## Announcements

- checkup #2 - 1st attempt tonight, last attempt +1 week
- HW1 - due Mon night
- Handout 1 - released

- checkup #3 - out soon, 1st attempt due Wed night, ditto

- How we grade/give feedback:
  - experimenting with pilot: grady.ai → Agreement w Columbia

    supposed to be:
    - consistent
    - lots of detailed feedback
    - fast(ish)

    great for ~300 students!

    hopefully! But we're checking:

  - TAs independently grade a subset of HW, compare with Grady, update instructions & rerun
  - Regrade requests always by human
- if things don't work, we'll revert to old-style.

Contact us if you have any concerns!
↳ Tal + William

- Our hope: grade + release feedback in time for Quiz 1 (HW1 solutions for sure)

Recap:

# NFA: nondeterministic finite Automaton

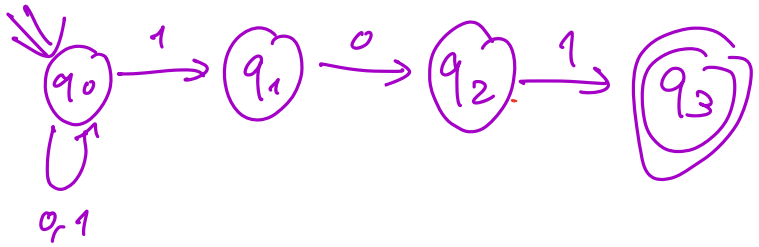
Informally: like a DFA, except we can have 0 or more transitions for each state and symbol. Transitions can also be labeled by $\varepsilon$

An NFA can have zero or more computations on a given string w

An NFA accepts w if there exists a computation that processes entire string and ends in an accepting state.

Ex:

N:




Computations of N on 101:

$q_0 \xrightarrow{1} q_0 \xrightarrow{0} q_0 \xrightarrow{1} q_0$ reject

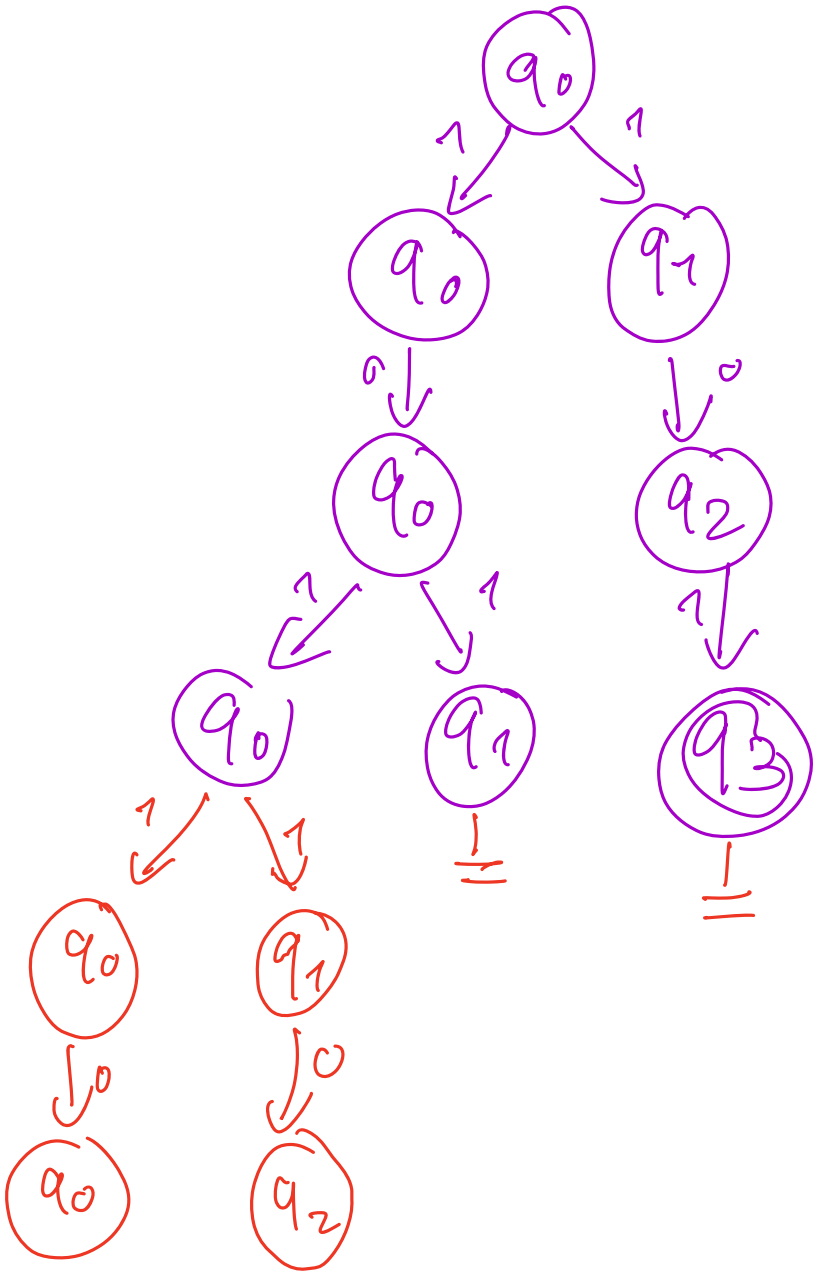
$q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{1} q_3$ accept

