BU CS 332 – Theory of Computation

Lecture 5:

- Closure Properties
- Regular Expressions

Reading:

Sipser Ch 1.2-1.3

Lecture recording viewings
8pm - 9:15 pm

Mark Bun

February 8, 2021

Last Time

- NFAs vs. DFAs
  - Subset construction: NFA -> DFA

- Intro to closure properties of regular languages

Closure Properties

Operations on languages

Let $A, B \subseteq \Sigma^*$ be languages. Define

Regular Operations

- Union: $A \cup B$ = $\{ a \mid a \in A$ or $a \in B \}$
- Concatenation: $A \circ B = \{ab \mid a \in A, b \in B\}$
- Star: $A^* = \{ a_1 a_2 \ldots a_n \mid n \geq 0$ and $a_i \in A\}$

Complement: $\bar{A}$ = $\{ a \mid a \notin A \}$

Intersection: $A \cap B$ = $\{ a \mid a \in A$ and $a \in B \}$

Reverse: $A^R = \{ a_1 a_2 \ldots a_n \mid a_n \ldots a_1 \in A\}$

Theorem: The class of regular languages is closed under all six of these operations

Proving Closure Properties

Complement

M' obtained by swapping accept/reject states of M

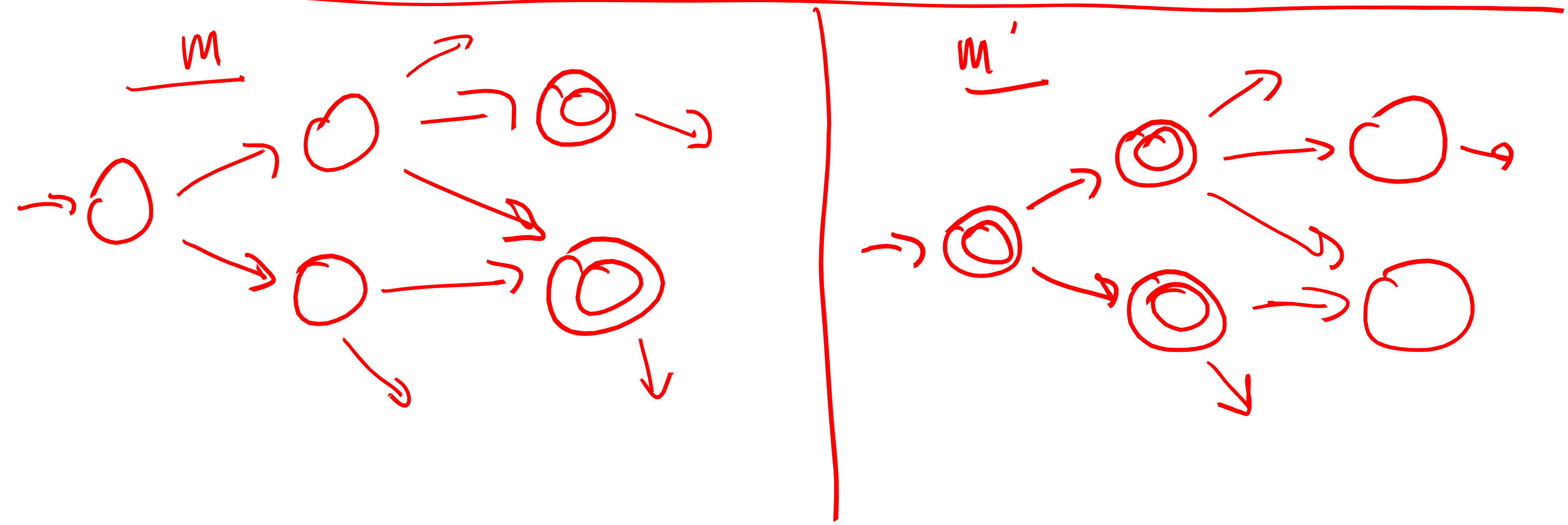
$\Rightarrow$ M' accepts w $\Leftrightarrow$ M rejects w

Complement: $\bar{A} = \{ w \mid w \notin A\}$

Theorem: If A is regular, then $\bar{A}$ is also regular

Proof idea: A regular $\Rightarrow$ $\exists$ a DFA M recognizing A

WTS: $\exists$ DFA M' recognizes $\bar{A}$ $\Rightarrow$ $\bar{A}$ is regular



Complement, Formally

Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA recognizing a language A. Which of the following represents a DFA recognizing $\bar{A}$?

