BU CS 332 – Theory of Computation

https://forms.gle/5sTNDCU1QtEemHHM7

Lecture 6:

- Regexes = NFAs
- Non-regular languages

Reading:

Sipser Ch 1.3

“Myhill-Nerode” note

Mark Bun

September 22, 2022

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 $\Sigma = \{a, b, c\}$) (with simplified notation)

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

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. $L\left((R_1^*)\right) = (L(R_1))^*$

Example: $L(a^* b^*) = \{a^m b^n \mid m, n \geq 0\}$

Regular Expressions Describe Regular Languages

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

recognized by a DFA
$\Leftrightarrow$ recognized by a NFA

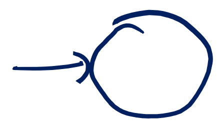
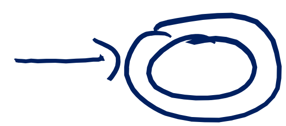
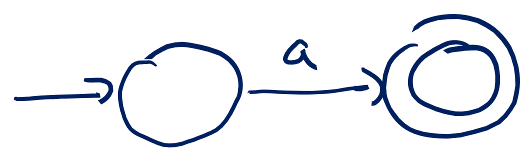
Theorem 1: Every regular expression has an equivalent NFA

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:	$L(R)$	NFA
$R = \emptyset$	$\emptyset$	
$R = \varepsilon$	$\{\epsilon\}$	
$R = a$	$\{a\}$	

Regular expression -> NFA

Theorem 1: Every regex has an equivalent NFA

Proof: Induction on $\underbrace{\text{size of a regex}}$

number of characters $(a, \emptyset, \epsilon, (,), *, \cup, \circ)$ making up the regex

What should the inductive hypothesis be?

a) Suppose **some** ($\exists$) regular expression of length k can be converted to an NFA

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

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

$\Rightarrow$ every regular expression of length $k+1$ has an NFA

Regular expression -> NFA

Theorem 1: Every regex has an equivalent NFA

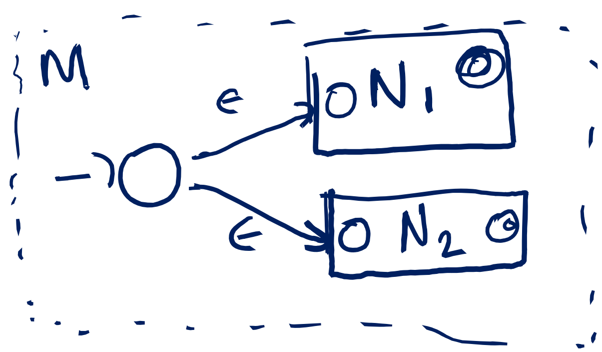
Proof: Induction on size of a regex

Assume every regex of length $\le k$ has an equiv NFA

Show every regex of length $k+1$ also has an equiv NFA

Inductive step:

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



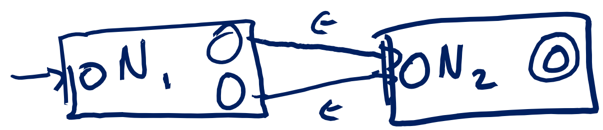
$L(N_1) = L(R_1)$

$L(N_2) = L(R_2)$

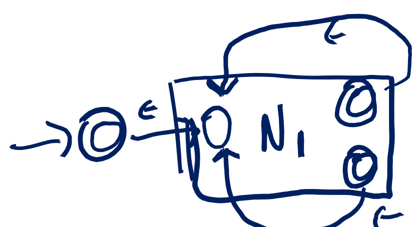
$L(M) = L(R_1) \cup L(R_2)$

$= L(R)$

$$R = (R_1 R_2)$$



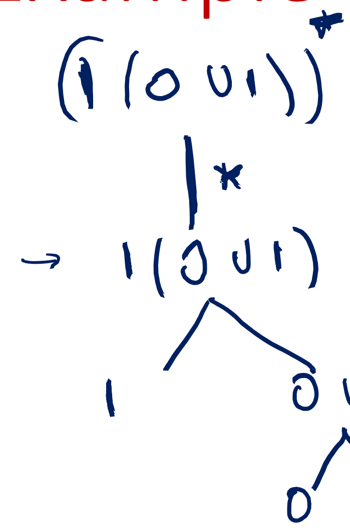
$$R = (R_1^*)$$



Example

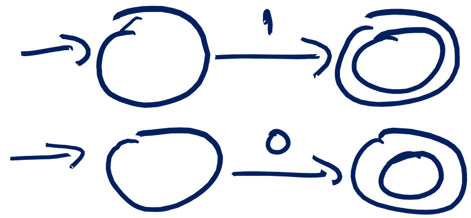
Convert $(1(0 \cup 1))^*$ to an NFA

Even length strings w/ 1s in odd positions

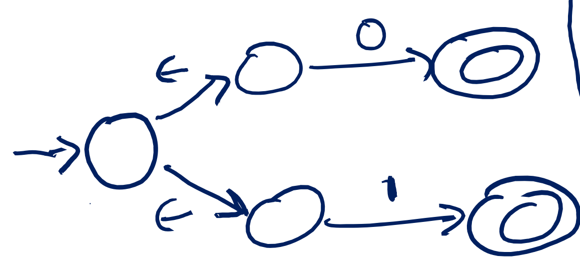


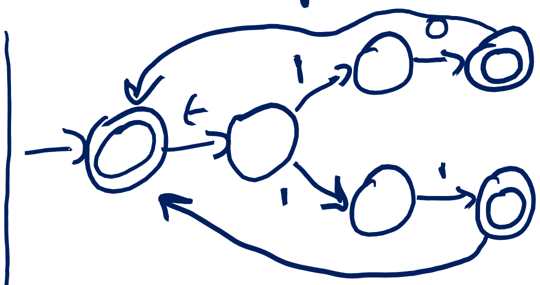
1:

