Harvard CS 121 and CSCI E-207
Lecture 3: Finite Automata

Salil Vadhan

September 11, 2012

Reading: Sipser, §1.1 and §1.2.

(Deterministic) Finite Automata

Example: Home Stereo

- P = power button (ON/OFF)
- S = source button (CD/Radio/TV), only works when stereo is ON, but source remembered when stereo is OFF.
- Starts OFF, in CD mode.
- A computational problem: does a given a sequence of button presses $w \in \{P, S\}^*$ leave the system with the radio on?

The Home Stereo DFA

Formal Definition of a DFA

- A DFA M is a 5-Tuple $(Q, \Sigma, \delta, q_0, F)$

  Q : Finite set of <u>states</u>
  Σ : Alphabet
  δ : "Transition function", Q x $\Sigma \to Q$
  q_0 : Start state, $q_0 \in Q$
  F : Accept (or final) states, $F \subseteq Q$

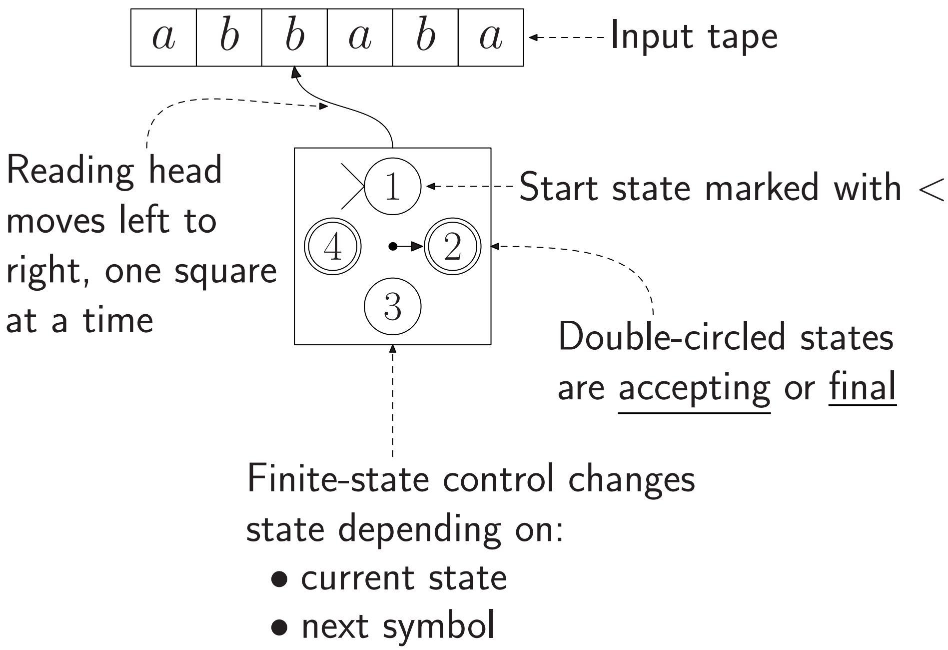
- If $\delta(p, \sigma) = q$,

  then if M is in state p and reads symbol $\sigma \in \Sigma$

  then M enters state q (while moving to next input symbol)

- Home Stereo example:

Another Visualization



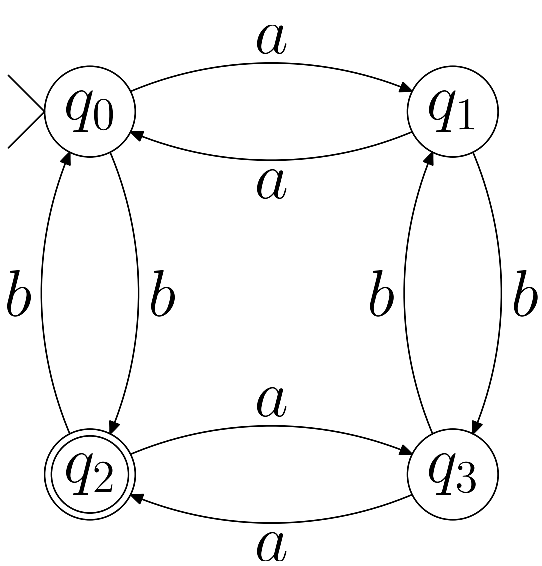
M accepts string x if

- After starting M in the start[initial] state with head on first square,
- when all of x has been read,
- M winds up in a final state.

