How to prove nonregularity? Myhill-Nerode theorem

1/27/21 9:45 AM

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?

- Rob. problem.
- ? Count length of input.
- add, subtract, multiply. "z = x + y"

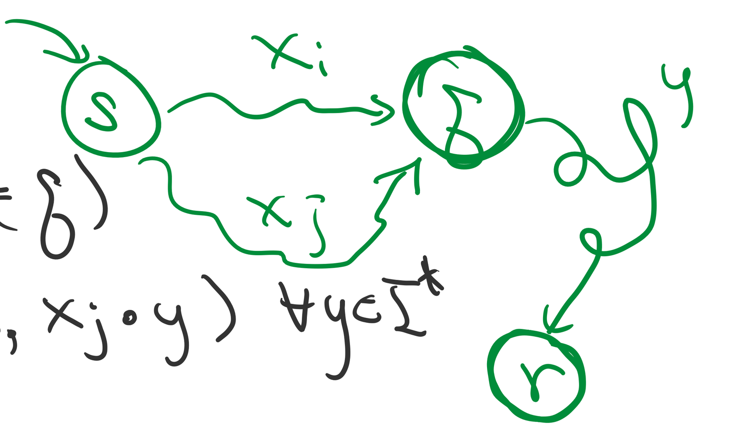
$L := \{ \boxed{x \# y} : \text{Is } x \text{ longer than } y \}$

Q. How do we prove this?

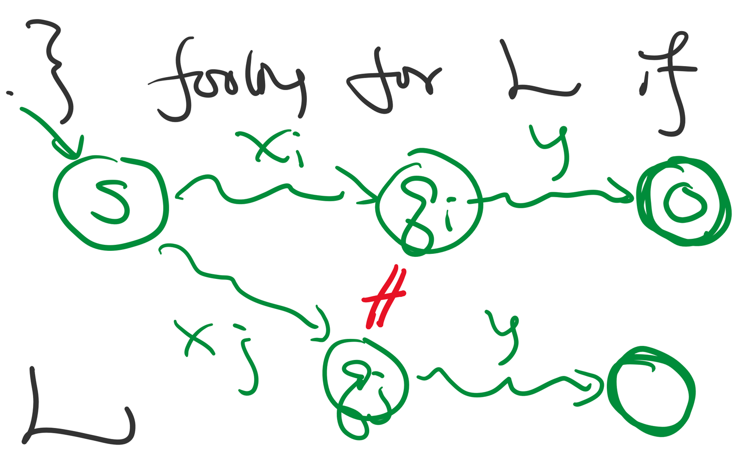
Most important restriction of DFA:

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

Observation. For any DFA M,
If $\delta^*_M(s, x_i) = \delta^*_M(s, x_j) (= q)$
then $\delta^*_M(s, x_i \cdot y) = \delta^*_M(s, x_j \cdot y) \; \forall y \in \Sigma^*$



Fooling prefixes/set: $F = \{x_1, x_2, \dots\}$ fooling for L if
for any given pair $x_i \neq x_j \in F$
you can choose suffix y. s.t.
exactly one $x_i \cdot y$, $x_j \cdot y$ is in L



example. $L := \{\text{binary integers divisible by } 3\}$

$F = \{0, 1, 10\}$ is fooling for L:

1 choose $y = 1$ $11 \in L$
10 $101 \notin L$

example. $L := \{x \# y : x \text{ is longer than } y\}$.

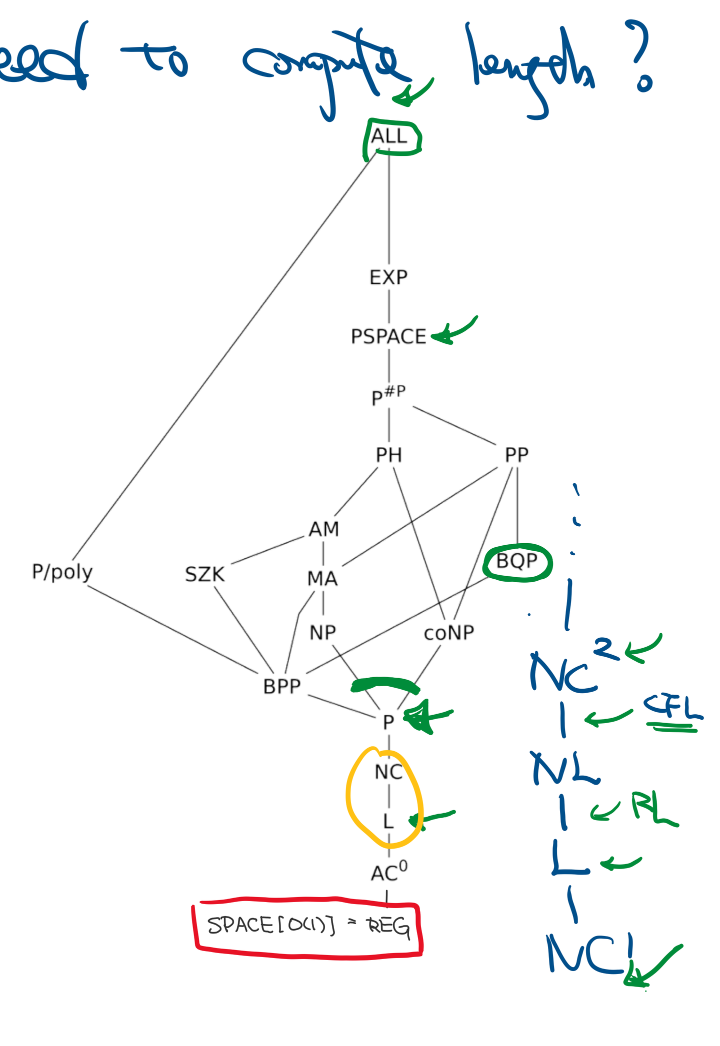
$F = \{0^k \# : \text{for } k > 0\}$, $|F| = \infty$.
$\forall x_i \neq x_j \in F$ $0^i \#$ $0^j \#$ $(i < j)$
$y := 0^i$ $x_i \cdot y = 0^i \# 0^i \notin L$ $x_j \cdot y = 0^j \# 0^i \in L$

Lemma. If $\exists$ fooling set F : $|F| = \infty$, for L.
then L is not regular.

Myhill-Nerode Thm. L regular,
Max # fooling prefixes for L = Min. # states in DFA for L.

Q. Alright. so what resource do we need to compute length?

- Counter : L
- stack : CFL
- circuit :

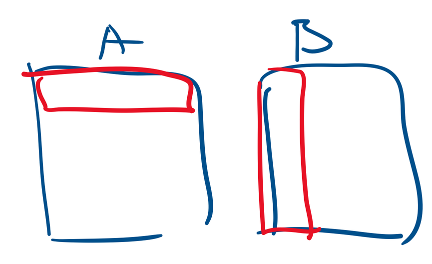


Q. How to solve a maze?
WFS

Q. Edit dist.? Colinearity?
Dynamic Prog.

Q. Parser

Q. Matrix multiplication/inversion.



Q. Linear Programs & Optimization?

Q. Untangling a knot? Factoring?

Q. Optimal strategy for games? Does your NFA accepts Σ^*?

Q. Does my code run forever?

Q. What resource do we need to perform "universal computation"?

Theme for the rest of the class:

- Solve problems in the most efficient way (algorithms)
- Show problems can't be solved under given resource (complexity)
- Some problems are universal: solving them solves a whole class of problems (reductions) / suggests other problems cannot be solved.
- There's a single universal model capturing all computations (computability)
- Some problems cannot be solved.