a) $(F, \Sigma, \delta, q_0, Q)$

b) $(Q, \Sigma, \delta, q_0, Q \setminus F)$, where $Q \setminus F$ is the set of states in Q that are not in F

c) $(Q, \Sigma, \delta', q_0, F)$ where $\delta'(q, s) = p$ such that $\delta(p, s) = q$

d) None of the above

M: $q \xrightarrow{s} p$

M': $p \xrightarrow{s} q$

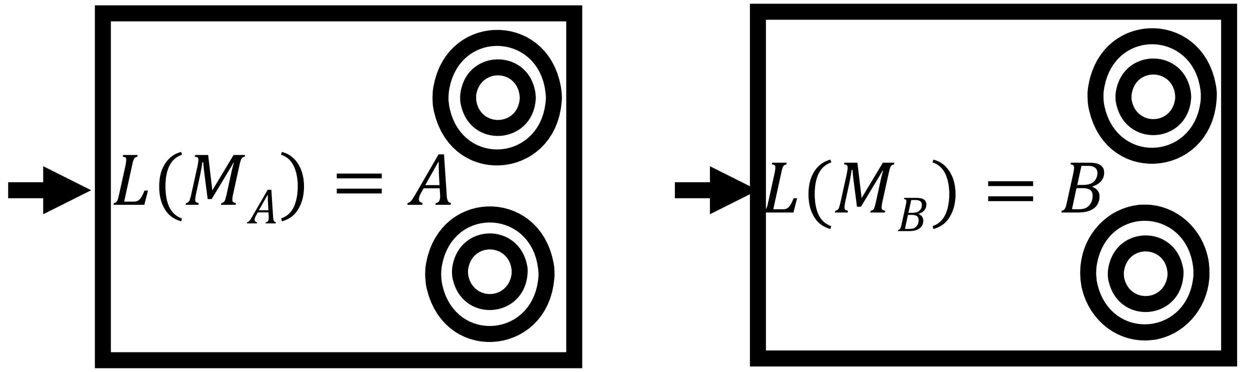
Closure under Concatenation

Concatenation: $A \circ B \; = \; \{ \, xy \, | x \in A, y \in B \, \}$

Theorem. If A and B are regular, $A \circ B$ is also regular.

Proof idea: Given DFAs M_A and M_B, construct NFA by

- Connecting all accept states in M_A to the start state in M_B.
- Make all states in M_A non-accepting.



Closure under Concatenation

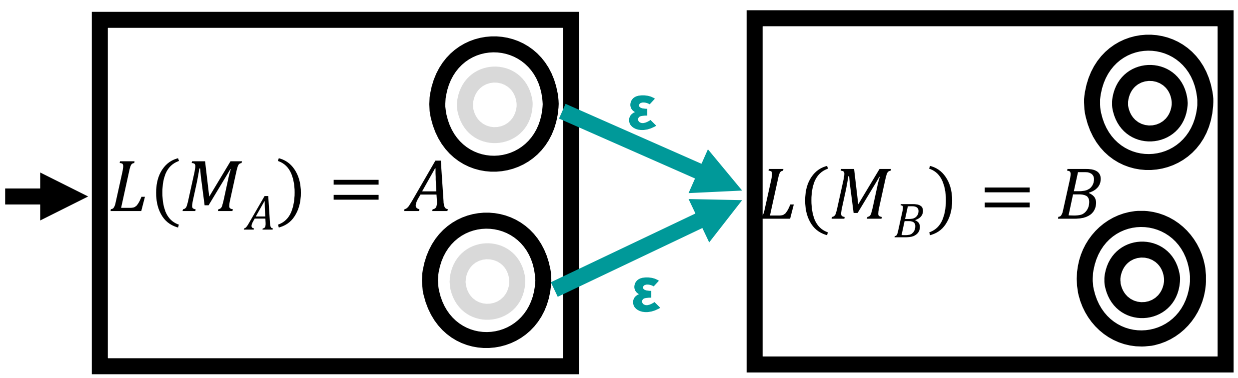
Concatenation: $A \circ B \ = \ \{ \, xy \,|x \in A, y \in B \, \}$

Theorem. If A and B are regular, $A \circ B$ is also regular.