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

$\Rightarrow$ N accepts 101

what about 10110 ?

# Computation tree for NFA on string "10110"




$\{q_0, q_1\}$: possible states after reading 1

$\{q_0, q_2\}$: possible states after reading 10

$\{q_0, q_1, q_3\}$: possible states after reading 101

$\Rightarrow 101 \in L(N)$

$\{q_0, q_1\}$: possible states after reading 1011

$\{q_0, q_2\}$: possible states after reading 10110

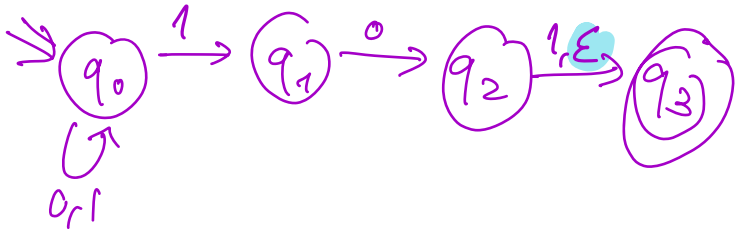
$\Rightarrow 10110 \notin L(N)$ (rejected)

$L(N) = \{w \mid w \text{ ends in } 101\}$

When there's an $\varepsilon$-transition, computation can either take it or not

Ex:

$N'$:




Computations of $N'$ on string 10:

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

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

$q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{\varepsilon} q_3$ accepting!

$\Rightarrow N'$ accepts 10

$L(N') = \{w \mid w$ ends in 10 or $w$ ends in $101\}$

Another example:

$N''$:



$L(N'') = \{\varepsilon\}$

(for any other string, the computation dies, as there is no transition coming out of start state $\Rightarrow$ only string that can be accepted is $\varepsilon$.)

2 views on NFA computation:

- try all possible computations in parallel, accept if at least one succeeds

or

- whenever there is more than one option, guess and hope for the best.

If there is a way to get lucky (sequence of choices leading to accept), the string is accepted.

Note: this does not seem like a realistic model of computation... why do we define it?
Turns out to be super useful, we'll see later!

Formal Def of NFA (difference from DFA highlighted)

For alphabet $\Sigma$ define $\Sigma_\varepsilon := \Sigma \cup \{\varepsilon\}$

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

where:

- $Q$ is a finite set of states
- $\Sigma$ is an alphabet
- $\delta : Q \times \Sigma_\varepsilon \to \mathbb{P}(Q)$
- $q_0 \in Q$ called "start state"
- $F \subseteq Q$ the set of accepting states

Recall that $\mathbb{P}(Q)$ is the power set of $Q$, namely the set of all subsets of $Q$: $\mathbb{P}(Q) = \{S \mid S \subseteq Q\}$

The transition function takes a (state, symbol)
- state: in $Q$
- symbol: either in $\Sigma$ or $\varepsilon$

and returns a subset of states (all states the transition can go to)
- subset: 0 or more

**Def:** for a NFA $N=(Q,\Sigma,\delta,q_0,F)$ and $w\in\Sigma^*$

a computation of $N$ on $w$ is a sequence of states $(r_0,\dots,r_n)$ such that

- $w$ can be written as $w=w_1\dots w_n$, where $w_i\in\Sigma_\varepsilon$ (allow to insert $\varepsilon$)
  - $r_0=q_0$
  - $\forall i\in\{1,\dots,n\}\quad \delta(r_{i-1},w_i)=r_i$

A computation $(r_0,\dots,r_n)$ is accepting if $r_n\in F$.

