COMS 3261 : Theory of Computation

Lecture 3: Closure of the regular languages under regular operations $(\cup, \circ, *)$

Intro to regular expressions.

Announcements: HW 1 due 7/5 @ 11:59

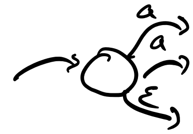
HW 2 (short) due Monday 7/12 @ 11:59 (EST)

Today:

1. Review
2. Proofs of closure: regular languages closed under $\cup, \circ, *$

3. Regular Expression
4. Regular Expressions describe regular languages (given any reg. expression, show how to build an equivalent NFA.)

Last week:

- CS theory ≈ formal science on computation.
- Language = set of strings ≈ concept
- Automata. read in input strings and accept/reject.
  - ↳ DFAs. 
  - ↳ NFAs. transition from one state ↳ a set of states. 

– Regular languages = recognized by a DFA
= recognized by an NFA.

– Proof structure:
Want to show 'if this, then that'
(A regular, B regular $\to A \cup B$ regular)
(A recognized by NFA $\to A$ recognized by some DFA.)

Strategy: Suppose we have <u>these things</u>.
We can use them to build <u>those things</u>.

$\circledast$ proof plan B

– Regular operations.
$\cup$, $\circ$, $*$: take languages $\to$ new languages
(Proved that regular languages are closed under union.)

To prove regular languages closed under union:
Simultaneously simulated DFAs for A, B and made a new DFA that accepted if either simulated machine accepted.

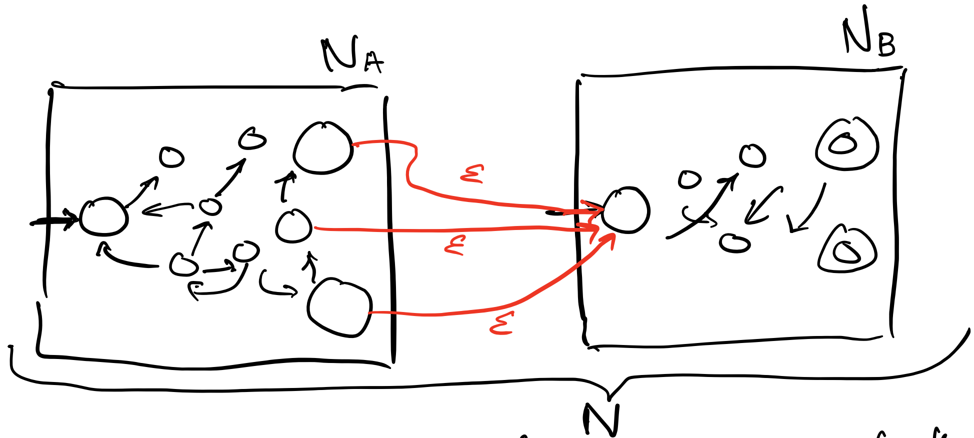
Concatenation? ($\circ$). Can't do this.

$A \circ B: \{ w \mid w = xy,\ x \in A,\ y \in B \}.$

<u>Theorem</u>: The class of regular languages is closed under concatenation.

<u>Idea</u>: Build a machine that reads in a string w, nondeterministically guesses how to split it into substrings x and y, and accepts if and only if $x \in A$, $y \in B$ for regular languages A and B.

Proof. Suppose A and B are regular languages recognized by the NFAs N_A and N_B, respectively. We'll show a new NFA, N, that recognizes $A \circ B$.



1. Given each accept state of N_A an ε-arrow to the start state of N_B.
2. Turn the accept states of N_A into regular states.
3. N is the resulting NFA (with same states, start state from N_A alphabet, and accept states.

Claim: N accepts string $w \iff w = xy$ for some $x \in A$, $y \in B$.

$\Leftarrow$.) Suppose $w = xy$ where $x \in A$, $y \in B$. Some branch of computation reaches a (former) accept state of N_A after reading in x. That branch then takes an ε-edge to the (former) start state of N_B. N_B accepts on y.

