Announcements:

- office Hours start today - see calendar on webpage
  click entry to see location!

  (most OH held in Schermerhorn SLC 3rd floor
  Some in CSB 5th floor, mine CSB 514)

- Reminder: checkup #1 first attempt due tonight

Recap from last time:

Def: A **DFA** is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$

where:

- Q is a finite set of states
- Σ is an alphabet
- $\delta : Q \times \Sigma \to Q$
- $q_0 \in Q$ called "start state"
- $F \subseteq Q$ the set of accepting states

the **computation** of M on $w = w_1 \dots w_n$

is a sequence of states $(r_0, \dots, r_n)$ s.t.

- $r_0 = q_0$
- $\forall i \in \{1, \dots, n\}$ $\delta(r_{i-1}, w_i) = r_i$)

It is accepting if $r_n \in F$

In this case, M accepts w → if the computation ends in an accepting state

(otherwise M rejects w) → if the computation ends in a rejecting state.

Note: • For any DFA M and string $w \in \Sigma^*$ there's one computation of M on w

eg for $w = \varepsilon$ the computation is (q_0).
M accepts ε iff $q_0 \in F$.

The language recognized by DFA M is

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

A language is regular if

there exists a DFA that recognizes it.

(i.e, A is regular $\Leftrightarrow$ $\exists$ DFA M s.t. $L(M) = A$)

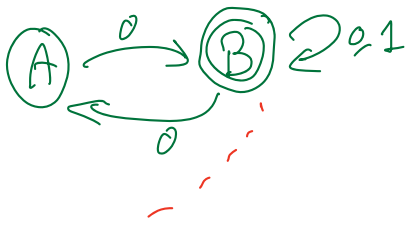
Intuition: you only have a fixed finite memory. Can you use it to process arbitrarily long inputs and determine if they are in A or not?

(need M accepts $w \iff w \in A$)

- A DFA can be specified by listing $(Q, \Sigma, \delta, q_0, F)$ or by a state diagram

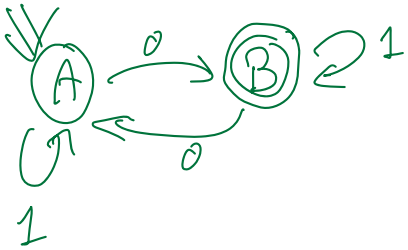
- must have one start state
- every state & symbol must have exactly one transition

NOT A VALID DFA!



- no start state!
- two transitions for (B, 0)
- no transition (A, 1)

M:



valid DFA!

$L(M) = ?$

attempts:

$L(M) \stackrel{?}{=} \{0,1\}$ No! Counter example: $1 \in \{0,1\}$ but not accepted

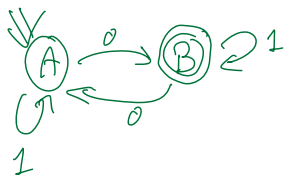
$\stackrel{?}{=} \{w \in \{0,1\}^* \mid w \text{ ends in } 00 \text{ or in } 01\}$

No! Counter example: 001 not accepted.

$\stackrel{\checkmark}{=} \{w \in \{0,1\}^* \mid w \text{ has an odd \# of 0's}\}$

This works!

M:



w with even # of 0's $\Rightarrow$
M's computation ends in state A

w with odd # of 0's $\Rightarrow$
M's computation ends in state B

Last time,
We gave several example languages as exercise
(solved in notes of last lecture)