Examples

- Bounded Counting: A DFA for

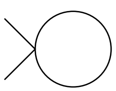
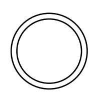
  $\{x : x$ has an even # of a's and an odd # of b's$\}$



Transition function δ:

	a	b
q_0	q_1	q_2
q_1	q_0	q_3
q_2	q_3	q_0
q_3	q_2	q_1.

i.e. $\delta(q_0, a) = q_1$, etc.

 = start state  = final state

$$Q = \{q_0, q_1, q_2, q_3\} \quad \Sigma = \{a, b\} \quad F = \{q_2\}$$

Another Example, to work out together

- Pattern Recognition: A DFA that accepts $\{ x : x$ has aab as a substring$\}$.

Another Example

- Pattern Recognition: A DFA that accepts $\{\ x : x$ has $ababa$ as a substring$\}$.

Another Example

- A DFA that accepts $\{\ x : x$ has $ababa$ as a substring$\}$.

You are going through a constructive process

string $\rightarrow$ DFA

that is automated in every text editor!

Really a compiler that generates DFA code from an input string pattern

Formal Definition of Computation

$M = (Q, \Sigma, \delta, q_0, F)$ accepts $w = w_1 w_2 \cdots w_n \in \Sigma^*$ (where each $w_i \in \Sigma$) if there exist $r_0, \ldots, r_n \in Q$ such that

1. $r_0 = q_0$,
2. $\delta(r_i, w_{i+1}) = r_{i+1}$ for each $i = 0, \ldots, n-1$, and
3. $r_n \in F$.

The language recognized (or accepted) by M, denoted $L(M)$, is the set of all strings accepted by M.

Example:

Transition function on an entire string

More formal (not necessary for us, but notation sometimes useful):

- Inductively define $\delta^* : Q \times \Sigma^* \to Q$ by $\delta^*(q, \varepsilon) = q$, $\delta^*(q, w\sigma) = \delta(\delta^*(q, w), \sigma)$.
- Intuitively, $\delta^*(q, w) =$ "state reached after starting in q and reading the string w".
- M accepts w if $\delta^*(q_0, w) \in F$.

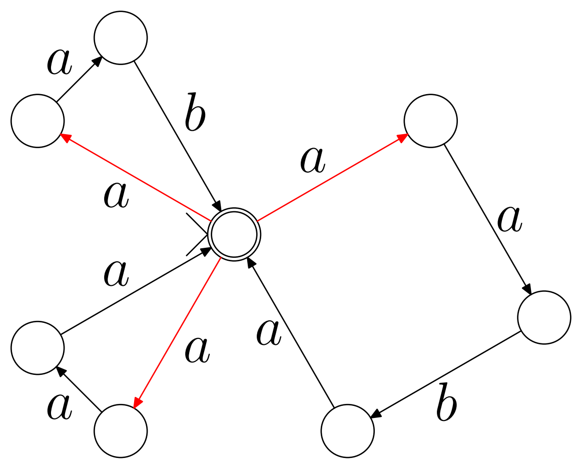
Determinism: Given M and w, the states $r_0, \ldots, r_n$ are uniquely determined. Or in other words, $\delta^*(q, w)$ is well defined for any q and w: There is precisely one state to which w "drives" M if it is started in a given state.