Proof idea: Given DFAs M_A and M_B, construct NFA by

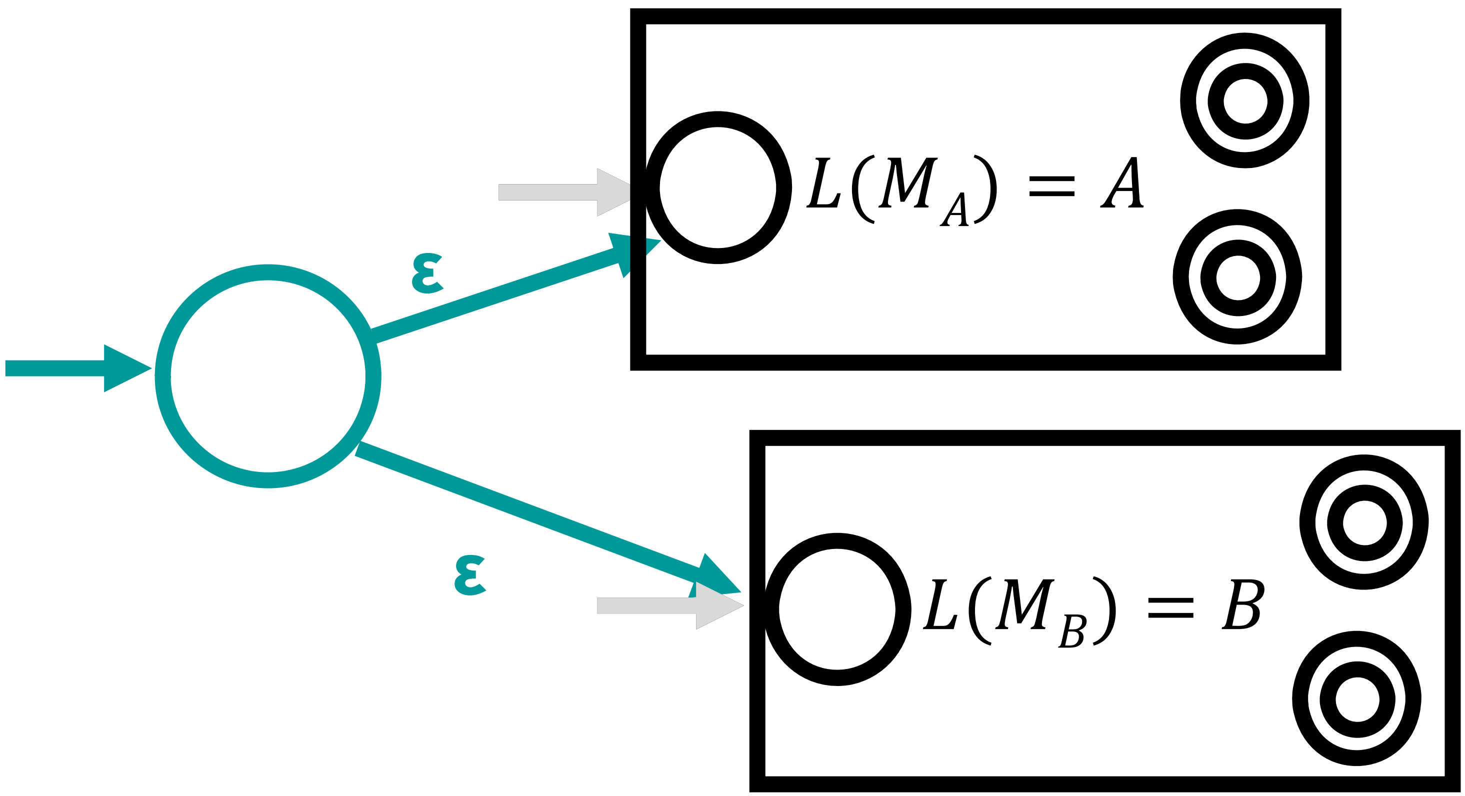
suffices to prove regularity bc NFAs = DFAs in power

- Connecting all accept states in M_A to the start state in M_B.
- Make all states in M_A non-accepting.