$N$ accepts $w$ iff there exists an accepting computation of $N$ on $w$.

(if all computations of $N$ on $w$ are not accepting, we say that $N$ rejects $w$)

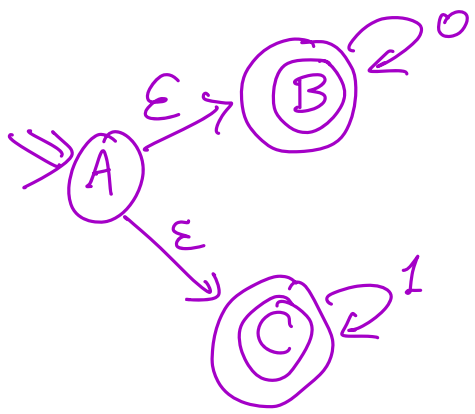
**Def:** For an NFA $N$,

the language recognized by $N$ is

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

## Examples:

N:




$L(N) = ?$

N in formal notation:

$$N = (\{A, B, C\}, \{0, 1\}, \delta, A, \{B, C\})$$

$Q$ — $\Sigma$ — start state — accepting state

where

| $\delta$: | 0 | 1 | $\varepsilon$ |
|---|---|---|---|
| A | $\{\}$ | $\{\}$ | $\{B, C\}$ |
| B | $\{B\}$ | $\{\}$ | $\{\}$ |
| C | $\{\}$ | $\{C\}$ | $\{\}$ |

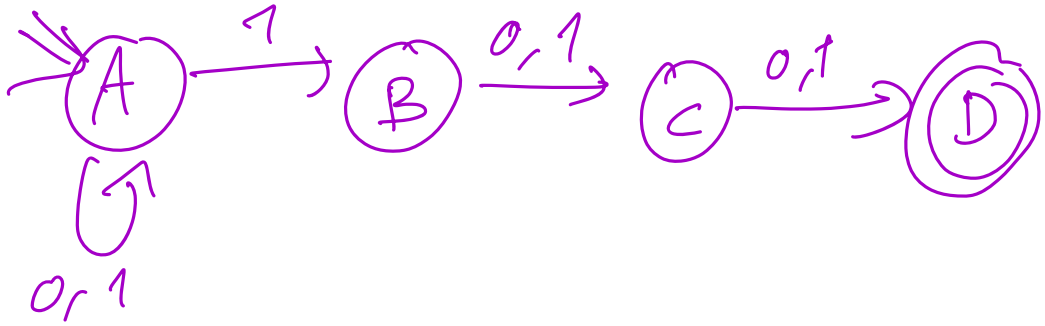
($\{\}$ is also denoted $\emptyset$)

Note that $\delta : Q \times \Sigma_\varepsilon \to \mathcal{P}(Q)$

$L(N) = \{ w \in \{0,1\}^* \mid w$ consists of all 0's or $w$ consists of all 1's $\}$

$= \{0^n \mid n \geq 0\} \cup \{1^n \mid n \geq 0\}$

- N:




$L(N) = \{w \in \{0,1\}^* \mid$ 3-rd to last symbol in $w$ is $1\}$

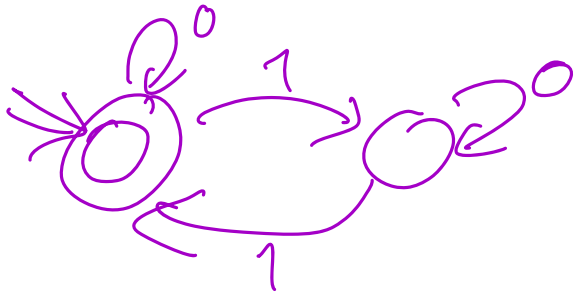
- Revisiting question from end of lecture 2:

$L = \{w \in \{0,1\}^* \mid w = x \cdot y$ x has even # of 1's, y has even # of 0's$\}$

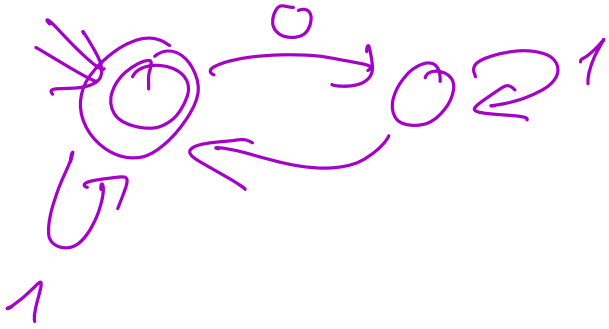
(possible but) hard to construct DFA for it from scratch...

but NFA seems much easier!

