Kleene's Theorem: where multiple models meet

Wednesday, April 16, 2025 12:53 PM

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

$$\begin{tikzcd} \text{R.E.} \arrow[rr, bend right, "\text{parse tree.}"'] & \text{DFA} \arrow[r] & \text{NFA} \arrow[l, bend right, "\text{subset}"'] \end{tikzcd}$$

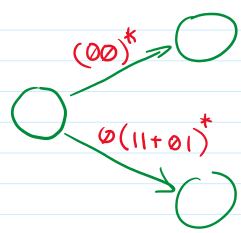
Kleene Thm. [1951]

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

[Han-Wood'05]

Pf. generalize NFAs even further.

$q \xrightarrow{R} q'$: take transition if reading w that matches R (R.E.)

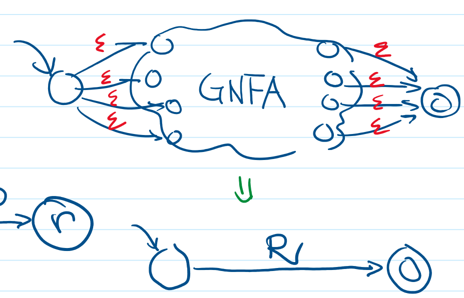


GNFA accepts w if $\exists$ decomp. $w = x_1 \circ x_2 \circ \cdots \circ x_k$

$\exists$ $s \xrightarrow{R_1} q_1 \xrightarrow{R_2} q_2 \to \cdots \to q_k$ $\quad x_i \in R_i$

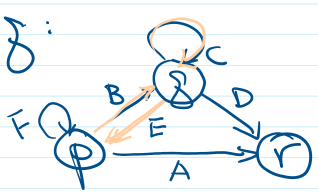
(intuition: any decomposition of w matching any walk in GNFA)

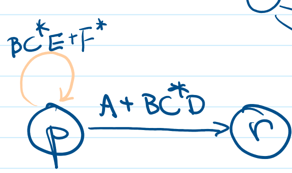
- Now, turn GNFA into RE.



to remove q:

$\forall p, r$:

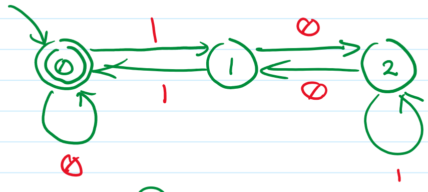


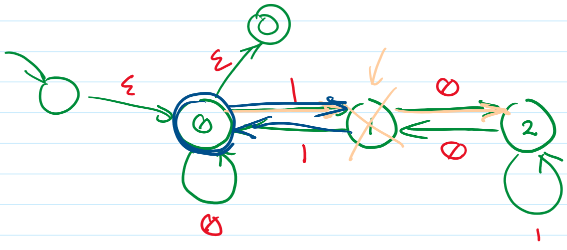


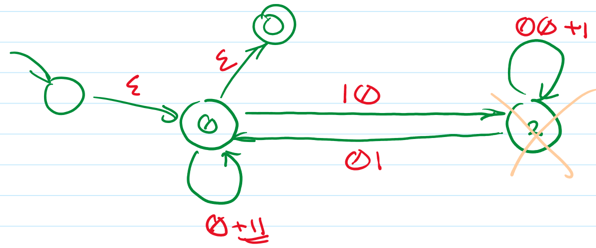
example. { binary repr. of n divisible by 3 }

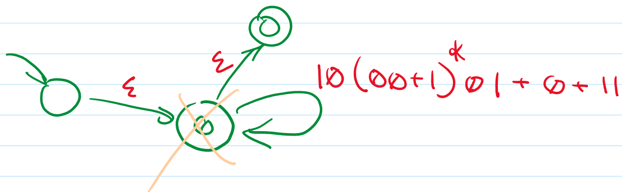
$1001_2 = 9$

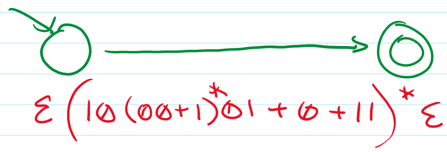
$11011_2 = 27$









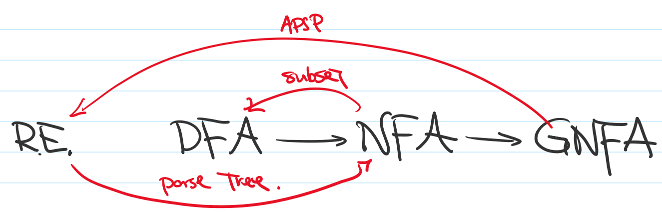


1001

100001

11011

Cor. Regular languages can be modeled as:



Moral. Different models work better in diff. scenarios.

- RE : recursive def., good for induction. the AND/OR/NOT of regular language.
- 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?

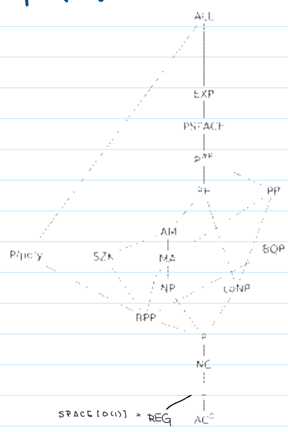
Question. How do we show that no program (w/ restriction)
can solve a specific problem?

Answer. We need to analyse
the structure of programs

the simpler the better !

- This is incredibly hard in general.

Q. What can't a DFA do?
what problem is too hard to be solved by DFA/NFAs?



addition?

counting ?
majority ?
nested parentheses ?))((

Q. How do we prove this?

Most important restriction of DFA:

- Finite #states. (indep. to input length)
- Deterministic.