Lecture 7:

What did we cover last time?

(I'll ask you after annoucements!!!)

Exciting announcements:

* checkup 3 due Wed
* checkup 2 due Thur
* Quiz 1 next Tuesday
  - ↳ come to OH if you're worried

The wrapper had the skills you'll be tested on!

* HW1 was due last night
* challenge problem!!!

NFAs: we saw the formal def

* Formal definition of NFA:

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

↳ now $\delta : \Sigma_\varepsilon \times Q \to P(Q)$

Σ_ε: $\Sigma \cup \{\varepsilon\}$

$P(Q)$: power set of Q

$L(N) = \{w \mid N \text{ accepts } w\}$

? If there $\exists$ a computation that ends in an accept state

L is regular $\Leftrightarrow$ there's a DFA with $L(D) = L$

$\Leftrightarrow$ there's an NFA with $L(N) = L$

Main thing: how to turn an NFA into a DFA!

NFA: Q

DFA: $P(Q)$

* we had one state in the DFA for each subset of state in the NFA

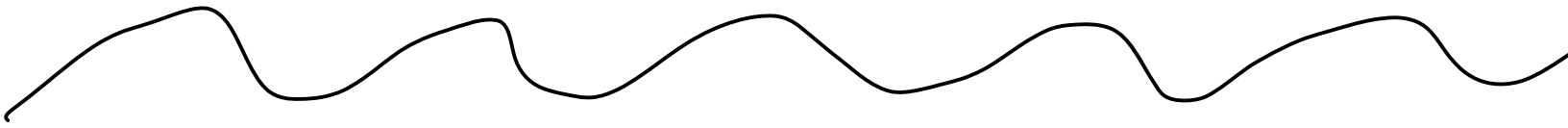
=> If NFA has $|Q| = k$ states, the DFA has $2^k = |P(Q)|$ states!

Note: is this blowup necessary?

Sometimes yes: $L = \{w \mid n\text{-th to last symbol is } 1\}$

=> Try it : an NFA with $n+1$ states

Can use MN to show you need 2^n states !



So if NFAs are as powerful as DFAs why even define them:

* Non-determinism is cool (we'll revisit it)

* NFAs are like a higher level language => it's easier to use than DFAs!

Today : closure properties of regular languages.

Def: If L_1, L_2 are languages over Σ we define:

* concatenation : $L_1 \circ L_2 = \{ w \mid w = xy \text{ where } x \in L_1, y \in L_2 \}$

$L_1 = \{\varepsilon, 001, 1\}$ $L_2 = \{1, 11\}$

$L_1 \cdot L_2 = \{\varepsilon\cdot 1, \underbrace{\varepsilon \cdot 11}_{11}, 001\cdot 1, 001\cdot 11, \overbrace{1\cdot 1}^{11}, 1\cdot 11\}$

⚠ $L_2 \cdot L_1 \neq L_1 \cdot L_2$ in general

ex : $11\,001 \in L_2 \cdot L_1$

Power: $L^0 = \{\varepsilon\}$, $L^{i+1} = L^i \cdot L$ for all $i \geq 0$

So L^k is all strings w such that w can be written as $x_1 x_2 \dots x_k$ where each $x_i \in L$

Ex: $L = \{11, 1, 0\}$

what's $L^0 = \{\varepsilon\}$, $L^1 = L$

$L^2 = \{11\cdot 11, 11\cdot 1, 11\cdot 0, 1\cdot 11, 1\cdot 1, 1\cdot 0$
$0\cdot 11, 0\cdot 1, 0\cdot 0\}$

$L_1 = L^0 \cdot L = \{\varepsilon\} \cdot L = \{\varepsilon\}\cdot\{11, 1, 0\}$
$= \{11, 1, 0\} = L$

$L^* = \bigcup_{k=0}^{\infty} L^k = L^0 \cup L^1 \cup L^2 \cup \dots$

$= \{ w \in \Sigma^* \mid w$ can be written as

$x_1 \cdot x_2 \cdots \cdot x_k$ where $k \geq 0$

and each $x_i \in L \}$

$\{1, 00\}^* = \underbrace{\{\varepsilon\}}_{L^0} \cup \underbrace{\{1, 00\}}_{L^1} \cup \underbrace{\{11, 100, 001, 0000\}}_{L^2} \cup \ldots$

L^* is infinite unless $L = \{\varepsilon\}$ or $L = \emptyset$

$L^*, L'^* = \{\varepsilon\}$

Theorem: The class of regular languages is closed under concatenation, power (for any k) and $*$