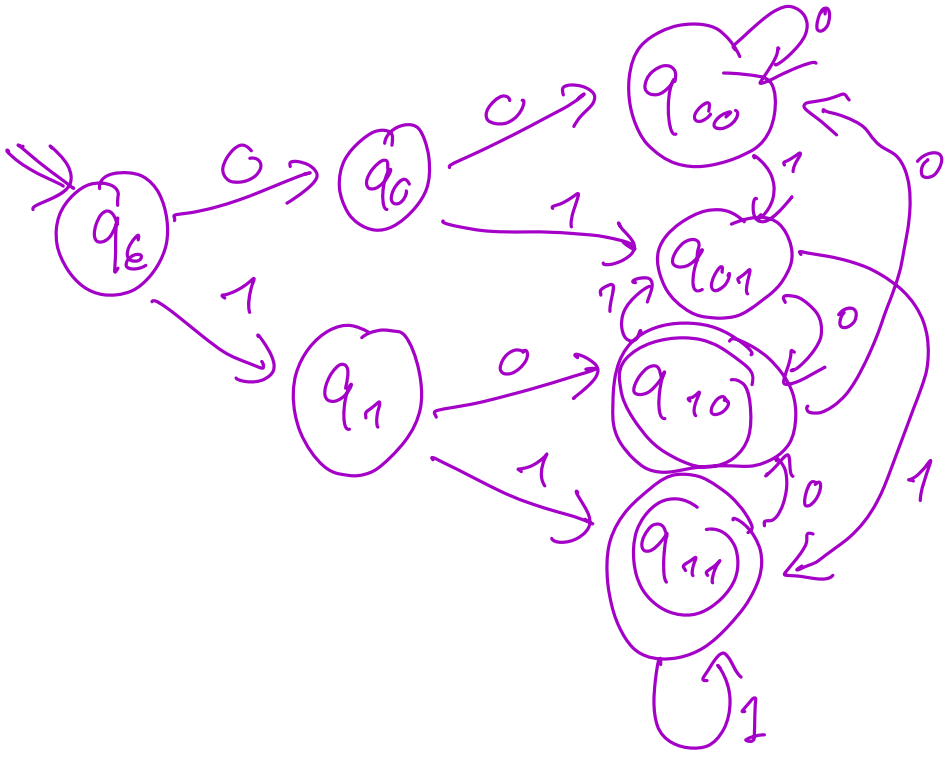
Next: A couple more examples of regular lang.

Example:

$L = \{w \in \{0,1\}^* \mid |w| \geq 2,$ second to last char in w is $1\}$

To show regular, construct DFA.

Idea: keep track of the last two characters you've seen



Example: $L = \{a^i b^j \mid i,j \geq 0\}$

Note: a^i denotes $\underbrace{a \ldots a}_{i \text{ times}}$, a^0 denotes the string ε

$\left(\begin{array}{l} a^0 = \varepsilon \\ a^{i+1} = a^i \cdot a \text{ for } i \geq 0 \end{array}\right)$

so L (=)

$\{a^i b^j \mid i,j \geq 0\} = \{$all strings of the form $\underbrace{a \ldots a}_{\geq 0} \underbrace{b \ldots b}_{\geq 0}\}$

example strings in/not in L:

$a^3b^1 = aaab \in L$

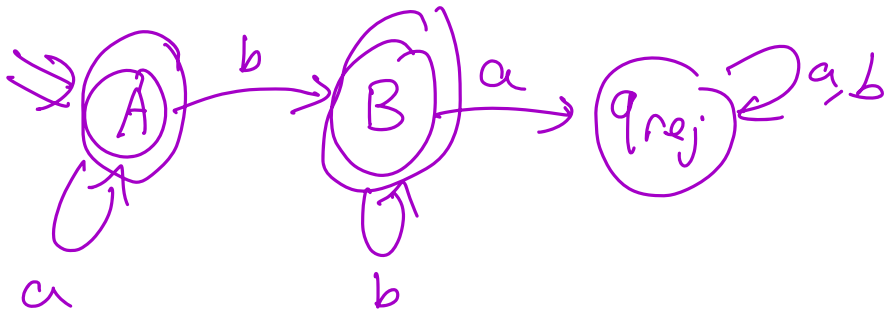
$a^0b^2 = bb \in L$

$bba \notin L$

$a^4b^0 = aaaa \in L$

$a^0b^0 = \varepsilon \in L$

DFA recognizing L:



state A: string so far is a^i $i \geq 0$

state B: string so far is a^ib^j $i \geq 0, j \geq 1$

state q_{rej}: string of wrong format (contains ba)

Next: our first theorem

Thm: All finite languages are regular.

