BILL, RECORD LECTURE!!!!

BILL RECORD LECTURE!!!

Regex: Closure Properties

Terminology: Regular Languages

Def We will say **regex** when we mean a lang generated by a regex.
For example we might say:

regex is closed under Isaac

if Isaac was some operation on sets.

Terminology: Regular Languages

Def We will say **regex** when we mean a lang generated by a regex.
For example we might say:

regex is closed under Isaac

if Isaac was some operation on sets.

We prove (or disprove) closure properties of regex.

Terminology: Regular Languages

Def We will say **regex** when we mean a lang generated by a regex. For example we might say:

regex is closed under Isaac

if Isaac was some operation on sets.

We prove (or disprove) closure properties of regex.

We already know all of these closure properties since we did closure proofs with DFA's and NFA's; however, we are curious which ones can be proven **easily** with regex's.

Regex Closed Under Complementation

How do you complement a regular language (not a joke)?

Regex Closed Under Complementation

How do you complement a regular language (not a joke)?

While not a joke, there is no easy way to go from regex α to regex β such that $L(\beta) = \overline{L(\alpha)}$.

Regex Closed Under Complementation

How do you complement a regular language (not a joke)?

While not a joke, there is no easy way to go from regex α to regex β such that $L(\beta) = \overline{L(\alpha)}$.

Here is how you can do it:

Regex Closed Under Complementation

How do you complement a regular language (not a joke)?

While not a joke, there is no easy way to go from regex α to regex β such that $L(\beta) = \overline{L(\alpha)}$.

Here is how you can do it: Given regex α of length n.

Regex Closed Under Complementation

How do you complement a regular language (not a joke)?

While not a joke, there is no easy way to go from regex α to regex β such that $L(\beta) = \overline{L(\alpha)}$.

Here is how you can do it: Given regex α of length n.

Create NFA N such that $L(N) = L(\alpha)$. $\sim n$ states.

Regex Closed Under Complementation

How do you complement a regular language (not a joke)?

While not a joke, there is no easy way to go from regex α to regex β such that $L(\beta) = \overline{L(\alpha)}$.

Here is how you can do it: Given regex α of length n.

Create NFA N such that $L(N) = L(\alpha)$. $\sim n$ states.

Convert N to a DFA M such that $L(N) = L(M)$. $\sim 2^n$ states.

Regex Closed Under Complementation

How do you complement a regular language (not a joke)?

While not a joke, there is no easy way to go from regex α to regex β such that $L(\beta) = \overline{L(\alpha)}$.

Here is how you can do it: Given regex α of length n.

Create NFA N such that $L(N) = L(\alpha)$. $\sim n$ states.

Convert N to a DFA M such that $L(N) = L(M)$. $\sim 2^n$ states.

Swap the final and nonfinal states of M to get M'. $\sim 2^n$ states.

Regex Closed Under Complementation

How do you complement a regular language (not a joke)?

While not a joke, there is no easy way to go from regex α to regex β such that $L(\beta) = \overline{L(\alpha)}$.

Here is how you can do it: Given regex α of length n.

Create NFA N such that $L(N) = L(\alpha)$. $\sim n$ states.

Convert N to a DFA M such that $L(N) = L(M)$. $\sim 2^n$ states.

Swap the final and nonfinal states of M to get M'. $\sim 2^n$ states.

Convert M' to regex α. $\sim 2^{2^n}$ states.

Regex Closed Under Complementation

How do you complement a regular language (not a joke)?

While not a joke, there is no easy way to go from regex α to regex β such that $L(\beta) = \overline{L(\alpha)}$.

Here is how you can do it: Given regex α of length n.

Create NFA N such that $L(N) = L(\alpha)$. $\sim n$ states.

Convert N to a DFA M such that $L(N) = L(M)$. $\sim 2^n$ states.

Swap the final and nonfinal states of M to get M'. $\sim 2^n$ states.

Convert M' to regex α. $\sim 2^{2^n}$ states.

Are there α where you get $\sim 2^{2^n}$ blowup?

Regex Closed Under Complementation

How do you complement a regular language (not a joke)?

While not a joke, there is no easy way to go from regex α to regex β such that $L(\beta) = \overline{L(\alpha)}$.

Here is how you can do it: Given regex α of length n.

Create NFA N such that $L(N) = L(\alpha)$. $\sim n$ states.

Convert N to a DFA M such that $L(N) = L(M)$. $\sim 2^n$ states.

Swap the final and nonfinal states of M to get M'. $\sim 2^n$ states.

Convert M' to regex α. $\sim 2^{2^n}$ states.

Are there α where you get $\sim 2^{2^n}$ blowup?
I think so but the literature is unclear on this point.

Regular Lang Closed Under Union

Easy The regex for $L(\alpha) \cup L(\beta)$ is $\alpha \cup \beta$.

Regular Lang Closed Under Intersection

Hard Need to convert to NFA's and do it there and convert back.

Regular Lang Closed Under Intersection

Hard Need to convert to NFA's and do it there and convert back. Might be on a HW or Exam.

Regex Closed Under Concatenation

Easy The regex for $L(\alpha) \cdot L(\beta)$ is $\alpha \cdot \beta$.

Regular Lang Closed Under $*$?

Easy The regex for $L(\alpha)^*$ is α^*.

Summary of Closure Properties and Proofs

X means **Can't Prove Easily**

$n_1 + n_2$ (and similar) is number of states in new machine if L_i reg via n_i-state machine.

$L_1 + L_2$ (and similar) is length of regex of L_i length of α_i.

Closure Property	DFA	NFA	Regex
$L_1 \cup L_2$	$n_1 n_2$	$n_1 + n_2$	$L_1 + L_2$
$L_1 \cap L_2$	$n_1 n_2$	$n_1 n_2$	X
$L_1 \cdot L_2$	X	$n_1 + n_2 + 1$	$L_1 + L_2$
$\overline{L}$	n	X	X
L^*	X	$n + 1$	$L + 1$

BILL, STOP RECORDING LECTURE!!!!

BILL STOP RECORDING LECTURE!!!