0:

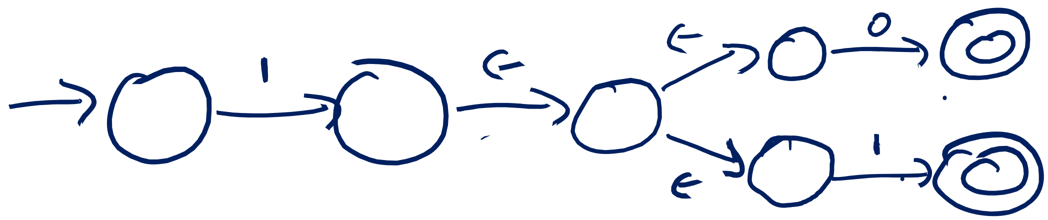


0 ∪ 1:

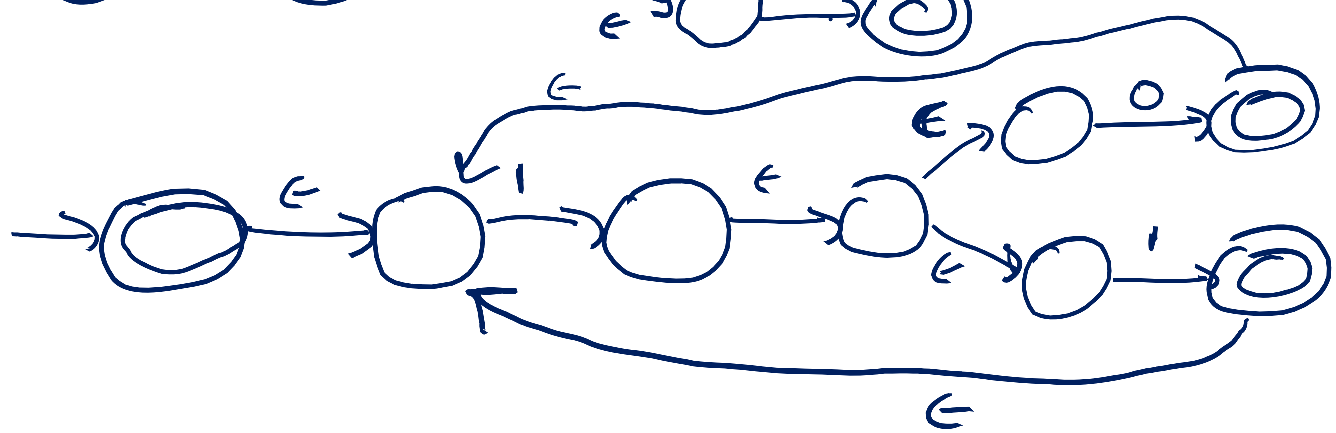




1(0 ∪ 1):



(1(0 ∪ 1))*:



Regular Expressions Describe Regular Languages

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

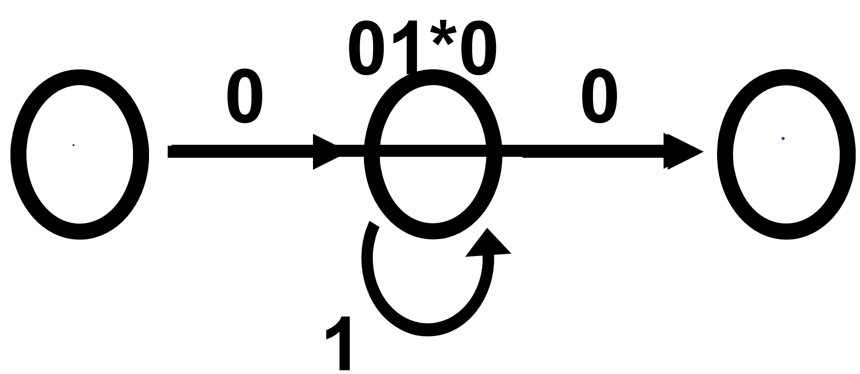
Theorem 1: Every regular expression has an equivalent NFA