I.e: If L_1, L_2 are regular languages over Σ

(1) - $L_1 \cdot L_2$ is regular

- L_1^k is regular for all $k \geq 0$

- L_1^* is regular

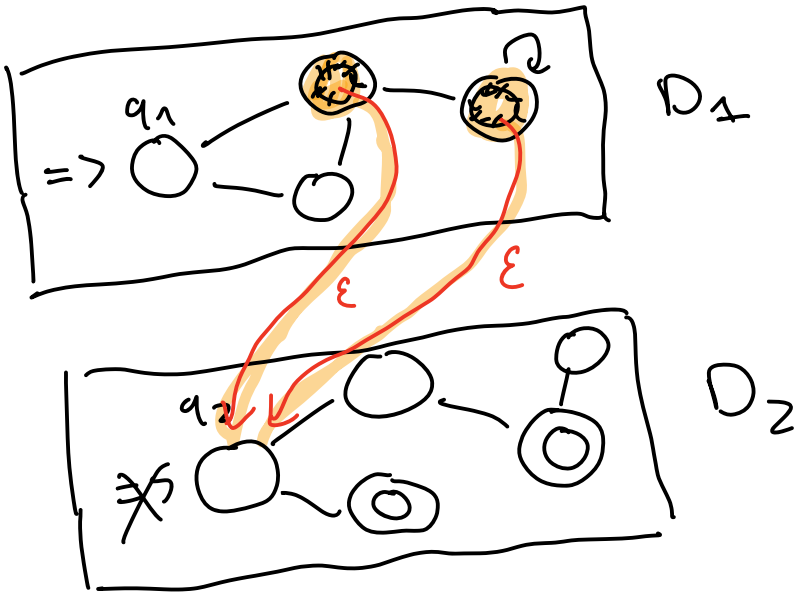
proof of (1). Let L_1, L_2 be regular languages

so let $D_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ and

$D_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ be DFAs

such that $L(D_1) = L_1$, $L(D_2) = L_2$

We'll build an NFA N recognizing $L_1 \circ L_2$



$D_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$

$D_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$

Idea: w can be written as $w = x \cdot y$ where x is accepted by D_1 and y is accepted by D_2

idea: "guess when we've read x"



Also need need to make states in F_1 not accept any more: else we would accept all strings in L_1 (they might not be in $L_1 \circ L_2$

$N = (Q_1 \cup Q_2, \Sigma, \delta, q_1, F_2)$ — q_1: start of D_1; F_2: accept of D_2

δ: For all $a \in \Sigma$. (transitions of D_2)

For $q \in Q_1$: $\delta(q,a) = \{\delta_1(q,a)\}$

For $q \in Q_2$: $\delta(q,a) = \{\delta_2(q,a)\}$ (transition of D_2)

For all $q \in F_1$: $\delta(q,\varepsilon) = \{q_2\}$

q: this is a state but δ outputs a set of states

ε from accept of D_1 to start of D_2

N accepts $w \iff w = xy$ where D_1 accepts x, D_2 accepts y

$\iff w = xy, x \in L_1, y \in L_2$

$\iff w \in L_1 \circ L_2$

- Power: Let $k \geq 0$. We want to show L^k is regular whenever L is regular.

$$L^k = \underbrace{L \circ L \circ \cdots L}_{k \text{ times}}$$

Proof by induction: Let L be a regular language....

$L^0 = \{\varepsilon\} \Rightarrow$  is regular.

Assume L^i is regular, we want to prove L^{i+1} is regular. By assuption L^i and L are regular so $\underbrace{L^i \cdot L}_{L^{i+1}}$ is regular □

$*$: we want to prove that if L is regular then L^* is regular. $L^* = \bigcup_{k=0}^{\infty} L^k$.

If L is regular, then $\forall\, k \geq 0$, L^k is regular.

We also know that if L_1, L_2 are regular then $L_1 \cup L_2$ are regular. So L^* is regular.