A Mystery Construction

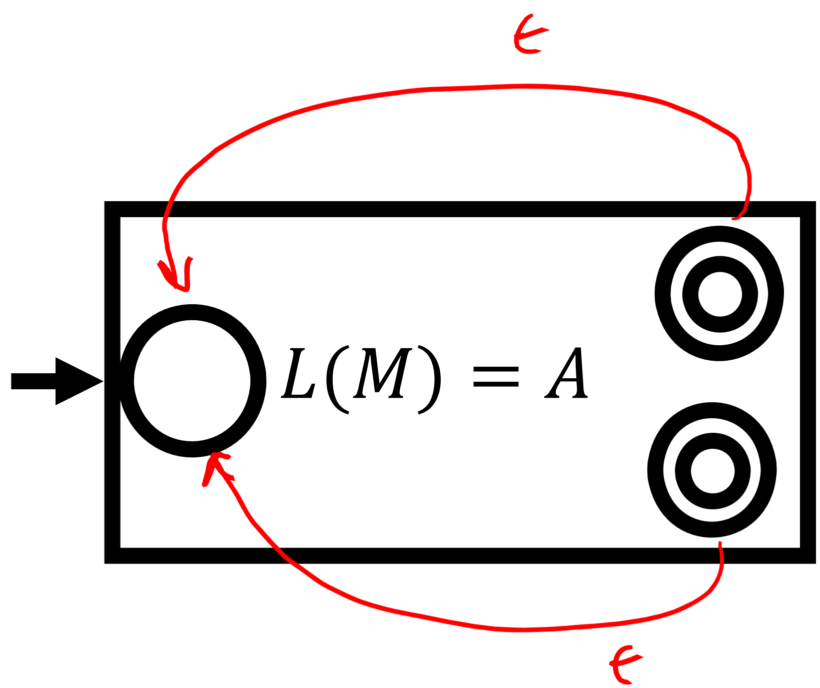
Given DFAs M_A recognizing A and M_B recognizing B, what does the following NFA recognize?



Closure under Star

Star: $A^* = \{ a_1 a_2 \ldots a_n \mid n \geq 0 \text{ and } a_i \in A\}$

Theorem. If A is regular, A^* is also regular.