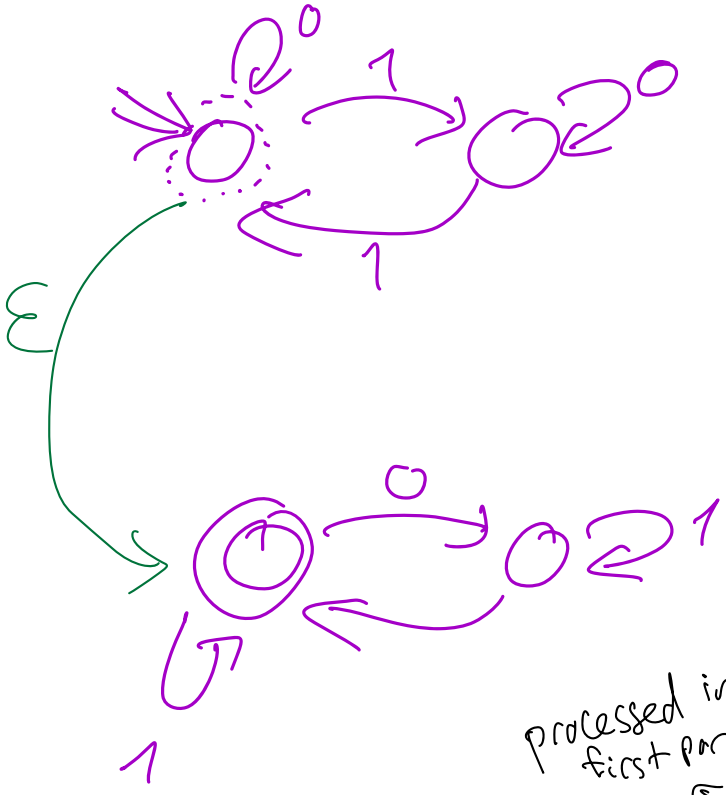
First show NFA for $\{x \mid x$ has even # of 1's$\}$



Now NFA for $\{y \mid y \text{ has even \# of 0's}\}$




NFA for $L_6$




processed in first part — processed in second part

This NFA accepts $w \iff w$ can be parsed as $w = x \circ y$
where $x$ has even # of 1s
$y$ has even # of 0's

So we showed an NFA for $L$.

This means $L$ is regular by the following theorem

**Theorem:** $L$ is regular (i.e. $\exists$ DFA recognizing $L$) $\iff$ $\exists$ NFA recognizing $L$

**Proof:** ($\Rightarrow$) immediate, since DFA is a special case of an NFA.

($\Leftarrow$) Let $N = (Q_N, \Sigma, \delta_N, q_N^0, F_N)$ be a NFA.

We will construct a DFA $D = (Q_D, \Sigma, \delta_D, q_D^0, F_D)$ recognizing the same language.

Idea: as you process the string, keep track of all possible states in $N$ I could possibly be in: $Q_D$ will have one state for each set of states in $Q_N$

$Q_D = \mathcal{P}(Q_N)$

For a set $S \subseteq Q_N$, define

$E(S) = S \cup \{$all states reachable from $S$ by following any # of $\varepsilon$ transitions$\}$

$q_D^0 = E(\{q_N^0\})$

$F_D = \{S \subseteq Q_N \mid S \cap F_N \neq \emptyset\}$

i.e. $S$ is an accept state of $D$ if it contains at least one accept state of $N$.

For $S \subseteq N$, $a \in \Sigma$, define

$$\delta_D(S,a) = \{ q' \in Q_N \mid q' \in E(\delta_N(q,a)) \text{ for some } q \in S \}$$

eg, if $S = \{q_1, \dots q_k\}$ and we read 'a', we first take $\delta_N(q_1,a) \cup \delta_N(q_2,a) \cup \dots \cup \delta_N(q_k,a)$, then add any other state we get by following any finite number of $\varepsilon$-transitions.

It can be shown that on input string $x$ the computation of $D$ ends in a state corresponding to

$$\{ q \in Q_N \mid \exists \text{ a computation of } N \text{ on } x \text{ ending in } q \}$$

this is an accepting state $\Leftrightarrow$ $N$ accepts $x$

$\Rightarrow$ $L(D) = L(N)$ ☒

This construction is called "subset construction"