Comp 330 - Lecture 7 - September 21st

Italian expression: Parlare a vanvera
↳ "Talk nonsense"

Regular expressions

Def (Informal) Reg. exp. are mathematical objects which explicitly describe patterns in strings

Ex $\Sigma = \{0,1\}$

$r = \underbrace{(0+1)}^{*}01$

$0+1$ → the string has either a 0 or a 1

$(0+1)^*$ → _________ " _________, any # of times

$(0+1)^*01$ → the string starts w/ any # of 0s or 1s, and ends in 01

r describes strings in Σ^* ending in 01

$L(r) = \{w \in \{0,1\}^* : w \text{ ends in } 01\}$

Reg exp → Declarative

FA → Imperative

Ex $\Sigma = \{a, b\}$

$$r = (a+b)^* \, ab \, (a+b)^*$$

$$x \in \Sigma^*, \quad \underbrace{\qquad\qquad}_{\text{any \# of as or bs}} \; ab \; \underbrace{\qquad\qquad}_{\text{any \# of as or bs}}$$

$$L(r) = \{ w \in \{a,b\}^* : w \text{ contains substring } ab \}$$

Ex $\Sigma = \{0, 1\}$ Create reg. exp. r s.t.

$$L(r) = \{ w \in \{0,1\}^* : w \text{ has odd \# of 1s} \}$$

$$x = \underbrace{0\ldots 0 1 0 \ldots 0 1}\underbrace{0 \ldots 0 \; 0 \ldots 0 1}\underbrace{0 \ldots 0 1}\underbrace{0 \ldots 0 \; 0}\;\underbrace{0 \ldots 0 1 0 \ldots 0}$$

$$r = (0^* 1 0^* 1 0^*)^* \, 0^* 1 0^*$$

$$\not\equiv (\qquad\qquad)^* \boxed{1 0^*}$$

$x = 1 \quad 1 \quad 1 \qquad 0 0 .. 0 1 0 \ldots 0$

Def (Reg Exp – Stephen Kleene – 1951)
Given $\Sigma \neq \emptyset$, a valid reg. exp. consists of atomic reg. exp. joined together with reg. exp. operators.

Atomic reg. exp. :

① ϕ

② ε

③ a, $a \in \Sigma$

①, ②, ③ have Type reg. exp.

Given valid reg. exp. r_1, r_2, The reg. exp. operators are the following:

① $r_1 + r_2$

② $r_1 \cdot r_2$

③ r_1^*

④ (r_1)

Ex $\Sigma = \{a, b\}$

$$r = (a+b) \cdot (\overset{\overset{ba}{\downarrow}}{b \cdot a} + \phi^*)$$ Valid reg. exp.

$$r = (a \cap b) - (b^R \cdot \phi)$$ Not valid

Remark / Fact Reg. exp. describe regular languages.

How do we determine $L(r)$?

Recursive procedure:

① Languages described by atomic reg. exp.

$$L(\phi) = \phi$$

$$L(\varepsilon) = \{\varepsilon\}$$

$$L(a) = \{a\}$$

② We map reg. exp. operators to language operators : r_1, r_2 valid reg. exp.

$$L(r_1 + r_2) = L(r_1) \cup L(r_2)$$
$$L(r_1 \cdot r_2) = L(r_1) \cdot L(r_2)$$
$$L(r_1^*) = (L(r_1))^*$$

Order precedence of reg. exp : $(\,) \to * \to \cdot \to +$

$$a^* b + a \to ((a)^* \cdot b) + (a)$$
$$\not\to a^* \cdot (b + a)$$

Ex What language is described by $ab^* + b^*$?

$$L(ab^* + b^*) = L(a \cdot b^*) \cup L(b^*)$$
$$= L(a) \cdot L(b^*) \cup (L(b))^*$$
$$= L(a) \cdot (L(b))^* \cup (L(b))^*$$
$$= \{a\} \cdot \{b\}^* \cup \{b\}^*$$
$$= \{ab^n : n \geq 0\} \cup \{b^n : n \geq 0\}$$

$$aab^* \to (aa)(b)^*$$

Ex $\Sigma = \{a, b\}$

$$L(a^* + b^*) = \{a^n : n \geq 0\} \cup \{b^n : n \geq 0\}$$
$$L((a+b)^*) = \Sigma^*$$

In general, $(r_1 + r_2)^* \neq r_1^* + r_2^*$

Kleene algebra:

Ex $\Sigma = \{a, b\}$. Design a reg. exp. r s.t. $L(r) = \{ w \in \{a,b\}^* :$ every a in w is followed by at least one $b \}$

bbababb ✓

b b b ✓

bbba ×

+ ① b^*

② $\underbrace{b \dots b}_{\text{optional}} \; ab \; \underbrace{b \dots b}_{\text{optional}} \; ab \; \underbrace{b \dots b}_{\text{optional}}$

$b^*(abb^*)^*$

$r = b^* + b^*(abb^*)^*$

$\equiv b^*(abb^*)^*$

Thm Given Σ, the family of languages described by reg. exp. $\mathcal{L}_{REX} = \{ L(r) :$ r is a valid reg. exp. over $\Sigma \}$ is equal $\mathcal{L}_{REG}$.

Implication: Proving REG DFA, NFA, NFA+ε, Closure properties, REX.

Σ

Ex (Finite language $L = \{a_1, a_2, \dots, a_n\}$
$\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad\quad \langle a_i \in \Sigma^*$

$$r = a_1 + a_2 + a_3 + \dots + a_n$$

$$L_{REX} = L_{REG}$$

Proof $L_{REX} \subseteq L_{REG}$

① Show atomic reg. exp. denote reg. lang.