Theorem 2: Every NFA has an equivalent regular expression

NFA -> Regular expression

Theorem 2: Every NFA has an equivalent regex

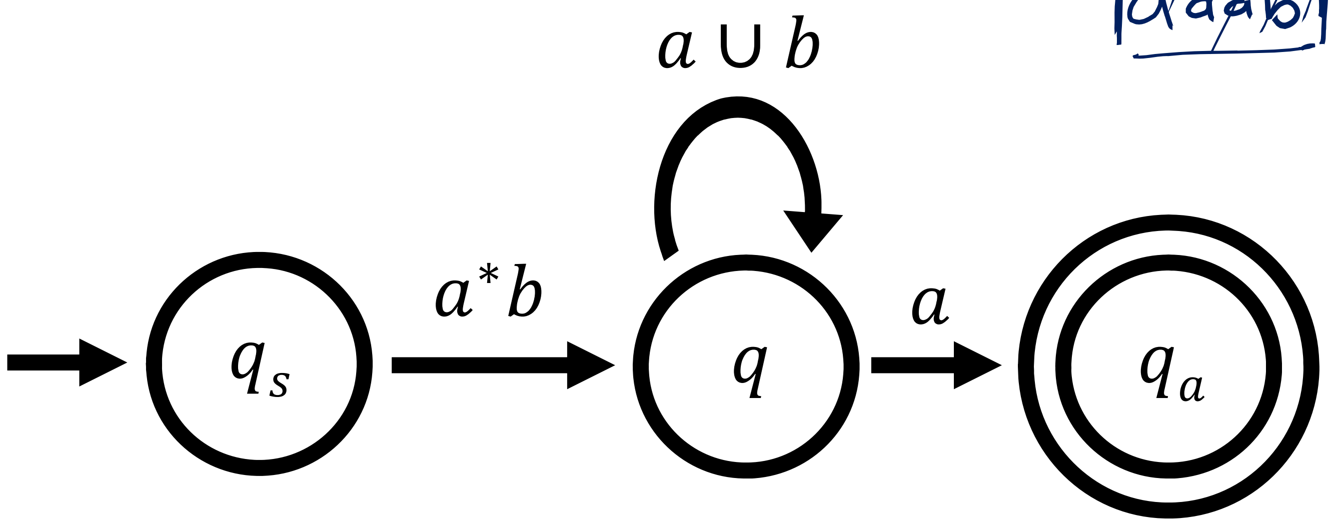
Proof idea: Simplify NFA by "ripping out" states one at a time and replacing with regexes



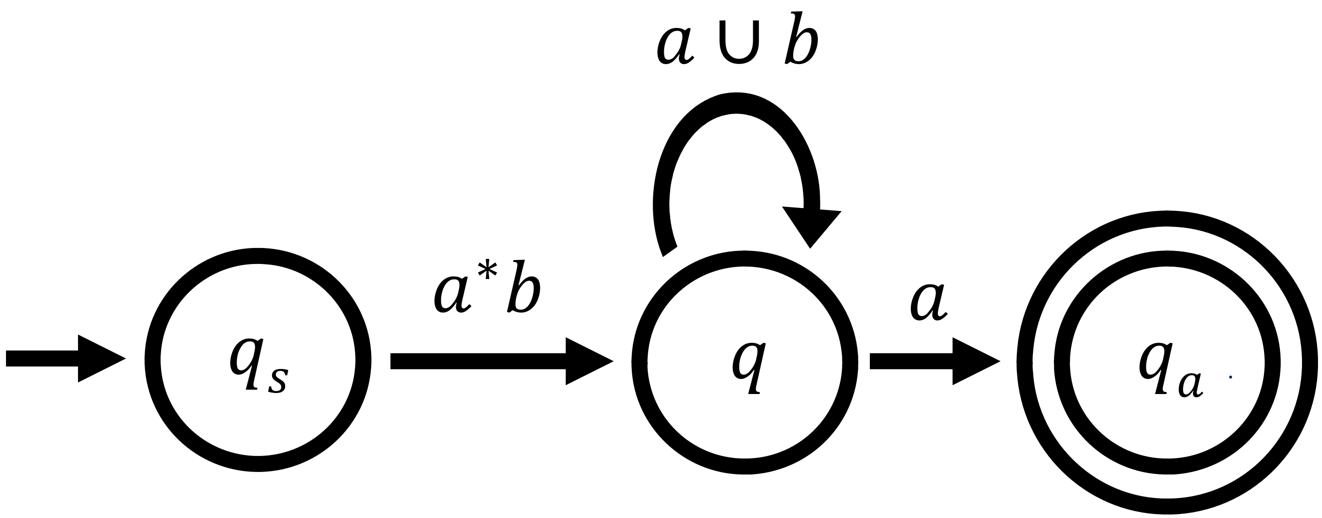
Generalized NFAs (GNFAs)

- **Every transition is labeled by a regex**
- One start state with only outgoing transitions
- Only one accept state with only incoming transitions
- Start state and accept state are distinct