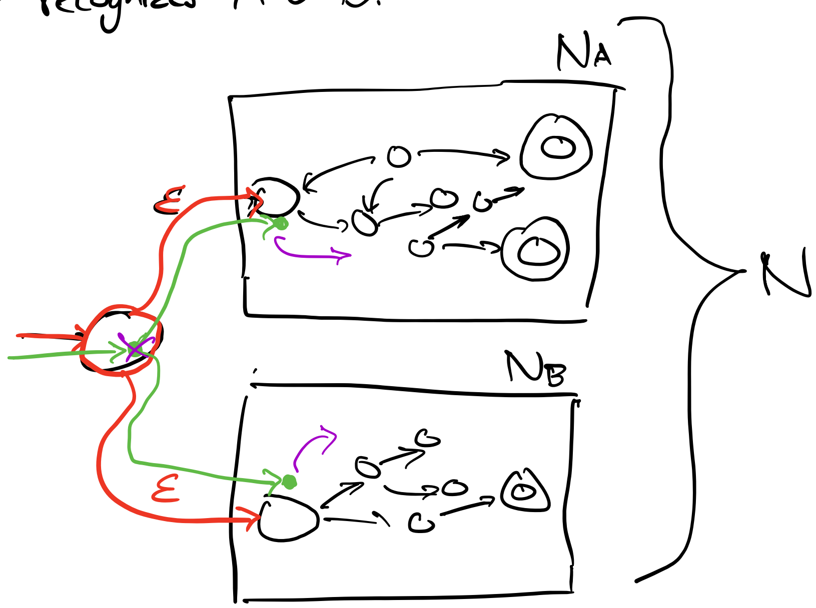
$\Rightarrow$.) N accepts w. Then there must exist some branch of computation that reaches an ~~se~~ (former) accept state of N_A. By definition, there must exist a set of states

$$r_1, r_2, \dots, r_{(N_B)}, \dots, r_m$$

that tracks our accepting branch, and where $r_{(N_B)}$ is the start state of N_B. We have $r_1 \dots r_{(N_B)-1}$ is an accepting computation for N_A, $r_{(N_B)} \dots r_m$ for N_B, and the strings corresponding to these computations are $x \in A$, $y \in B$.

<u>Theorem:</u> (already proved.) The regular languages are closed under union. ($\cup$).

<u>Proof by picture:</u> Suppose we have regular languages A and B recognized by NFAs N_A and N_B. Build an NFA N that recognizes $A \cup B$.



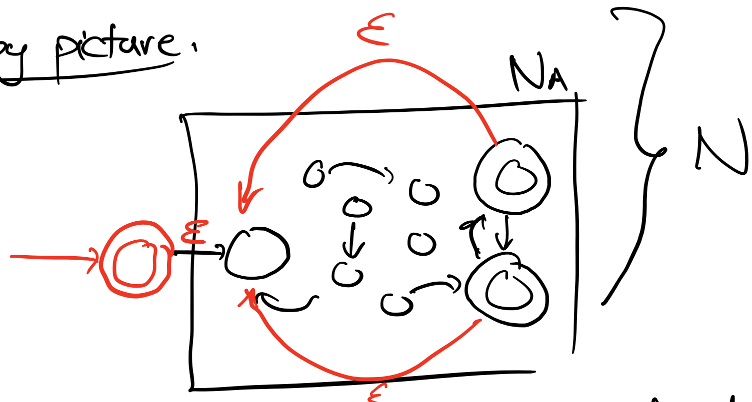
Create N by creating a new start state and ε-branches to the start states of N_A and N_B. $\square$

<u>Theorem:</u> The class of regular languages is closed under star ($*$).

Proof by picture. (Recall: $A^* = \{x_1 x_2 x_3 \dots x_k \mid k \geq 0, x_i\text{'s} \in A\}$)

Let N_A be an NFA that accepts A. We'll build an NFA N that accepts A^*.

Proof by picture.