Atomic reg. exp. r	$L(r)$	FA
ϕ	$\{\}$	$\to \bigcirc$
ε	$\{\varepsilon\}$	$\to \circledcirc$
a	$\{a\}$	$\to \bigcirc \xrightarrow{a} \circledcirc$

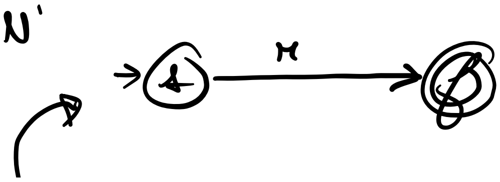
② The language denoted by any reg. exp. can be found by taking

language operators → $[\cup, \cdot, {}^*]$ of the languages denoted by atomic reg. exp. By closure properties, these lang. are also reg!

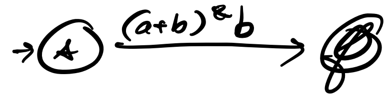
$L_{REG} \subseteq L_{REX}$. Create conversion algo. from FA to REX

The goal of this algo is to "shrink" any FA N s.t. it looks like the following GFA (generalized FA)

N'



At this point, the equivalent reg. exp of N' (N) is r.

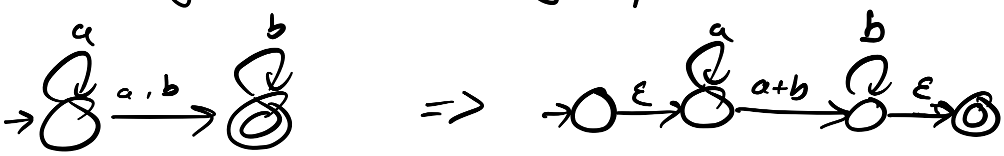


$\rightarrow \quad r = (a+b)^* b$

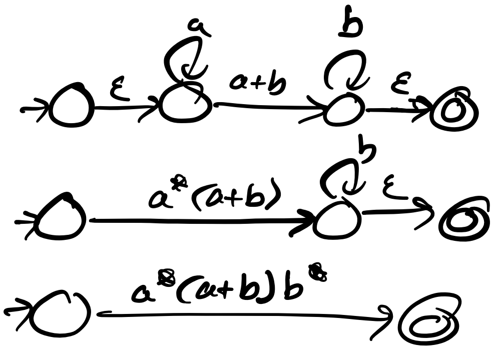
Input: FA N

1. Convert N to a GFA N'
   - (A) 1 start state w/ only outgoing edges
   - (B) 1 accept state w/ only incoming edges
   - (C) Edge labels are reg. exp.

NFA N

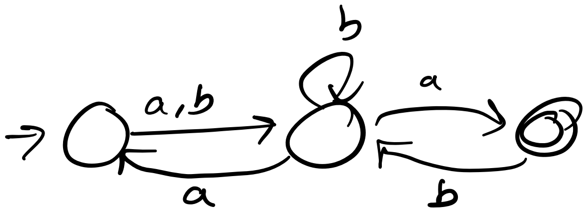


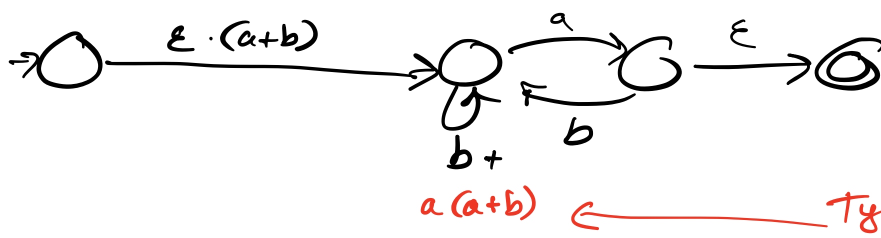
2. Rip out intermediate states until ○→◎ without losing any information.



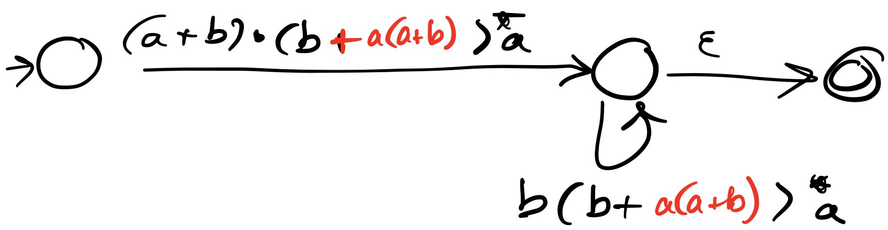
$r = a^*(a+b)b^*$

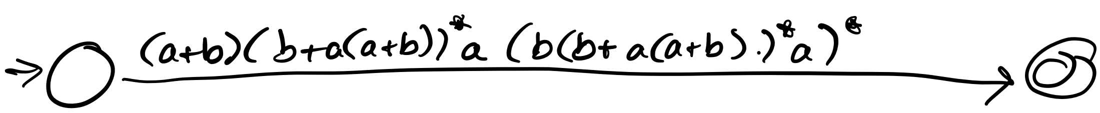
Ex





Typo in class, you read the a first.





$$r = (a+b)(b+a(a+b))^{*}a\,(b(b+a(a+b))^{*}a)^{*}$$

I will post more complete alg instructions, + extra examples, + argument of correctness