proof sketch: Since L is finite, there exists an integer $m \geq 0$ s.t. all strings $w \in L$ satisfy $|w| \leq m$.
We will construct a DFA M with states

$$Q = \{q_w\}_{|w| \leq m} \cup \{q_{long}\}$$

$\{q_w\}_{|w| \leq m}$: one state for each string of length up to m

q_{long}: one more state for strings longer than m

$$F = \{q_w\}_{w \in L}$$ ⇝ mark as accepting the states that correspond to strings $w \in L$

$$\delta(q_w, a) = \begin{cases} q_{wa} & \text{if } |w| < m \quad \forall a \in \Sigma \\ q_{long} & \text{if } |w| = m \quad \forall a \in \Sigma \end{cases}$$

$$\delta(q_{long}, a) = q_{long}$$

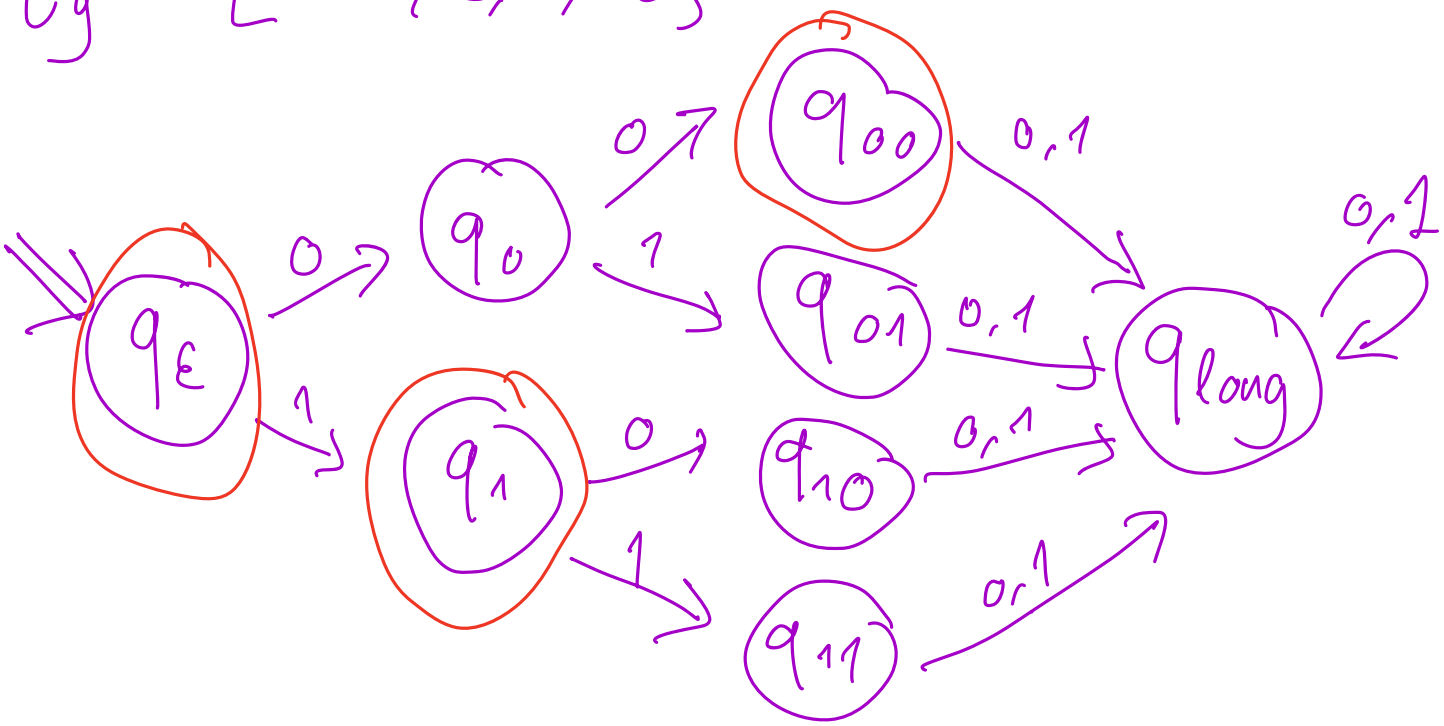
start state: q_ϵ

For any string w, the computation of M on w satisfies:
- if $|w| \leq m$ last state in computation is q_w ⇝ accepting $\Leftrightarrow w \in L$
- if $|w| > m$ last state in computation is q_{long}

Thus, $L(M) = L$ ☒

QED: proof complete.

Eg $L = \{\varepsilon, 1, 00\}$



tree with state for every string of length $\leq m$ (finite number!)

includes a string for each $w \in L$

rejecting state for longer words

We proved every finite language is regular.

How about the converse?