1. Create ε-edges from accept states to start state.
2. Make sure we accept ε by adding a new start/accept state connected by an ε-edge.

Punchline: If we know that some set R of languages is regular, so are all languages we can make by using $\cup$, $\circ$, $*$.

$$A, B, C \text{ regular} \Rightarrow$$

$$A \cup B, \quad B \circ C \text{ regular}$$

$$(B \circ C)^* \cup A \text{ regular}$$

// Break — Back at 11:15

Next up: regular expressions.

Regular Expressions:

Idea: we can use the regular operations to build up expressions that represent languages.

star

Example. $(0 \cup 1)0^*$

the set containing $\{0\}$ union   (implicit) concatenation

Read this as: "the set of all strings that consist of either 0 or 1, followed by some non-negative number of zeroes."

We say that a regular expression <u>evaluates</u> to the language of strings it describes.

$$(0 \cup 1) = \{0, 1\}$$

Example: $(0 \cup 1)^* = \{\text{all binary strings}\}$

<u>Idea:</u> Our definition of a regular expression will be inductive.
≈ we'll recursively define how to build one.

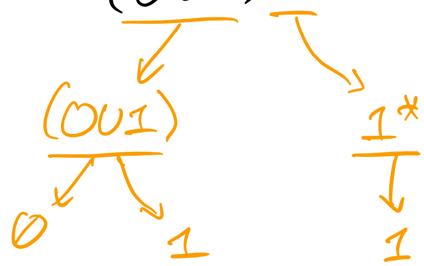
<u>Def.</u> (Regular Expression.) We say that R is a regular expression if it fits one of six cases:

- $R = a$, for some symbol a in an alphabet Σ. $(\{a\})$

- $R = \varepsilon$. $(\{\varepsilon\})$
- $R = \emptyset$. $(\{\})$
- $R = R_1 \cup R_2$, where R_1, R_2 are regular expressions
- $R = R_1 \circ R_2$, where R_1, R_2 regular expressions
- $R = R_1^*$, where R_1 is a regular expression.

$$R_1 = \underline{R_1 \circ \varepsilon} = R_1 \cup \emptyset$$

$$(0 \cup 1)1^*$$



<u>Shorthand</u>: ε, $\emptyset$

- we write Σ as a wild card – short for "any one character from the alphabet Σ."

- we write R^+ as shorthand for RR^*. "all strings consisting of <u>at least one</u> copy of a string from R.

$$RR^* = R^+$$

- we write R^k as shorthand for "all strings consisting of k concatenated strings from R."

$$\{0,1\}^k$$

Order of operations:

$R^+ \to RR^*$

1) Star and plus
2) concatenation
3) union.

$(0 \cup (1(0^*)))$

$0 \cup (10^*)$

Examples: alphabet is $\Sigma = \{0, 1\}$

$0^*10^* = \{w \mid w$ consists of some # of 0's, a 1, and then some # of 0's$\}$.

$\{w \mid w$ contains ~~exactly~~ one 1$\}$

$1^*(01^+)^* =$

some tests:

11101101

01

0111

10111101

ε

1111

$(01^+)^*$

$\{01, 011, 0111, 01111, \}$
$\vdots$

$((0^+)^*)$

$\{w \mid$ every 0 in w is followed by at least one 1$\}$.

$(\Sigma\Sigma\Sigma)^*$

$\{w \mid |w|$ is divisible by 3$\}$.

000
001
010
⋮

$1 \cup 1 = 1$

$$\underbrace{0\Sigma^*0}_{0\cdots0} \cup \underbrace{1\Sigma^*1}_{1\cdots1} \cup \underbrace{0}_{0} \cup \underbrace{1}_{1}$$

$= \{w \mid$ w starts and ends with the same symbol. $\}$

$$(0 \cup \varepsilon)(1 \cup \varepsilon) = \{01, 0, 1, \varepsilon\}$$

$1^*\emptyset = \emptyset$. Concatenate any string w/ empty language = empty language.

$$\emptyset^* = \{\varepsilon\}.$$

Let $\Sigma = \{-, ., 0, 1, \dots 9\}$, and let $D = \{0, 1, \dots 9\}$.

$$\underbrace{(\varepsilon \cup -)}_{\text{negative?}}(\underbrace{D^+}_{101,\ 3,\ 0} \cup \underbrace{D^+ . D^+}_{0.9,\ 3.666})$$

Break: 12:05

Teaser:

Write down a regular expression for the language consisting of even-length substrings of 0's, separated by single 1's.

