Power of nondeterminism: Closure properties of NFAs

Friday, April 11, 2025 11:33 AM

Nondeterministic Finite Automata (NFA).

- Q.
- S multiple starting states
- A multiple accepting states
- Σ_ε w/ ε-transition
- $\delta : 2^Q \times \Sigma_\varepsilon \to 2^Q$.

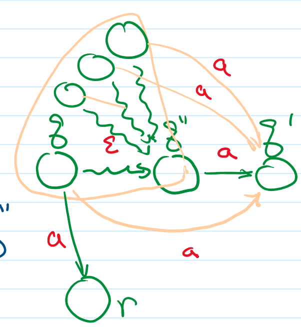
definitions are not sacred.

$$\delta^*(P, w) := \begin{cases} \varepsilon\text{-Reach}(P) & \text{if } w = \varepsilon \\ \delta^*(\boxed{\delta(\varepsilon\text{-Reach}(P), a)}, x) & \text{if } w = ax \end{cases}$$

Looks deterministic to me ... if we record all fingers.

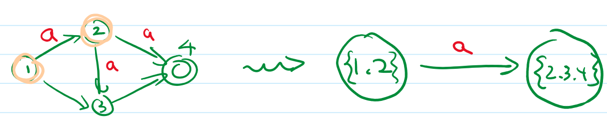
Thm. For any NFA N, there's a DFA M recognizing the same language.

pf. 1. Construct NFA N' w/o ε-transitions.



- if $q'' \xrightarrow{a} q'$ exists, then add $q \xrightarrow{a} q'$ $\forall q$ ε-reachable to q''
- Add q into A if $q \overset{\varepsilon}{\rightsquigarrow} \circledcirc$ $\qquad \bigcirc \overset{\varepsilon}{\rightsquigarrow} \circledcirc$

2. Construct DFA M emulating NFA N':

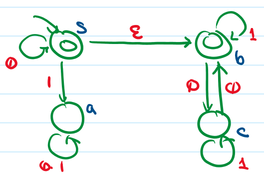


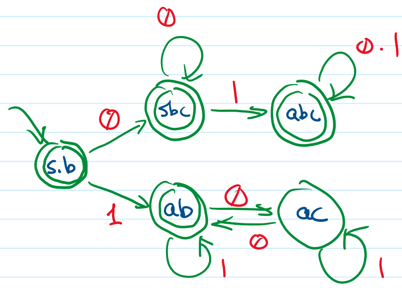
- $Q_M := 2^{Q_{N'}}$ ← exp. blowup.
- $s_M := S_{N'}$
- $A_M := \{ P \in Q_M = 2^{Q_{N'}} : \cancel{P \cap A_{N'} \neq \emptyset} \}$ $\quad P \subseteq A_{N'}$
- $\delta_M(P, a) := \delta_{N'}(P, a)$

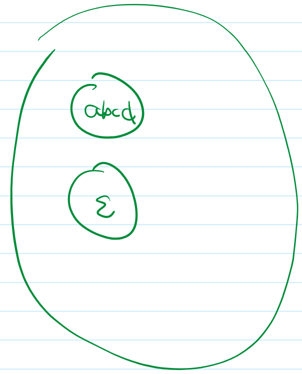
• $A_M := \{ P \in Q_M = 2^{Q_{N'}} : \cancel{P \cap A_{N'} \neq \emptyset} \}$

• $\delta_M(P, a) := \delta_{N'}(P, a)$

<u>example.</u> [Incremental Construction]







P	ε-Reach(P)	$\in A_M$?	$\delta_M(P, 0)$	$\delta_M(P, 1)$
s	sb	✓	sc	ab
sbc = sc	sbc	✓	sbc	abc
ab	ab	✓	ac	ab
abc	abc	✓	abc	abc
ac	ac	✗	ab	ac

<u>Cor.</u> A language is automatic if some NFA accepts it.

<u>Cor.</u> Regular languages are automatic.

$$\text{R.E.} \quad \text{DFA} \longrightarrow \text{NFA}$$

subset

parse tree.

<u>META-Question</u>: What questions should we ask at this point?

1. Which automatic language is not regular?
2. Which languages are not automatic?
3. L is non-automatic. How to represent?
4. How to store/emulate DFAs in CPUs?
5. Is the exp. blowup inherit?
6. Can I count in NFAs?

   take majority
   match parentheses?

• Closure under operations?

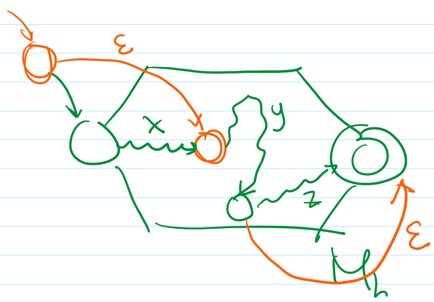
match parentheses?

- Closure under operations?

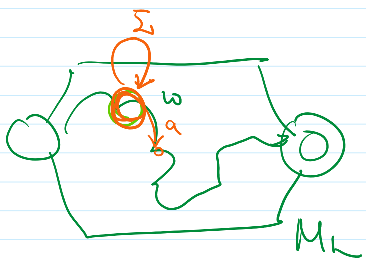
$L_1 \wedge L_2 := \{ w \in \Sigma^* : w \in L_1 \text{ and } w \in L_2 \}$

  - all NFAs: fingers have to be all in accepting states.
  - can be emulated by DFAs, using (modified) subset construction.

