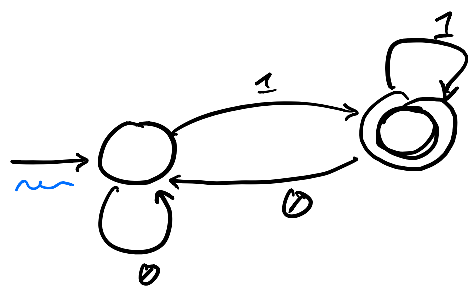
COMS W3261 – Lecture 1 – Part 2

(Deterministic) Finite Automata: YES/NO machine

Idea: Build a machine that decides whether a string is in a language. Consider an input string from left to right, then output YES or NO at the end.

$\Sigma = \{0, 1\}$ Goal: Accept / YES on strings that end in '1.'



test strings: 010 ε

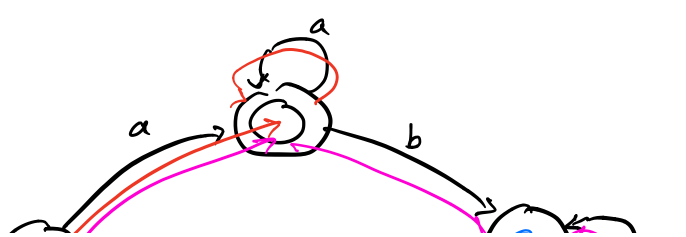
011

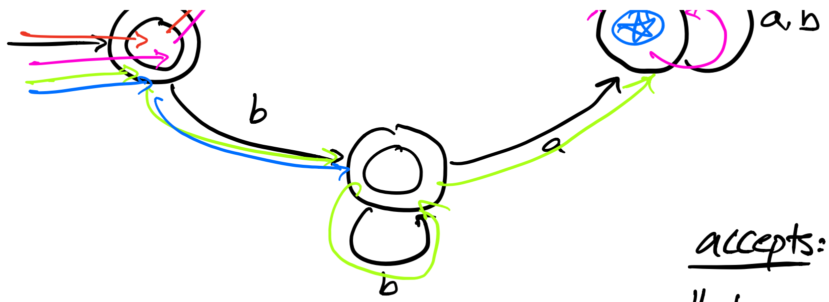
Def: A state diagram contains:

- a start state (mark with an arrow ⟶)
- zero or more accept states (marked by an inner circle ◎)
- a transition (⟶) indicating what to do at every state and for every symbol in our alphabet.

A state diagram accepts a string w iff it is in an accept state after the last symbol is read.

Ex. 2:) $\Sigma = \{a, b\}$





aa, aba, bba, b, ε

✓ ✗ ✗ ✓ ✓

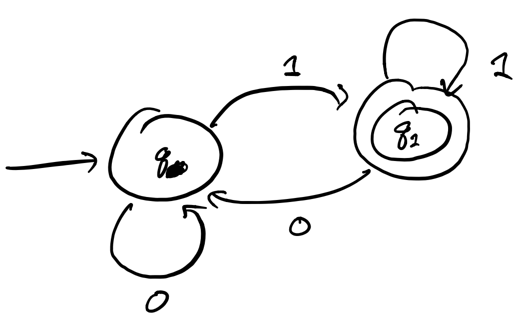
accepts:

all strings of all a,
all b,
ε

Def: A (Deterministic) Finite Automaton (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where:

- Q is a finite set of states
- Σ is a finite alphabet
- δ is a transition function $Q \times \Sigma \to Q$

  given a state, and a symbol, which state do I go to?
- q_0 is a start state
- F is a subset of Q, containing accept states.



$(Q, \Sigma, \delta, q_0, F)$

$Q = \{q_0, q_1\}$

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

$\delta =$

	0	1

$\delta(q_0, 0) = q_0$

q_0	q_0	$0'$
q_1	q_0	q_3

$q_0 = q_0$

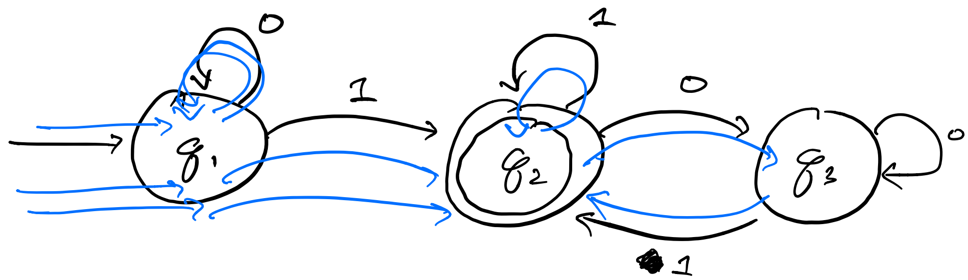
$F = \{q_1\}$

$M_1 = (Q, \Sigma, \delta, q_1, F)$,

$Q = \{q_1, q_2, q_3\}$, $\Sigma = \{0, 1\}$, $F = \{q_2\}$

δ:

	0	1
q_1	q_1	q_2
q_2	q_3	q_2
q_3	q_3	q_2



00	11	0	ε	101
✗	✓	✗	✗	✓

accepts: strings ending in 1.

<u>Def.</u> Let M be a DFA. $L(M)$ is defined to be the set of all strings M accepts, the <u>language</u> of M. We also say "M recognizes A." (M accepts A.)

<u>Def.</u> (Accepting a string) Let $M = (Q, \Sigma, \delta, q_0, F)$ be

a DFA and let $w = w_0 w_1 w_2 \cdots w_{n-1}$. Then, M accepts w if there exists a sequence of states $r_0, r_1, \ldots, r_n \in Q$ satisfying:

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

Zoom out. Languages are sets of strings.
Languages "capture" concepts.
DFA specify a finite procedure for deciding whether or not a string is in a language

$\approx$ decide whether an object is an instance of a concept

complexity of automata required to recognize L $\approx$ complexity of L $\approx$ difficulty of answering the YES/NO question: is $w \in L$?

Def. (Regular language.) A language is called regular if some DFA recognizes it.

(Warning! Not all languages are regular.)

Regular Operations:

Idea: When we code, we re-use code. We compose, self reference, make things modular.

Def. Let A, B be languages over some alphabet Σ.

Union: $A \cup B := \{x \mid x \in A \text{ OR } x \in B\}$

Concatenation: $A \circ B := \{xy \mid x \in A,\ y \in B\}$

(Kleene) Star: $A^* := \{x_1 x_2 \cdots x_k \mid k \geq 0,\ x_i \in A\}$

<u>Example:</u> $A = \{\text{cat}, \text{dog}\}$ $\quad B = \{\text{blue}, \text{red}\}$

$A \cup B = \{\text{cat}, \text{dog}, \text{blue}, \text{red}\}$

$A \circ B = \{\text{catblue}, \text{catred}, \text{dogblue}, \text{dogred}\}$

$A^* = \{\varepsilon, \text{cat}, \text{dog}, \text{catcat}, \text{catdog}, \text{dogdog}, \text{dogcat}, \ldots\}$

<u>Theorem:</u> The class of regular languages is <u>closed</u> under union, concatenation, and star.

If A, B regular:

$A \cup B$ regular

$A \circ B$, $B \circ A$, A^*, B^* all regular

Proof: next time.

Reading: Chapter $\emptyset$ (Review)

'To the Student,' 'To the educator', Sipser Introduction.

Material for today: Chapter 1.1.

Homework: PS #1 — Posted Monday, 6/28/2021.

Due Monday, 7/5/2021 @ 11:59pm EST.