"All regular languages are finite"

Not true! We saw many examples of regular languages that are infinite (eg Σ^*)

(some) Closure properties

for L_1, L_2 languages over Σ we define:

$\overline{L_1} = \Sigma^* \setminus L_1 = \{w \in \Sigma^* \mid w \notin L_1\}$ Complement

$L_1 \cap L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ and } w \in L_2\}$ intersection

$L_1 \cup L_2 = \{w \in \Sigma^* \mid w \in L_1 \text{ or } w \in L_2\}$ union

Thm: The class of regular languages is closed under complement, intersection, union

That is:

- if L is regular then $\bar{L}$ is regular
- if L_1, L_2 are regular then $L_1 \cap L_2$ is regular
- if L_1, L_2 are regular then $L_1 \cup L_2$ is regular.

proof:

- **Complement:** If L is regular there's a DFA $M = (Q, \Sigma, \delta, q_0, F)$ s.t. $L(M) = L$

We want to construct a DFA recognizing $\bar{L}$

Idea: flip accept & reject states

Let $M' = (Q, \Sigma, \delta, q_0, Q \setminus F)$

(Q, Σ, δ, q_0: identical to M; $Q \setminus F$: $\{q \in Q \mid q \notin F\}$)

i.e. q is accept state in M' $\iff$ q is reject state in M

$L(M') = \overline{L}$ because the computation of M' on w is same as computation of M on w, so

$w \in \overline{L} \iff w \notin L \iff M$ rejects w

$\iff$ Computation of M on w ends in $q \notin F$

$\iff$ Computation of M' on w ends in $q \notin F$

$\iff$ M' accepts w. ✓

- intersection: If L_1, L_2 are regular, there are DFAs $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ s.t. $L(M_1) = L_1$

$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ s.t. $L(M_2) = L_2$

we want to construct a DFA for $L_1 \cap L_2$.

Idea: run both M_1, M_2 in parallel:
keep track of which state you would be in in both M_1, M_2,
accept iff both are accepting

Let

$$M = (Q_1 \times Q_2, \Sigma, \delta, (q_1, q_2), F_1 \times F_2)$$

$\{(r_1, r_2) \mid r_1 \in Q_1, r_2 \in Q_2\}$ $\{(r_1, r_2) \mid r_1 \in F_1$ and $r_2 \in F_2\}$

$$\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$$

state of M | symbol | state of M

[colors were added after class. In section 2 we didn't write the analysis.]

$L(M) = L_1 \cap L_2$ because ...

Consider any string $w = w_1 \dots w_n$

Let $(r_0, \dots, r_n)$ be the computation of M_1 on $w \rightarrow$

$r_0 = q_1$

$r_i = \delta_1(r_{i-1}, w_i)$

Let $(r_0', \dots, r_n')$ be the computation of M_2 on $w \rightarrow$

$r_0' = q_2$

$r_i' = \delta_2(r_{i-1}', w_i)$

Then from our construction, the computation of M on w is

$$((r_0, r_0'), \ldots, (r_n, r_n'))$$

(r_0, r_0'): start state of M

one step: $\delta((r_{i-1}, r_{i-1}'), w_i)$

(r_n, r_n'): last state in computation of M on w.

$$= (\delta_1(r_{i-1}, w_i), \delta_2(r_{i-1}', w_i))$$

$$= (r_i, r_i')$$

Thus, M accepts $w \iff (r_n, r_n') \in F_1 \times F_2$

$\iff r_n \in F_1$ and $r_n' \in F_2$

$\iff M_1$ accepts w and M_2 accepts w

$\iff w \in L_1 \cap L_2$ ✓

- Union:

[Proof 1: Use a similar construction to the intersection construction above, but modify it to recognize union.
How? try to do it yourself, but solution below.]

Alternative proof: Use the theorems we just proved!

Note that $L_1 \cup L_2 = \overline{\overline{L_1} \cap \overline{L_2}}$ ("De Morgan law")

Let L_1, L_2 be regular.

$\Rightarrow \overline{L_1}, \overline{L_2}$ are both regular (closure under complement)

$\Rightarrow \overline{L_1} \cap \overline{L_2}$ is regular (closure under intersection)

$\Rightarrow \overline{\overline{L_1} \cap \overline{L_2}}$ is regular (closure under complement)

"$L_1 \cup L_2$"

$\boxtimes$