$\text{Substring}(L) := \{ y \in \Sigma^* : xyz \in L \text{ for some } x, z \in \Sigma^* \}$

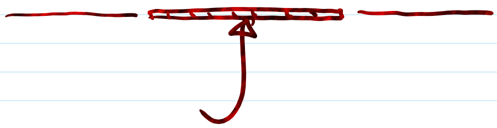


$\text{superseq}(L) := \{ x \in \Sigma^* : x \text{ is superseq. of some } w \in L \}$



$\text{left}(L) := \{ x \in \Sigma^* : xy \in L \text{ for some } y \in \Sigma^*, \text{ and } |x| = |y| \}$

Question. Are O(1)-memory programs better than no-memory ones?



Kleene Thm. [1951]

Every automatic language is regular. i.e.
every language accepted by some DFA has a reg. expression.

[Han-Wood'05]
Pf. generalize NFAs even further.

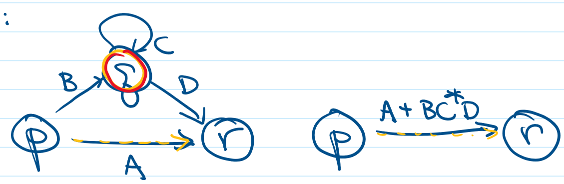
$q \xrightarrow{\cancel{x}\ R} q'$: take transition after reading $x \in R$,

GNFA accepts w if $\exists\, s \xrightarrow{R_1} q_1 \xrightarrow{R_2} \cdots \xrightarrow{R_\ell} q_\ell$
$w = x_1 \cdot x_2 \cdot \cdots \cdot x_\ell$, $x_i \in R_i$

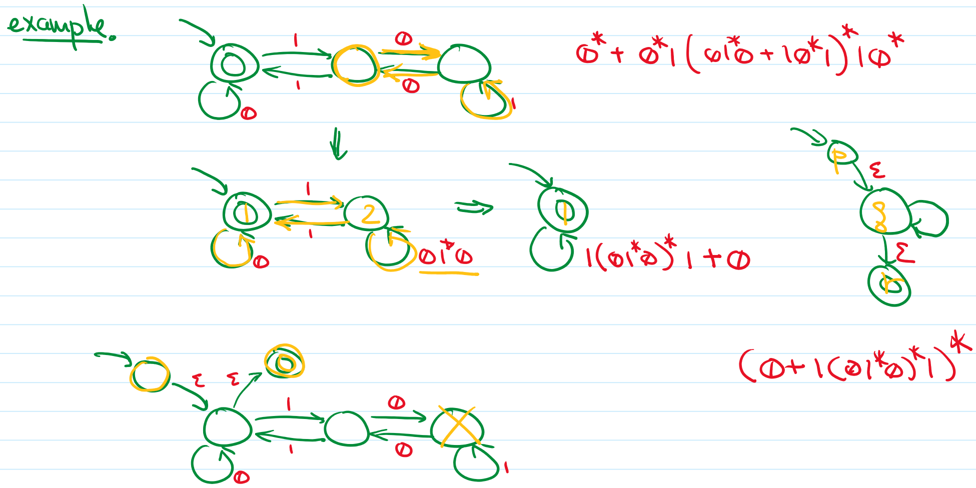
(intuition: any decomposition of w matching any walk in GNFA)

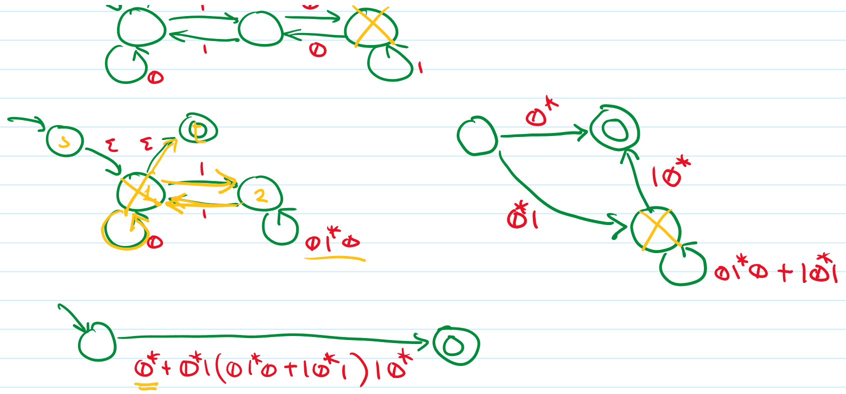
∘ Now, turn GNFA into RE.

To remove q:



example.





Cor. Regular languages can be modeled as:

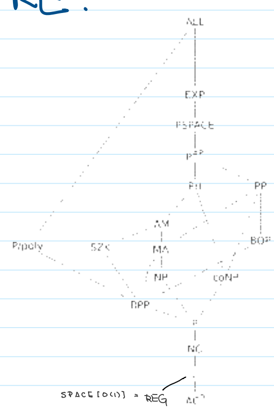
$$\text{RE.} \xleftarrow{\text{APSP}} \text{GNFA}$$

$$\text{RE.} \quad \text{DFA} \longrightarrow \text{NFA} \longrightarrow \text{GNFA}$$

$$\text{NFA} \xrightarrow{\text{subset}} \text{DFA}, \qquad \text{RE.} \xrightarrow{\text{parse Tree.}} \text{NFA}$$

Moral. Different models work better in diff. scenarios.

- RE: recursive def., good for induction.
- DFA: deterministic. good for what can't be done.
- NFA: good for algorithm design.
- GNFA: exist for the sake of reduction to RE. (middle-step object).



Concluding Question. DFAs are surprisingly powerful.
What can't DFAs do?