s_0, *A*) where aka finite state automaton.

S = set of states – finite

Σ = alphabet – finite.

δ = state transition function S×Σ→S – given a state & a symbol, returns a state.

s_0 = start state – only 1, $s_0 \in S$

A = set of accepting (or final) states – must be a subset of S, i.e. $A \subseteq S$.

L(*M*) = language defined by FSM m.
↳ or accepted / recognized / decided

finite automata *M* **accepts** $x = x_1x_2x_3...x_n$ iff

$\delta(\delta(\ldots\delta(\delta(\delta(s_0, x_1), x_2), x_3)\ldots, x_{n-1}), x_n) \in A$

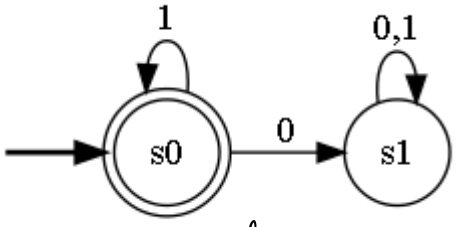
Two error situations:

1. $x_1, x_2, x_3 \ldots x_j$ is not valid prefix for any string in L(m) (i.e get stuck).
2. after processing x_m. FSM is not an accepting state.

Finite State Machines (continued)

Finite Automata Examples

What language is recognized by the following finite automata?



$\Sigma = \{0, 1\}$. $A = \{s_0\}$.

$1^k = \underbrace{111\cdots1}_{k \text{ times}}$

$L(m) = \{$ string of the form 1^n where $n \in \mathbb{N}\}$.

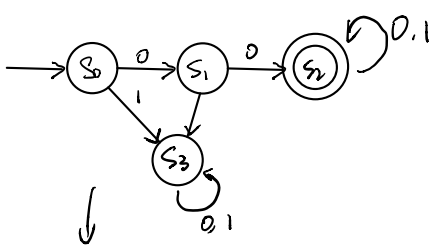
string accepted: λ (empty), 0, 1, 00, 01, 10, 11, 101.

✓ ✗ ✓ ✗ ✗ ✗ ✓ ✗

Construct a DFA that recognizes each of these languages:

the set of bit strings that begin with two consecutive 0s

L_2 $\Sigma = \{0, 1\}$.



s_0 = no 0

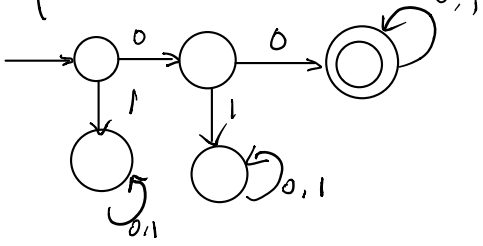
s_1 = 1 0

s_2 = 2 0s consecutive seen.

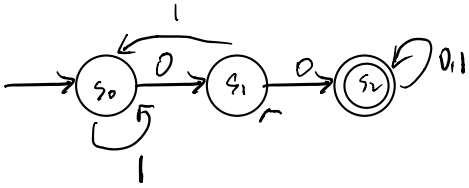
s_3 = invalid

equivalent.

Two FSMs are equivalent iff they recognize the same language.



the set of bit strings that contain two consecutive 0s



the set of bit strings that end with two 0s

Deterministic vs non-deterministic

deterministic = no state has >1 edge with same label & all edges are labeled with elements of Σ (i.e. λ is not allowed).
(DFA)

You will only be required to construct DFAs.

non-deterministic = states can have >1 edge with same label and edges can be labeled λ.
(NFA)