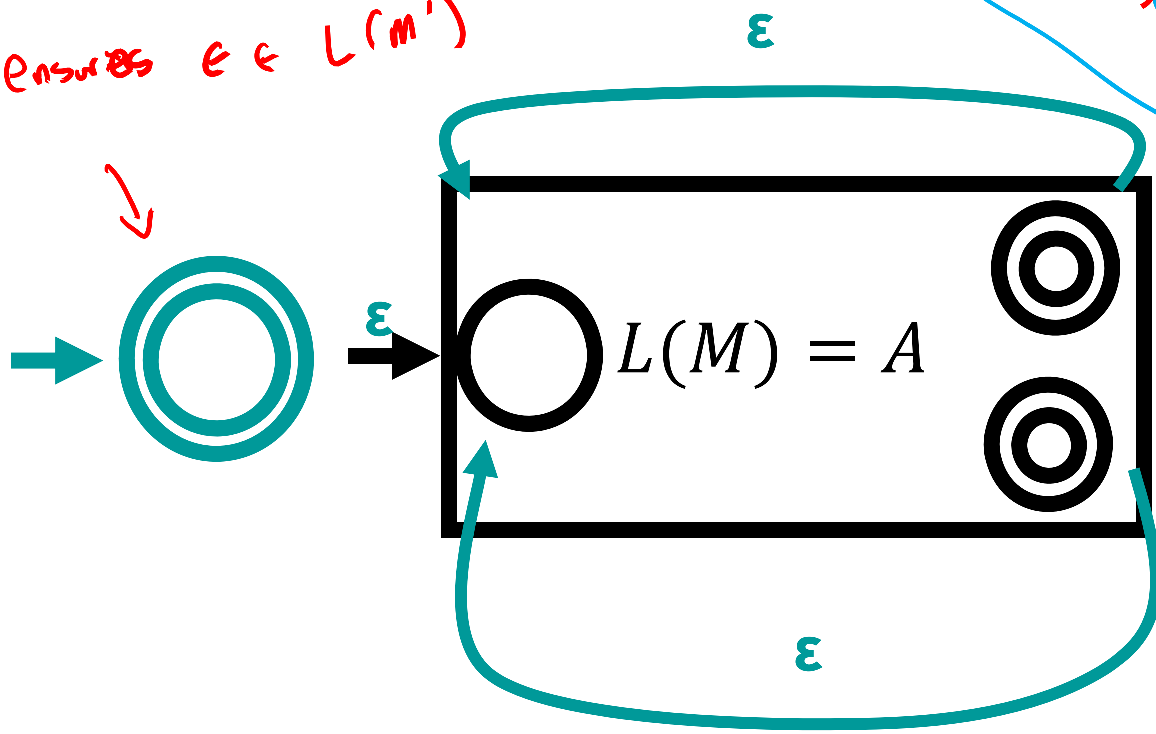


Closure under Star

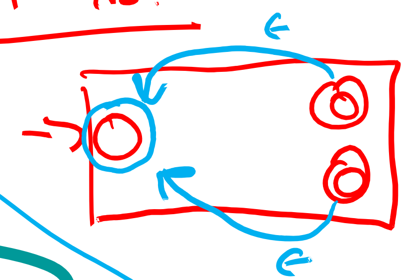
Star: $A^* = \{\, a_1 a_2 \ldots a_n \,|\, n \ \geq 0 \text{ and } a_i \in A\}$

Theorem. If A is regular, A^* is also regular.

ensures $\epsilon \in L(M')$



can't do:

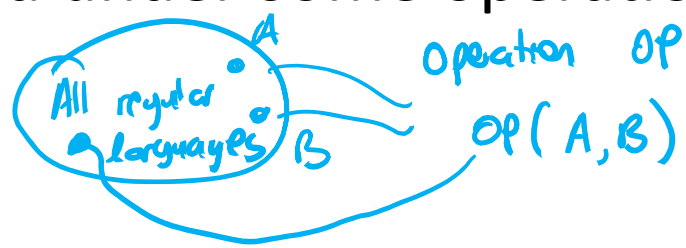


Problem: some strings not in A^* accepted

On proving your own closure properties

You'll have homework/test problems of the form "show that the regular languages are closed under some operation"

$\forall$ reg. lang A, B, $op(A, B)$ regular



What would Sipser do?

- Give the "proof idea": Explain how to take machine(s) recognizing regular language(s) and create a new machine
- Explain in a few sentences why the construction works
- Give a formal description of the construction

  5-tuple + 5-tuple = 5-tuple
- No need to formally prove that the construction works

Regular Expressions

Regular Expressions

- A different way of describing regular languages
- A regular expression expresses a (possibly complex) language by combining simple languages using the regular operations

"Simple" languages: $\emptyset, \{\varepsilon\}, \{a\}$ for some $a \in \Sigma$

Regular operations:

Union: $A \cup B$

Concatenation: $A \circ B = \{ab \mid a \in A, b \in B\}$

Star: $A^* = \{ a_1 a_2 \ldots a_n \mid n \geq 0$ and $a_i \in A\}$

Regular Expressions – Syntax

A regular expression R is defined recursively using the following rules:

1. ε, $\emptyset$, and a are regular expressions for every $a \in \Sigma$

2. If R_1 and R_2 are regular expressions, then so are
   $(R_1 \cup R_2)$, $(R_1 \circ R_2)$, and (R_1^*)

Examples: (over $\Sigma = \{a, b, c\}$)

$(a \circ b)$ $\quad$ $((((a \circ (b^*)) \circ c) \cup (((a^*) \circ b))^*))$ $\quad$ $(\emptyset^*)$

Regular Expressions – Semantics

$L(R)$ = the language a regular expression describes

1. $L(\emptyset) = \emptyset$
2. $L(\varepsilon) = \{\varepsilon\}$
3. $L(a) = \{a\}$ for every $a \in \Sigma$
4. $L((R_1 \cup R_2)) = L(R_1) \cup L(R_2)$
5. $L((R_1 \circ R_2)) = L(R_1) \circ L(R_2)$
6. $L\big((R_1^*)\big) = (L(R_1))^*$

Example: $L(((a^*) \circ (b^*))) =$

5) $L((a^*)) \circ L((b^*))$

$=$ 6) $(L(a))^* \circ (L(b))^*$

$=$ 3) $(\{a\})^* \circ (\{b\})^*$

By def of $*$

$= \{a^i \mid i \geq 0\} \circ \{b^j \mid j \geq 0\}$

$= \{a^i b^j \mid i, j \geq 0\}$

$L^* = \{a_1 a_2 \ldots a_n \mid n \geq 0, a_i \in L\}$

$= \{\varepsilon\} \cup L \cup L \circ L \cup L \circ L \circ L \cup \ldots$

Simplifying Notation

- Omit $\circ$ symbol: $(ab) = (a \circ b)$

- Omit many parentheses, since union and concatenation are associative:

$$(a \cup b \cup c) = (a \cup (b \cup c)) = ((a \cup b) \cup c)$$

- Order of operations: Evaluate star, then concatenation, then union

$$ab^* \cup c = (a(b^*)) \cup c$$

Examples

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

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

   $0^* 1 0^*$

2. $\{w \mid w$ has length at least 3 and its third symbol is $0\}$

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

3. $\{w \mid$ every odd position of w is $1\}$ $w_1 w_2 w_3 \ldots$

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

   catches 10, 11, 1011, 1010, ...