$(00)^*$

11

$((00)^*1)^*$

001
000001
001000001

$((00)^*1)^*(00)^*$

$\cup ((00)^*1)^* \cup (1(00)^*) \cup 1((00)^*1)^*$

$0^{2k}, \; k \in \mathbb{N}_{\geq 0}$

$0^8 \qquad \{w \mid w = 0^{2k}, \; k \in \mathbb{N}_{\geq 0}\}$

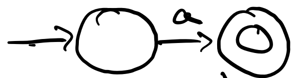
Lemma (Regular Expression $\to$ NFA.) If a language is described by a regular expression, then it is regular.

Idea: Take a generic regular expression R. We can assume that it fits our inductive definition and show how to build an NFA that captures each of our six cases.

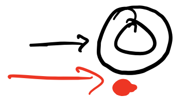
Proof: Let R be a regular expression. We'll show an NFA N that recognizes R. According to our definition, there are six forms R can take.

1. $R = a$, for some $a \in \Sigma$. Then $L(R) = a$,

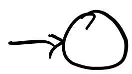
and the following NFA is equivalent:



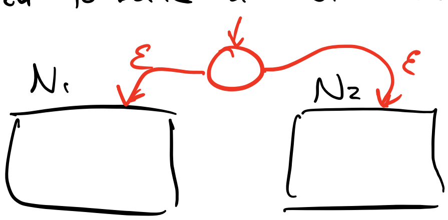
2. $R = \varepsilon$. Then $L(R) = \{\varepsilon\}$, and this NFA is equivalent:



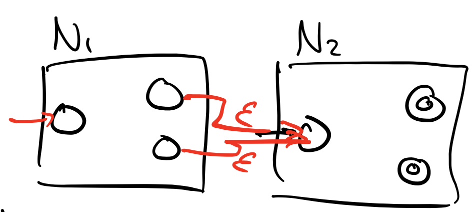
3. $R = \emptyset$. $L(R) = \emptyset$. NFA:



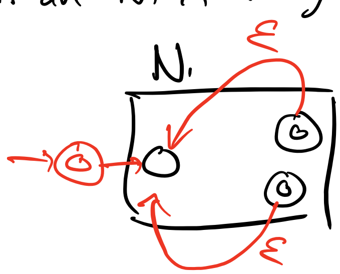
4. $R = R_1 \cup R_2$, for R_1, R_2 regular expressions. Assume $\exists$ NFA's recognizing R_1, R_2. Then we can use our union construction to build a NFA recognizing R.



5. $R = R_1 \circ R_2$, for R_1, R_2 regular expressions: Then if $\exists$ NFA's recognizing R_1, R_2, there exist NFA's recognizing $R_1 \circ R_2$



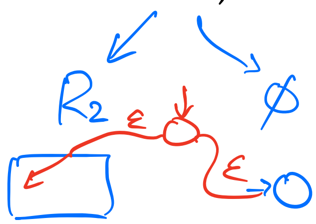
6. $R = R_1^*$, where R_1 is a regular expression. Then there exists an NFA recognizing R_1^*:



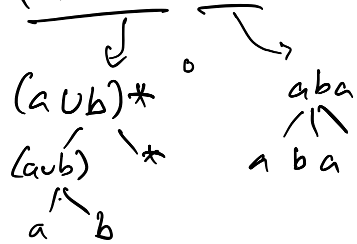
Given any regular expression, we can recursively draw an equivalent NFA by following the steps above. ∎

Maybe: $R_1 \cup \varepsilon$

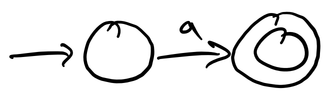
$R_2 \cup \emptyset$



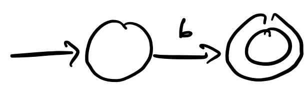
Example: Converting $(a \cup b)^* aba$ to an NFA. (On $\Sigma = \{a,b\}$.)



