Lecture 2:

From last time:

An alphabet Σ ("sigma") is a non-empty (i.e $\Sigma = \{\}$ isn't allowed) finite set of symbols

Ex: $\Sigma = \{0, 1\}$, $\Sigma = \{a, b, c\}$

A string over Σ is a finite sequence of characters from Σ

Ex: (over $\Sigma = \{0, 1\}$) 0, 00, 1101, 11

ε ← empty string ("epsilon")

A language over Σ is a $L \subseteq \Sigma^*$ ← all strings over Σ ("sigma star")

Ex: (over $\Sigma = \{0, 1\}$)

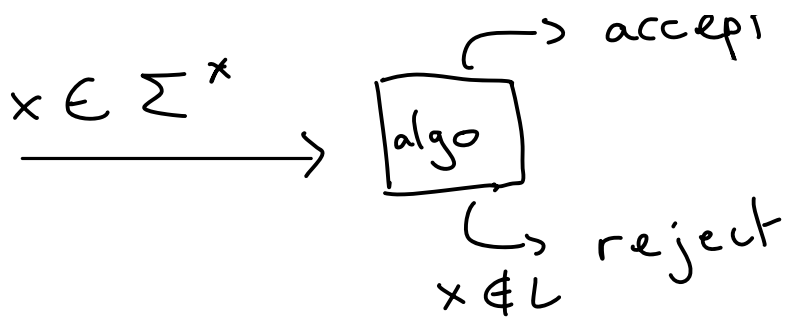
$L = \emptyset$, $L = \{0, 00, 11\}$

$L = \{x \mid x \text{ has an even \# of 0s}\}$

L can be finite or infinite

Languages correspond to problems

$x \in L$



Languages correspond to functions: given $L \subseteq \Sigma^*$ we define $f_L : \Sigma^* \to \{0,1\}$ as

$$f_L(x) = \begin{cases} 1 & \text{if } x \in L \\ 0 & \text{if } x \notin L \end{cases}$$

Today ~ Chapter 1 up to "regular operators" (page 44)

First checkup out later today

- due Tuesday Sept 15th
  - ↑ 1 attempt
- Final attempt : Tuesday Sept 22nd
  - ↑ ∞ attempts

for checkups

=> do the first attempt by yourself!

We drop the 2 lowest checkups

(we know you might sometimes forget)

Tentative quizz schedule:

- Oct 6 (Tue)
- Oct 22 (Thur)
- Nov 17 (Tue)
- Dec 8 (Tue)
- Final is 17th December

should attend your section

If you foresee an issue, let us know ASAP

OHs will start next week!

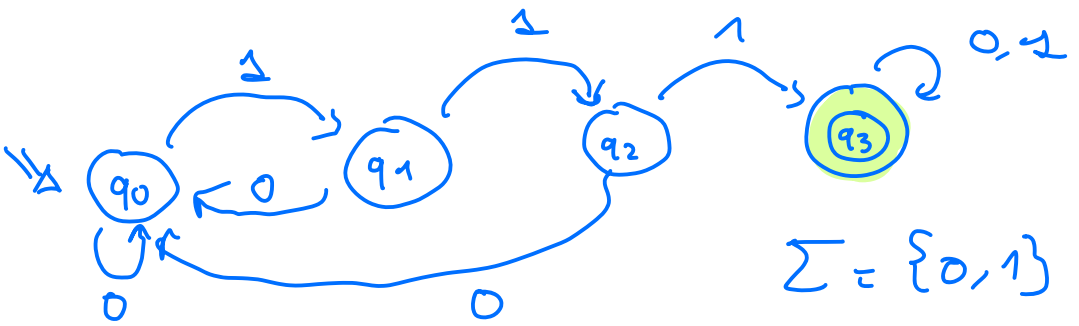
An handout coming soon: a discrete math review

Today: Deterministic finite automaton (DFA)

A DFA is roughly "a computer with very little memory"

A DFA is a machine that takes as input a string x (from a certain alphabet), it reads x from left to right & when it reaches the end of x it accepts or rejects