aaab|ba



Generalized NFA Example



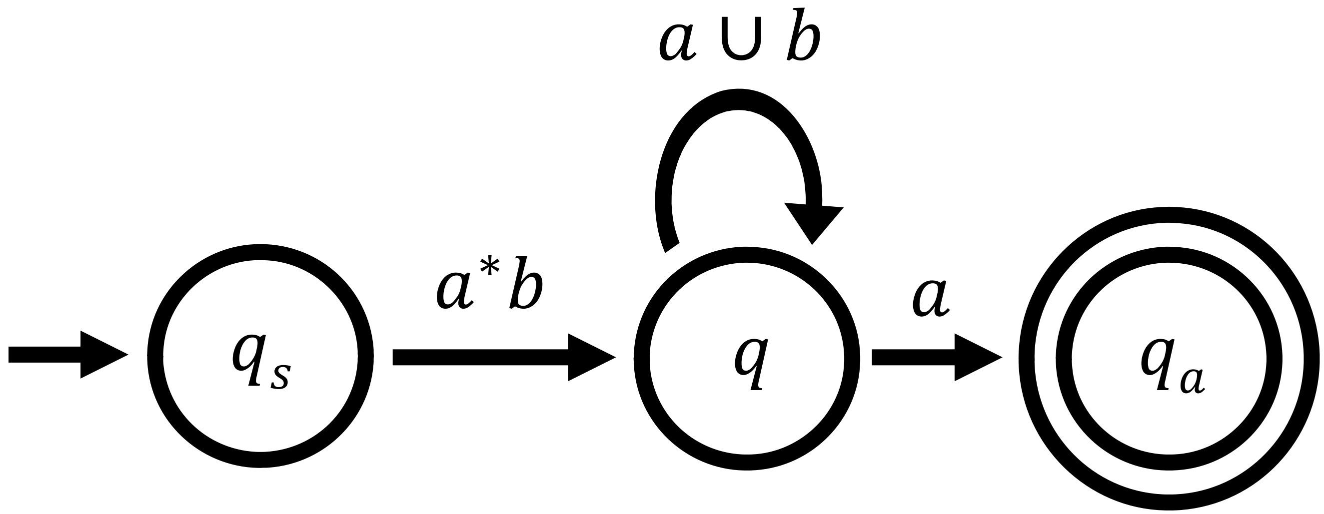
$$R(q_s, q) = a^*b$$

$$R(q_a, q) = \emptyset$$

$$R(q, q_s) = \emptyset$$

Which of these strings is accepted?

Which of the following strings is accepted by this GNFA?