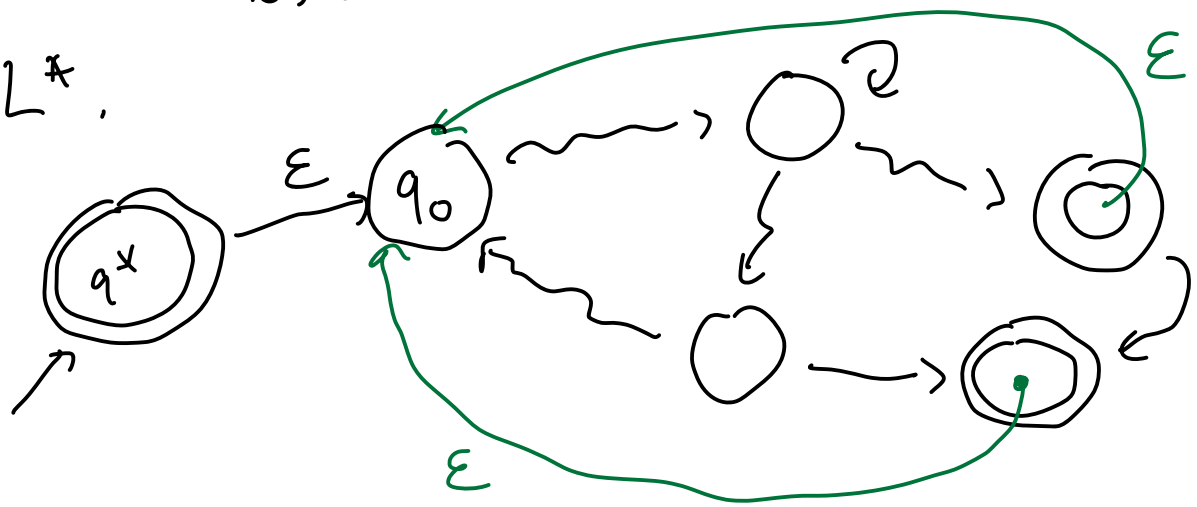
not true

We can show by induction that for any $k \geq 0$ the union of k regular languages is regular (finite union)

But regular languages are not closed under ∞ unions

Try it : Find some language L that's not regular but $L = L_1 \cup L_2 \cup \dots$ where each L_i is regular.

Suppose we have a DFA $D = (Q, \Sigma, \delta, q_0, F)$ with $L(D) = L$. We want to build an NFA for L^*.



$w \in L^*$, if we can write $w = x_1 \cdots x_k$ for some $k \geq 0$ where each $x_i \in L$. If we put x_i through the DFA we'll end at an accept state.

This recognizes $\bigcup_{k=1}^{\infty} L^k$ (after we add the ε transitions)

Making q_0 accept doesn't quite work

We add a new start state q^* with ε transition to q_0

new state ↓ new start ↓ same accept ↙

$$N = (Q \cup \{q^*\}, \Sigma, \delta', q^*, F \cup \{q^*\})$$

δ': for all symbol $a \in \Sigma$, for all $q \in Q$

$$\delta'(q, a) = \{\delta(q, a)\}$$

back to original start state

For all $q \in F$: $\delta'(q, \varepsilon) = \{q_0\}$

$$\delta'(q^*, \varepsilon) = \{q_0\}$$

Σ, $\delta: \Sigma_\varepsilon \times Q \to \mathbb{P}(Q)$

$\Sigma_\varepsilon = \Sigma \cup \{\varepsilon\}$

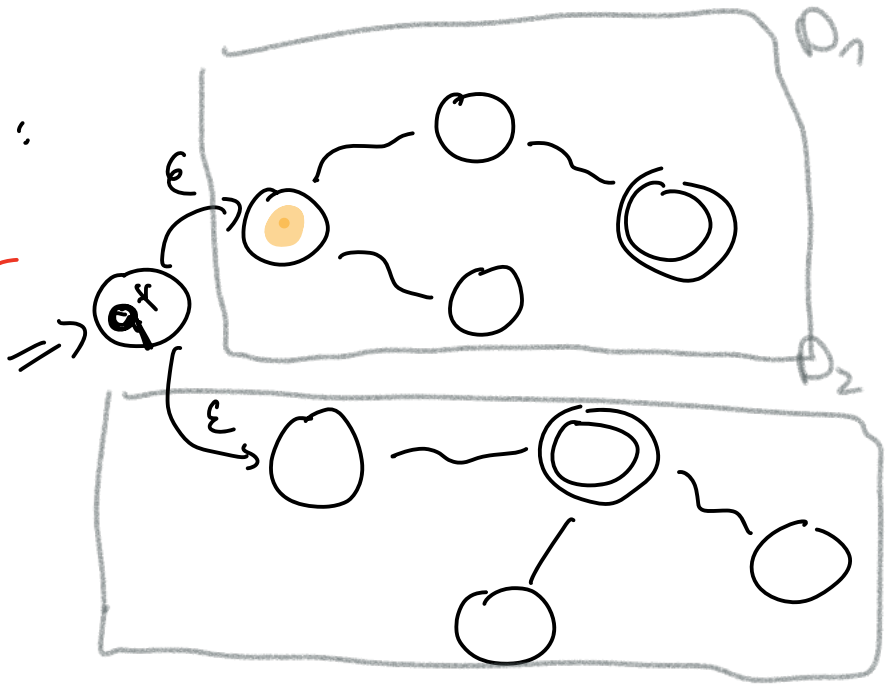
Another proof for union: If L_1, L_2 are regular languages then $L_1 \cup L_2$ is regular

Let D_1 be a DFA with $L(D_1) = L_1$

D_2 be a DFA with $L(D_2) = L_2$

NFA for $L_1 \cup L_2$:

* ε from D_1's start to D_2's start doesn't work :



This works : on any input w we can "choose" to run D_1 or D_2 or w

(or you can think we run both in parallel)

We saw closure under $\cup, \overline{\ }, \cap, \circ$, power, $*$

Def: the regular operations $\cup, \circ, *$ are called the regular operators

$\Rightarrow$ L is regular if and only if L can be written as a finite combination of characters in Σ, ε, $\emptyset$ and the regular operators.

$\Rightarrow$ Regular expressions