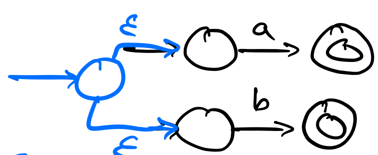
NFA for $\{a\}$:



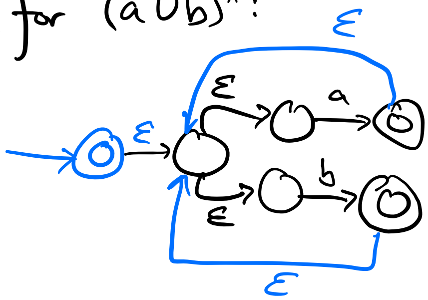
NFA for $\{b\}$:



NFA for $a \cup b$:

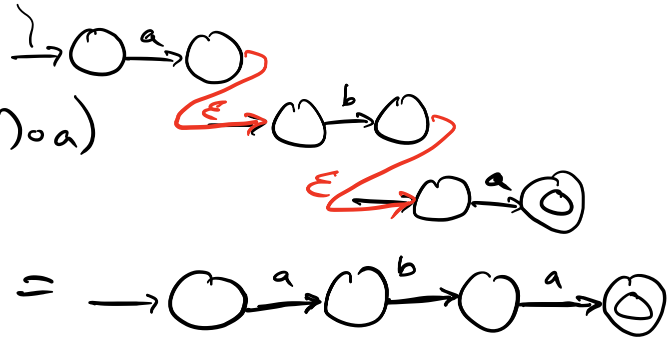


NFA for $(a \cup b)^*$:

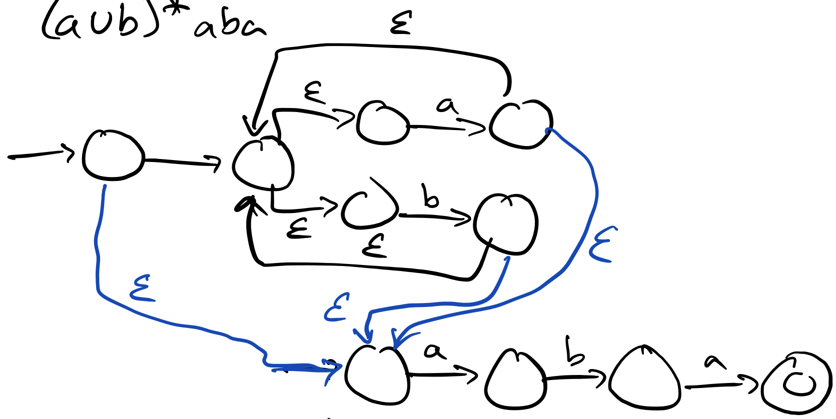


$\boxed{(a \cup b)^* aba}$

NFA for aba:
(either $a \circ (ba)$, $(ab) \circ a$)



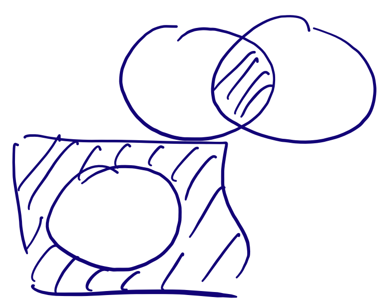
NFA for $(a \cup b)^* aba$



Reading: Sec. 1.2 (end)
1.3 (reg. expressions.)

$A \cap B$

$\neg A$



$$A \cap B = \neg(\neg A \cup \neg B)$$

$(01)^* \setminus 01$