The impulse for nondeterminism

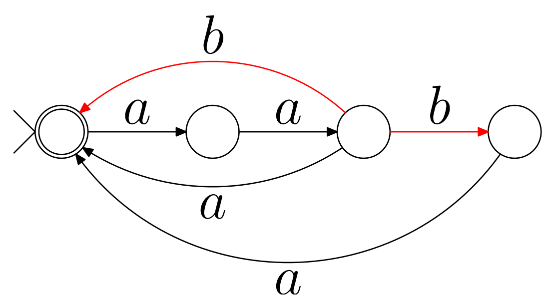
A language for which it is hard to design a DFA:

$$\{x_1 x_2 \cdots x_k : k \geq 0 \text{ and each } x_i \in \{aab, aaba, aaa\}\}.$$

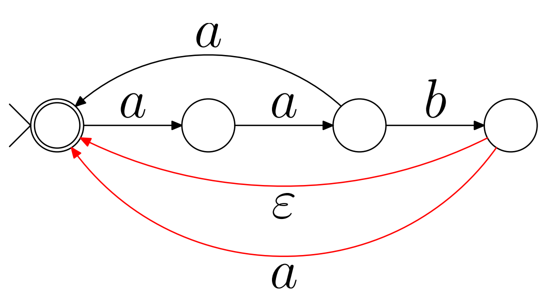
But it is easy to imagine a “device” to accept this language if there sometimes can be several possible transitions!



OR



OR



Nondeterministic Finite Automata

An $\boxed{\text{NFA}}$ is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

- Q, Σ, q_0, F are as for DFAs
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \to P(Q)$.

When in state p reading symbol σ, can go to <u>any</u> state q in the <u>set</u> $\delta(p, \sigma)$.

- there may be more than one such q, or
- there may be none (in case $\delta(p, \sigma) = \emptyset$).

Can "jump" from p to any state in $\delta(p, \varepsilon)$ without moving the input head.

Computations by an NFA

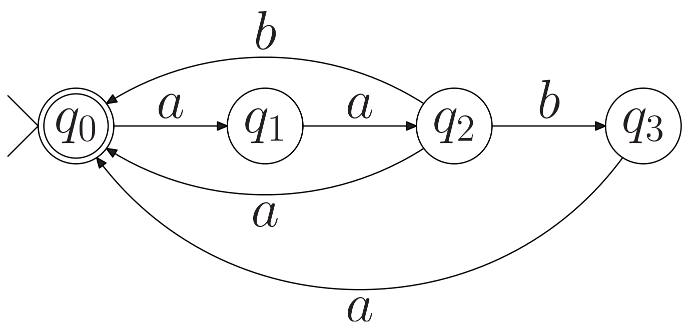
$N = (Q, \Sigma, \delta, q_0, F)$ accepts $w \in \Sigma^*$ if we can write $w = y_1 y_2 \cdots y_m$ where each $y_i \in \Sigma \cup \{\varepsilon\}$ and there exist $r_0, \ldots, r_m \in Q$ such that

1. $r_0 = q_0$,
2. $r_{i+1} \in \delta(r_i, y_{i+1})$ for each $i = 0, \ldots, m-1$, and
3. $r_m \in F$.

Nondeterminism: Given N and w, the states $r_0, \ldots, r_m$ are not necessarily determined.

Example of an NFA

N :



$N = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_0\})$, where δ is given by:

	a	b	ε
q_0	$\{q_1\}$	$\emptyset$	$\emptyset$
q_1	$\{q_2\}$	$\emptyset$	$\emptyset$
q_2	$\{q_0\}$	$\{q_0, q_3\}$	$\emptyset$
q_3	$\{q_0\}$	$\emptyset$	$\emptyset$

Tree of computations

Tree of computations of NFA N on string $aabaab$:

How to simulate NFAs?

- NFA accepts w if there is at least one accepting computational path on input w
- But the number of paths may grow exponentially with the length of w!
- Can exponential search be avoided?

NFAs vs. DFAs

NFAs seem more “powerful” than DFAs. Are they?