~~a) aaa~~

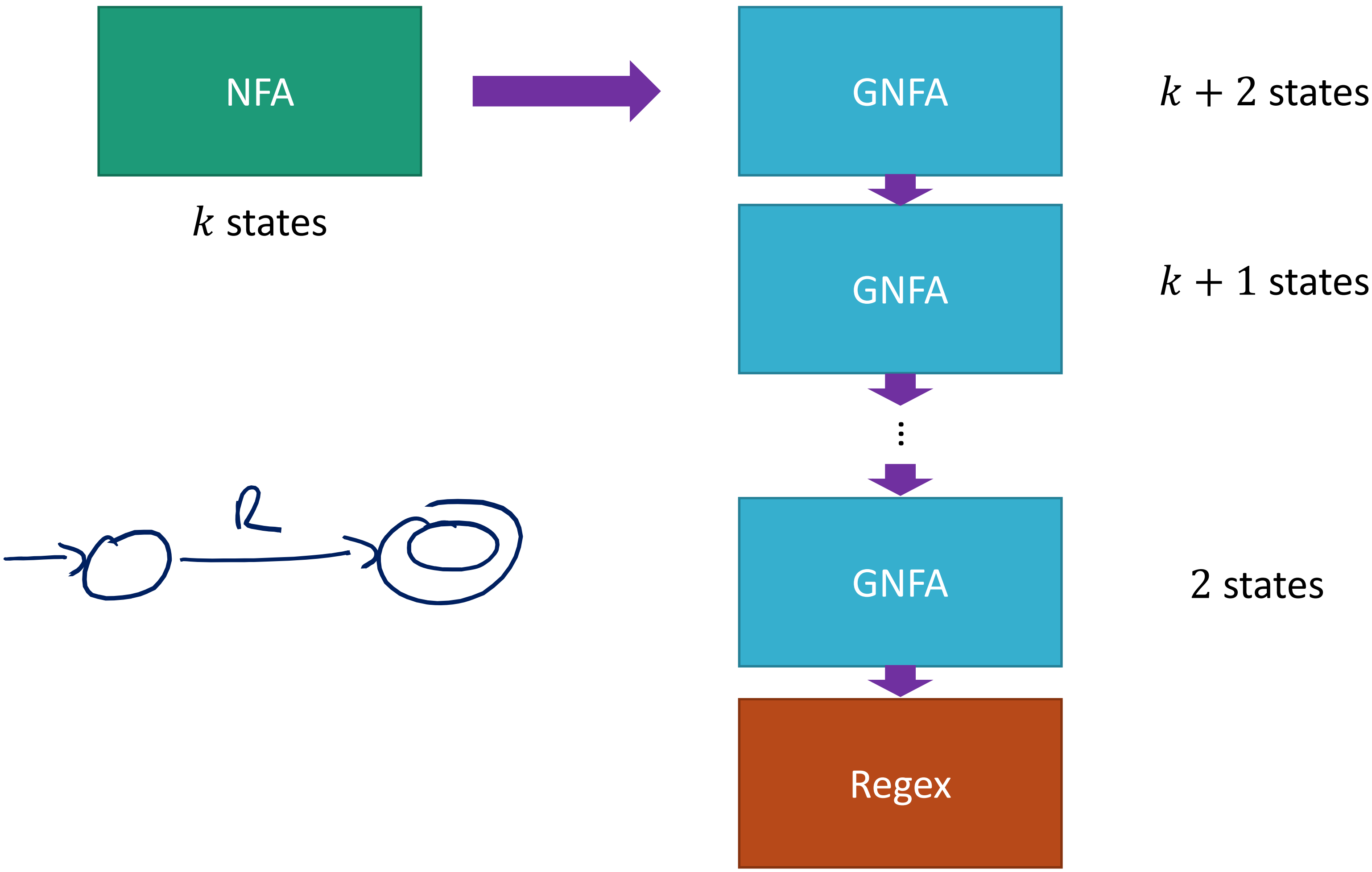
~~b) $aabb$~~

~~c) bbb~~

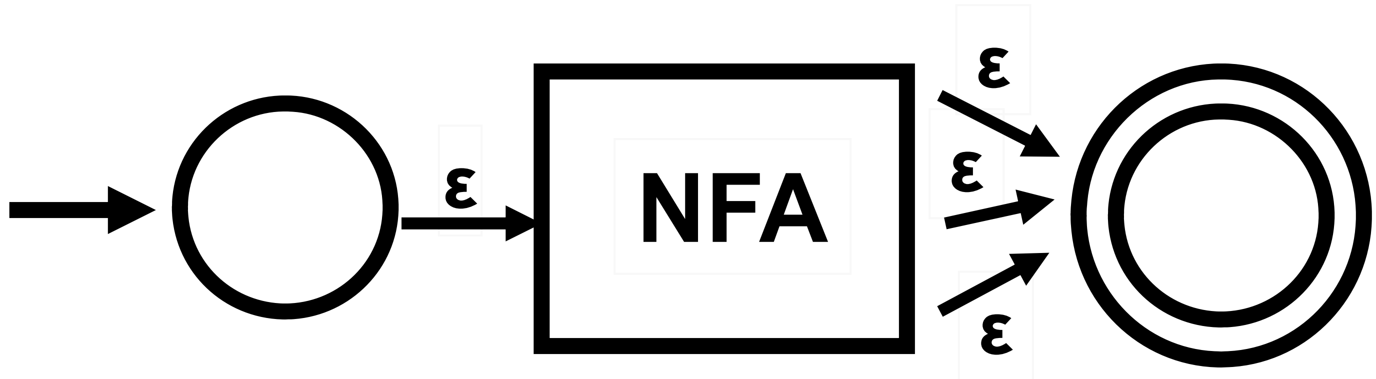
→ d) bba

$q_s \xrightarrow{b} q \xrightarrow{b} q \xrightarrow{a} q_a$

NFA -> Regular expression



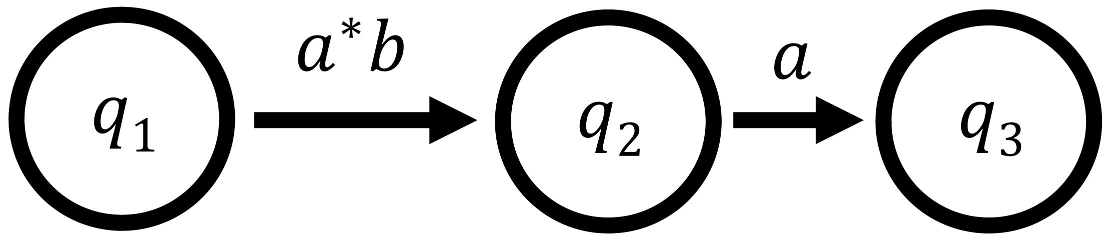
NFA -> GNFA

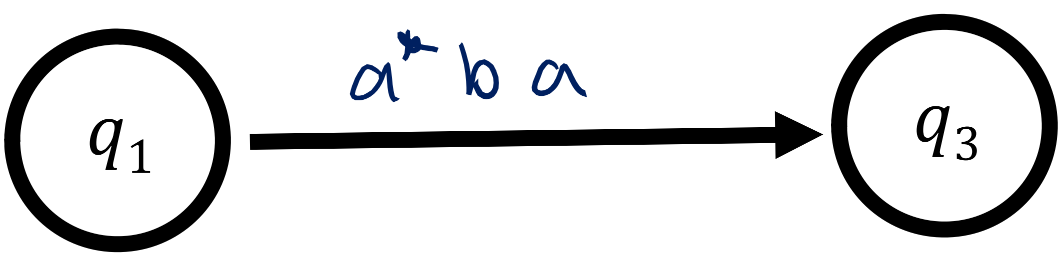


- Add a new start state with no incoming arrows.
- Make a unique accept state with no outgoing arrows.

GNFA -> Regular expression

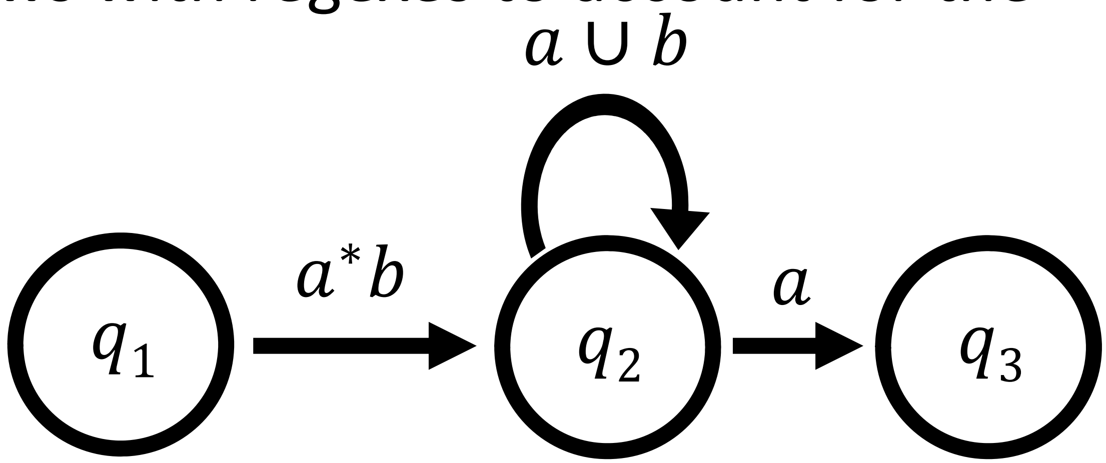
Idea: While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state