   catches odd length

Syntactic Sugar

- For alphabet Σ, the regex Σ represents $L(\Sigma) = \Sigma$

- For regex R, the regex $R^{+} = RR^{*}$

$L(R^{+}) = L(R) \cup (L(R) \circ L(R)) \cup (L(R) \circ L(R) \circ L(R)) \ldots$

Regexes in the Real World

`grep` = globally search for a regular expression and print matching lines

```
guru99@guru99-VirtualBox:~$ cat sample|grep "a\+t"
bat
eat
guru99@guru99-VirtualBox:~$
```

Not captured by regular expressions: Backreferences

Equivalence of Regular Expressions, NFAs, and DFAs

Regular Expressions Describe Regular Languages

Theorem: A language A is regular if and only if it is described by a regular expression

Theorem 1: Every regular expression has an equivalent NFA

Theorem 2: Every NFA has an equivalent regular expression

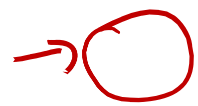
Regular expression -> NFA

Theorem 1: Every regex has an equivalent NFA

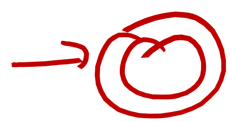
Proof: Induction on size of a regex

Base cases:

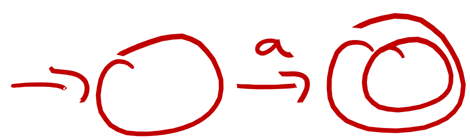
$R = \emptyset$



$R = \varepsilon$



$R = a$



Regular expression -> NFA

Theorem 1: Every regex has an equivalent NFA

Proof: Induction on size of a regex

What should the inductive hypothesis be?

a) Suppose **some** regular expression of length k can be converted to an NFA
b) Suppose **every** regular expression of length k can be converted to an NFA
c) Suppose **every** regular expression of length **at most** k can be converted to an NFA
d) None of the above

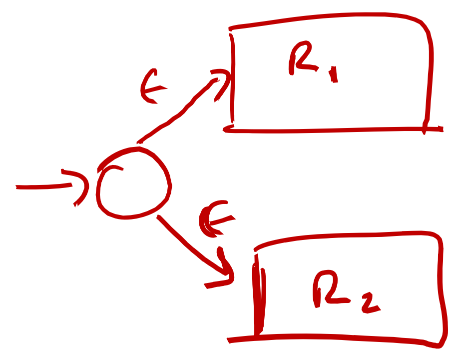
Regular expression -> NFA

Theorem 1: Every regex has an equivalent NFA

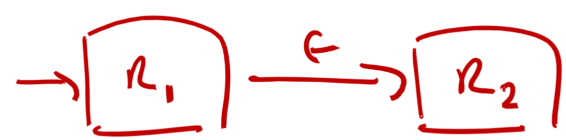
Proof: Induction on size of a regex

Inductive step:

$$R = (R_1 \cup R_2)$$



$$R = (R_1 R_2)$$



$$R = (R_1^*)$$

Example

Convert $(1(0 \cup 1))^*$ to an NFA

$\{01, 11\}^* = \{\underbrace{\epsilon}_{\text{0 copies}}, \underbrace{01, 11}_{\text{1 copy}}, \underbrace{0101, 0111, 1101, 1111}_{\text{2 copies}},$

$\underbrace{010101, 010111, \ldots}_{\text{3 copies}}, \ldots\}$

R^+ means $R R^*$ $= \{w_1 w_2 \ldots w_n \mid n \geq 1\}$

$R = \{ab, c\}$ $\quad L(R^+) = L(R) \circ \boxed{L(R^*) = \{w_1 w_2 \ldots w_n \mid n \geq 0\}}$

$= \{ab, c\} \circ \{\epsilon, ab, c, abab, abc, cab, cc, ababab, \ldots\}$

$= \{ab, abab, abc, abab, ababc, abcab, \ldots$
$c, cab, cc, cabab, \ldots\}$