Example

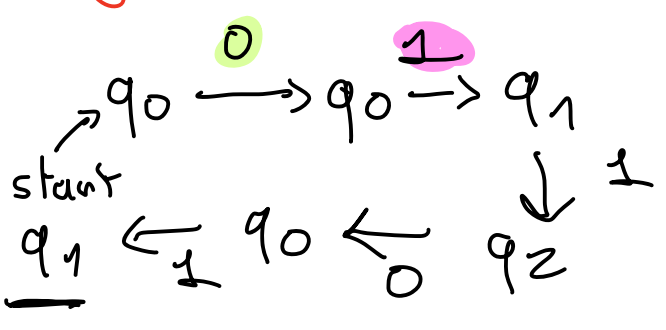


Every DFA has:

- a finite set of states (○)
- For every state and each alphabet symbol we have a transition
- Optionaly some accept states marked as ◎ (there can be 0, 1, 2 or more accept states)

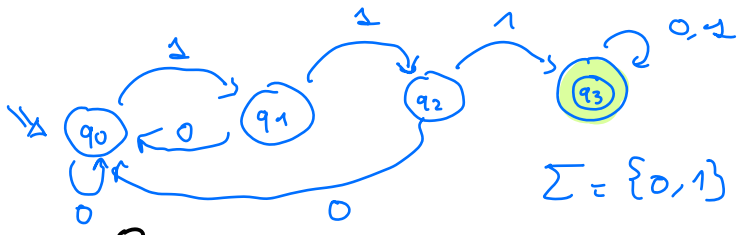
Computation of a DFA on a string:

Input is x = 01101



The DFA rejects x

y = 0 1 1 1 0 1



$\rightarrow q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2 \xrightarrow{1} q_3 \xrightarrow{0} q_3 \xrightarrow{1} q_3$

The DFA accepts y

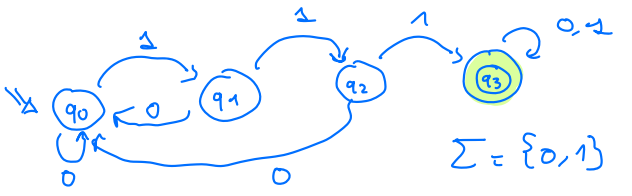
last example $z = \varepsilon \rightarrow q_0$ so ε is rejected by the DFA

Definition: A DFA is a 5-tuple $M = (Q, \Sigma, \delta, q_0, F)$

- Q is a FINITE set of states
- Σ is an alphabet
- q_0 is the name of the start state (a DFA always has exactly 1 start-state)
- $F \subseteq Q$, the set of accept states

"delta" δ is the transition function: $\delta: Q \times \Sigma \rightarrow Q$

δ maps a state + symbol to a "next" state



$Q = \{q_0, q_1, q_2, q_3\}$

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

q_0 is the start state

$F = \{q_3\}$ (~~not $F = q_3$~~)

δ	0	1
q_0	q_0	q_1
q_1	q_0	q_2
q_2	q_0	q_3
q_3	q_3	q_3

Def: the computation of a DFA on a string

- If $M = (Q, \Sigma, \delta, q_0, F)$ is a DFA and $w = w_1 w_2 \cdots w_n$ is an character string, where each $w_i \in \Sigma$, then the computation of M on w is a sequence of states: $r_0, r_1, r_2, \cdots, r_n$ where each $r_i \in Q$

- $r_0 = q_0$ (the start state)
- $\forall i \in \{1, \cdots, n\}$ we have $\delta(r_{i-1}, w_i) = r_i$

If $r_n \in F$ (the last state is an accept state) then we say $(r_0, r_1 \cdots, r_n)$ is an accepting computation

For each DFA M and string w, there is one computation of M on w

=> if we run M on w, we always go through the same sequence of states

M accepts w if and only if the computation of M on w is accepting (If we run M on w, we end in an accept state)

If M doesn't accept w, we say "M rejects w"