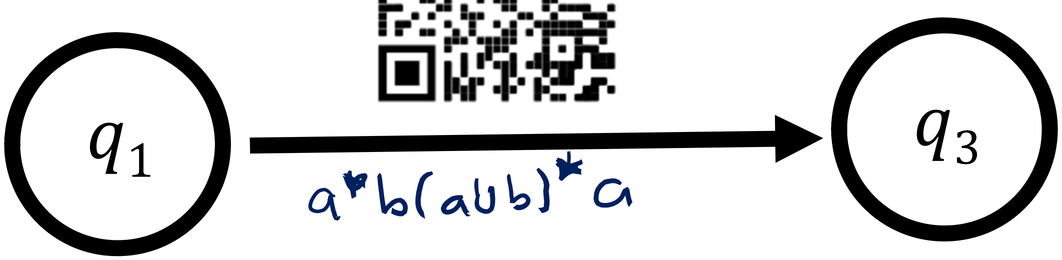


GNFA -> Regular expression

Idea: While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state

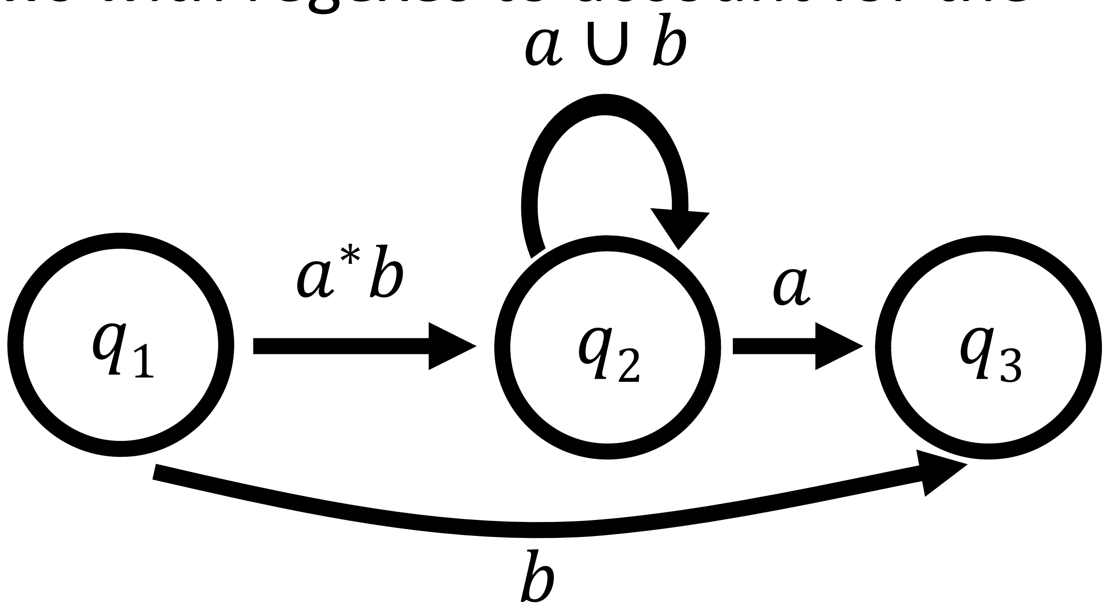


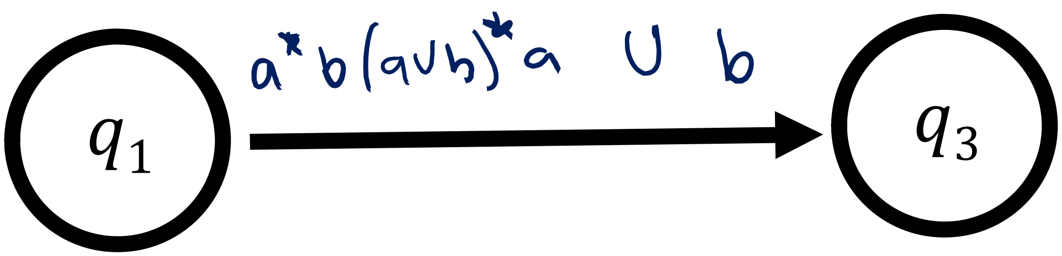
a) $a^*b(a \cup b)a$
b) $a^*b(a \cup b)^*a$
c) $a^*b \cup (a \cup b) \cup a$
d) None of the above



GNFA -> Regular expression

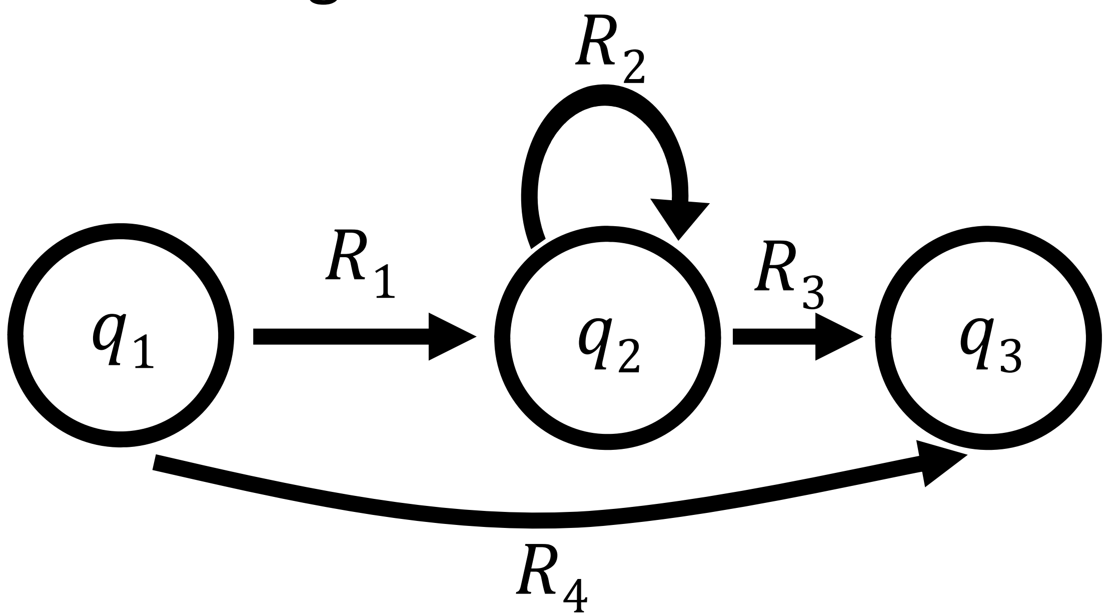
Idea: While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state

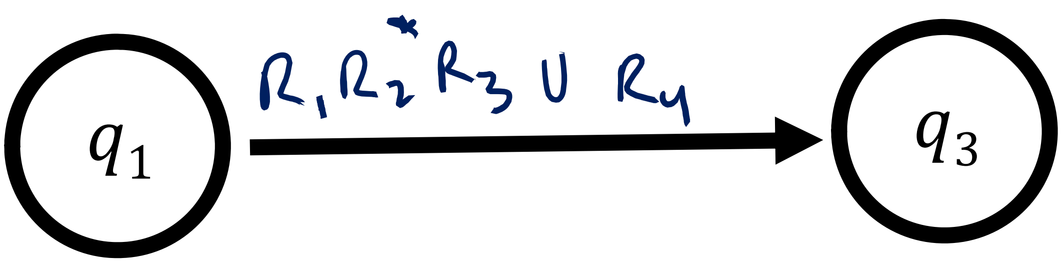


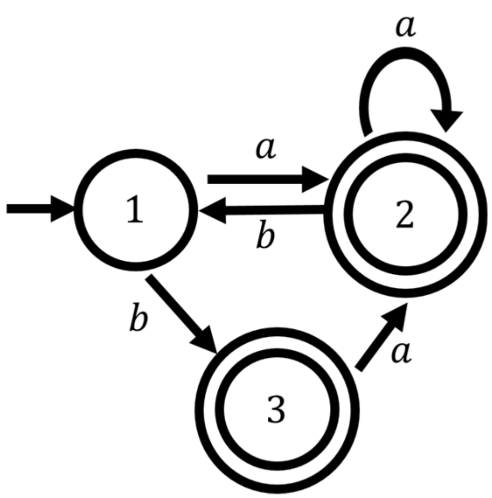


GNFA -> Regular expression

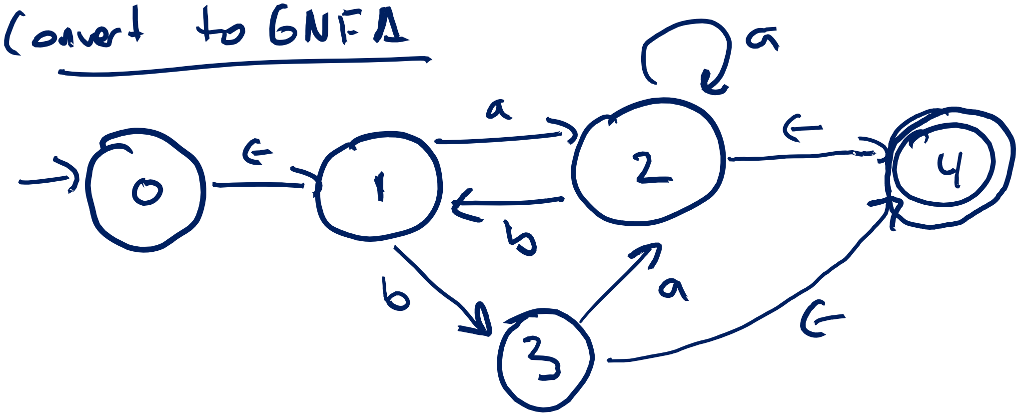
Idea: While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state