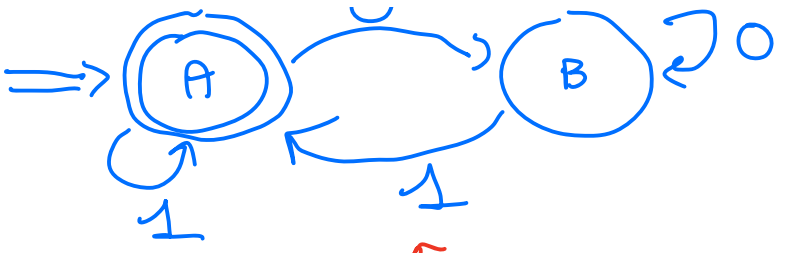
Def: For a DFA M, we define

$$L(M) = \{w \in \Sigma^* \mid M \text{ accepts } w\}$$

this is called the language recognized by M

doesn't accept 0

Example M :

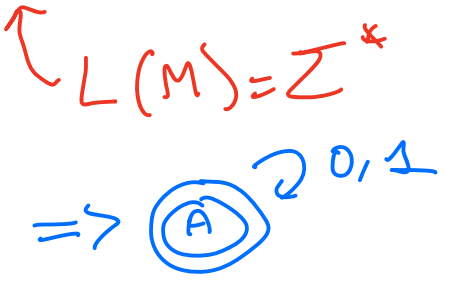


Σ is an alphabet

ex : $\{0,1\}$

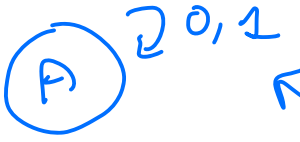
Σ^* is all strings over Σ

$\{0,1\}^* = \{\varepsilon, 0, 1, 01, 10, 00, 11, \ldots\}$

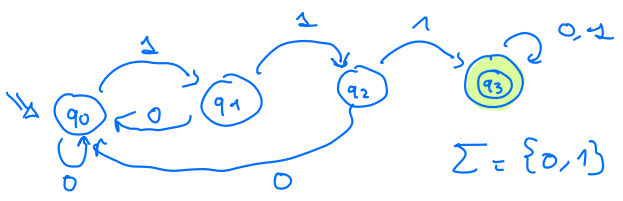


$\Rightarrow L(M) = \Sigma^*$

DFA for $\emptyset$?



$\Rightarrow L(M) = \emptyset$



what's $L(M)$?

$L(M) = \{w \mid w$ has 3 consecutive 1's$\}$

$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2 \xrightarrow{0} q_0$

110

$\Rightarrow$ any string with 3 consecutive 1s is accepted.

Since, from any state of M seeing 111 takes you to q_3. Once in q_3, we stay there, so we'll end in an accept state.

$\Rightarrow$ every other string is rejected.

$\Rightarrow$ whenever we see a 0, if we're not in q_3 we go back to q_0

$\Rightarrow$ only way of going from q_0 to q_3 is

$q_0 \xrightarrow{1} q_1 \xrightarrow{1} q_2 \xrightarrow{1} q$

In general to show $L(M) = L$, need to argue:

- every string in L is accepted by M
- every string not in L is rejected by M

Def: a language L is **REGULAR** if and only if there exists a DFA with $L(M) = L$

⇒ we saw Σ^*, $\emptyset$, $\{w \mid w \text{ contains } 111\}$ are regular

Are all languages regular?
How to prove a language is or isn't regular?

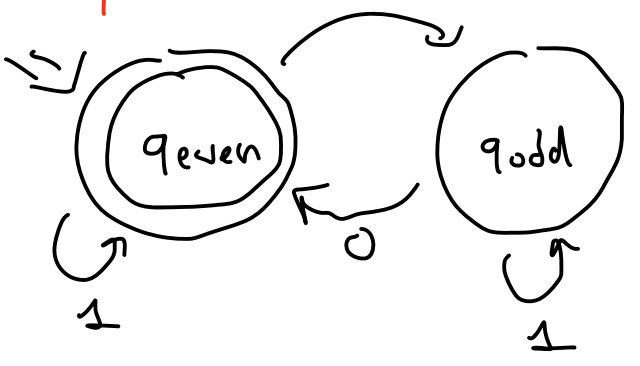
Ex: $L = \{ w \in \{0,1\}^* \mid w \text{ has an even \# of 0s} \}$