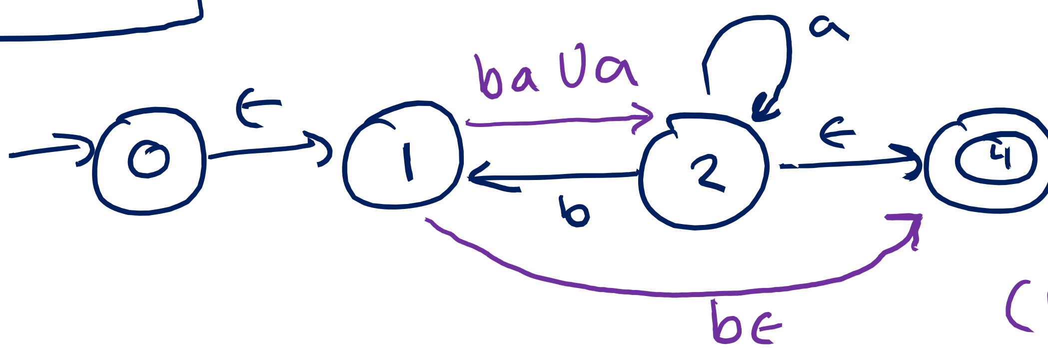




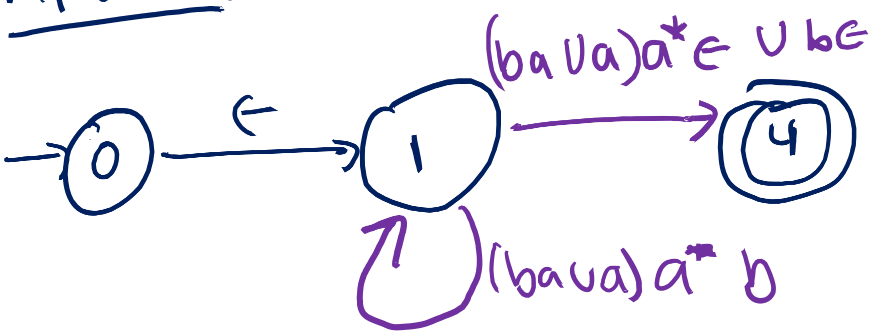
Convert to GNFA



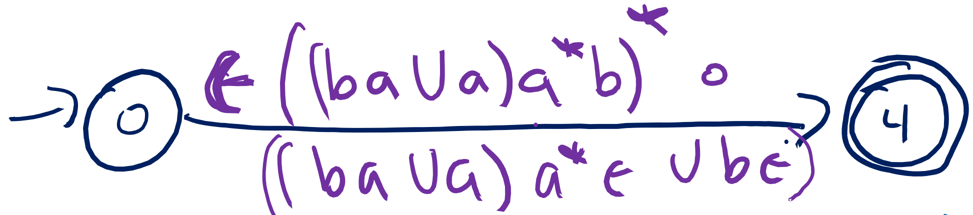
Rip out 3



Rip out 2



Rip out 1



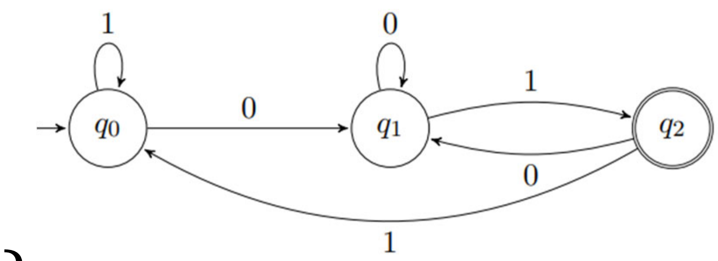
$= ((ba \cup a)a^*b)^*((ba \cup a)a^* \cup b)$

Non-Regular Languages

Motivating Questions

- We've seen techniques for showing that languages are regular
  - Construct a DFA
  - Construct an NFA
  - Construct a regular expression
  - Use closure properties
- How can we tell if we've found the smallest DFA recognizing a language?
- Are all languages regular? How can we prove that a language is not regular?

An Example



$A = \{\, w \in \{0,1\}^* \mid w \text{ ends with } 01 \,\}$

Claim: *Every* DFA recognizing A needs at least 3 states

Proof: Let M be any DFA recognizing A. Consider running M on each of $x = \varepsilon, y = 0, w = 01$

Let q_x = state M ends up in when reading x
q_y = " " y
q_w = " " w

Goal: Show q_x, q_y, q_w all different

Claim 1: $q_x \neq q_w$ and $q_y \neq q_w$

why? q_w is an accept state, but q_x and q_y are not accept states

Claim 2: $q_x \neq q_y$

why? Assume for contradiction that $q_x = q_y$

Run M on $x1 = 1 \notin L \Rightarrow q$ is a non-accept State

$y1 = 01 \in L \Rightarrow q$ is an accept state $\times$

A General Technique

$A = \{ w \in \{0, 1\}^* \mid w \text{ ends with } 01 \}$

Definition: Strings x and y are **distinguishable** by L if there exists a string z such that exactly one of xz or yz is in L.

Ex. $x = \varepsilon, \quad y = 0$

$z = 1$

$xz = \varepsilon 1 = 1 \notin L$

$yz = 01 \in L$

Definition: A set of strings S is **pairwise distinguishable** by L if every pair of distinct strings $x, y \in S$ is distinguishable by L.

Ex. $S = \{\varepsilon, 0, 01\}$