- First two algorithms:
  - count #of0s
  - count the length of w − # of 1's

↳ this seems hard

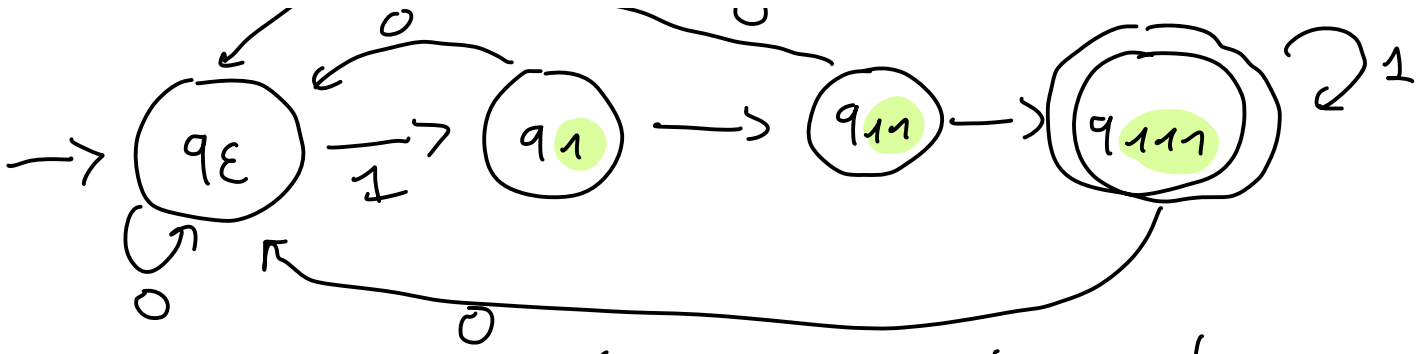
A DFA has a finite # of states, we can't store arbitrarily big number

The one bit algorithm:



A more challenging Language

$L = \{ x \in \{0,1\}^* \mid x \text{ ends in } 111 \}$



we keep track of the longest prefix of 111 we've seen so far

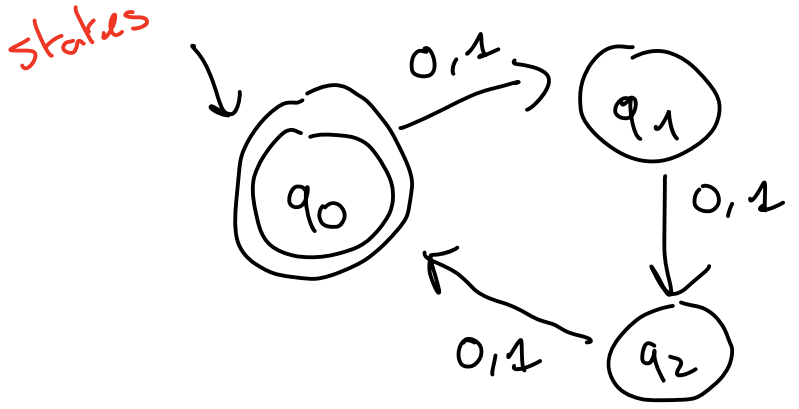
Ex: $L = \{x \in \{0,1\}^* \mid |x| \text{ is divisible by } 3\}$

$\varepsilon \in L$ $\quad$ $0000 \notin L$

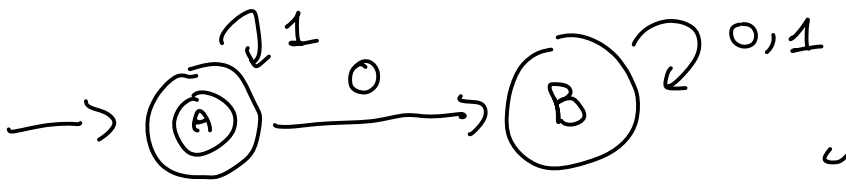
$000 \in L$ $\quad$ $001 \in L$

only need 3 states

$|x| \mod 3$



Ex: $L = \{x \in \{0,1\}^* \mid x \text{ contains no 0s}\}$



$1101 \notin L$

Challenge: is the following regular:

$$L = \{ w \in \{0,1\}^* \mid w = x \circ y \text{ such that}$$

x has an even # of 1s

y has an even # of 0s $\}$