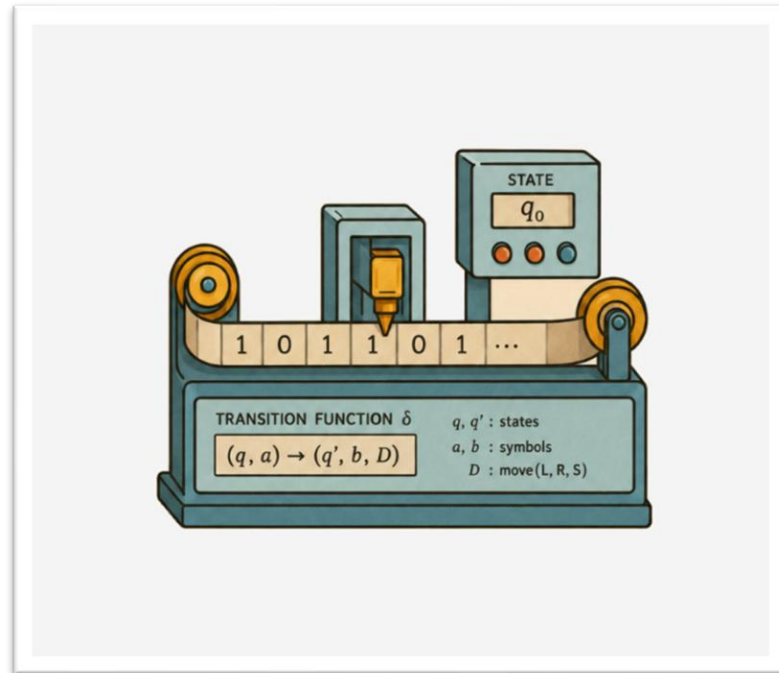


BU CS 332 – Theory of Computation

Lecture 6:

- Regexes = NFAs
- Quiz



Alexander Poremba

September 22, 2026



Recap: Languages & Regular Expressions

Regex

Regular Expressions – Syntax

A **regular expression** R is defined recursively via 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 , $ab^*c \cup (a^*b)^*$, $\emptyset$

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. $L((R_1^*)) = (L(R_1))^*$

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



regular expression



language described
by regular expression

Syntactic Sugar

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

$$\underbrace{\Sigma = \{a, b, c\}}_{\text{alphabet}} \rightarrow \underbrace{\Sigma = a \cup b \cup c}_{\text{regular expression}}$$

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



"one or more concatenations of strings
of the form R "



Equivalence: Regular Expressions & NFAs

Regular Expressions Describe Regular Languages

$\overset{n}{\underbrace{(\Rightarrow)}^n}$

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

$\Leftrightarrow$ recognized by a DFA

$\Leftrightarrow$ recognized by an NFA

$\overset{n}{\underbrace{(\Rightarrow)}^n}$

Theorem 1: Every regular expression has an equivalent NFA

$\overset{n}{\underbrace{(\Leftarrow)}^n}$

Theorem 2: Every NFA has an equivalent regular expression

Regular expression $\rightarrow$ NFA

Theorem 1: Every regex has an equivalent NFA

Proof: Induction on size of a regex

Base cases:

	$L(R)$	NFA
$R = \emptyset$	$\emptyset$	$\rightarrow \bigcirc$
$R = \varepsilon$	$\{\varepsilon\}$	$\rightarrow \bigcirc \bigcirc$
$R = a$	$\{a\}$	$\rightarrow \bigcirc \xrightarrow{a} \bigcirc \bigcirc$

Question: Regular expression $\rightarrow$ 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 $\rightarrow$ NFA

Theorem 1: Every regex has an equivalent NFA

Proof: Induction on size of a regex. Base case ✓

I.H. Assume every regex of size $\leq k$ has an equivalent NFA.

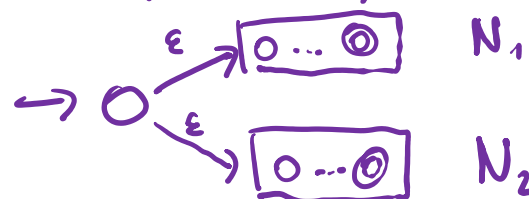
want to show: If R is a regex of size $k+1$, then R has an equiv. NFA.

Inductive step: $k \mapsto k+1$: Size of R is $k+1$.

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

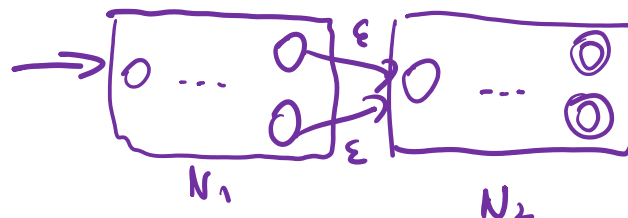
$$R = (R_1 | R_2)$$

$|R_1| \leq k \quad |R_2| \leq k$
I.H. $\Rightarrow \exists N_1$ recognizing R_1
 $\exists N_2$ recognizing R_2

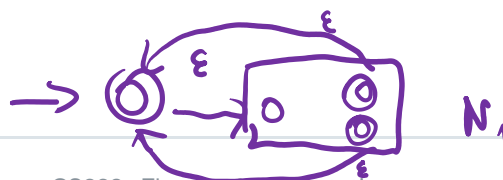


$$R = (R_1^*)$$

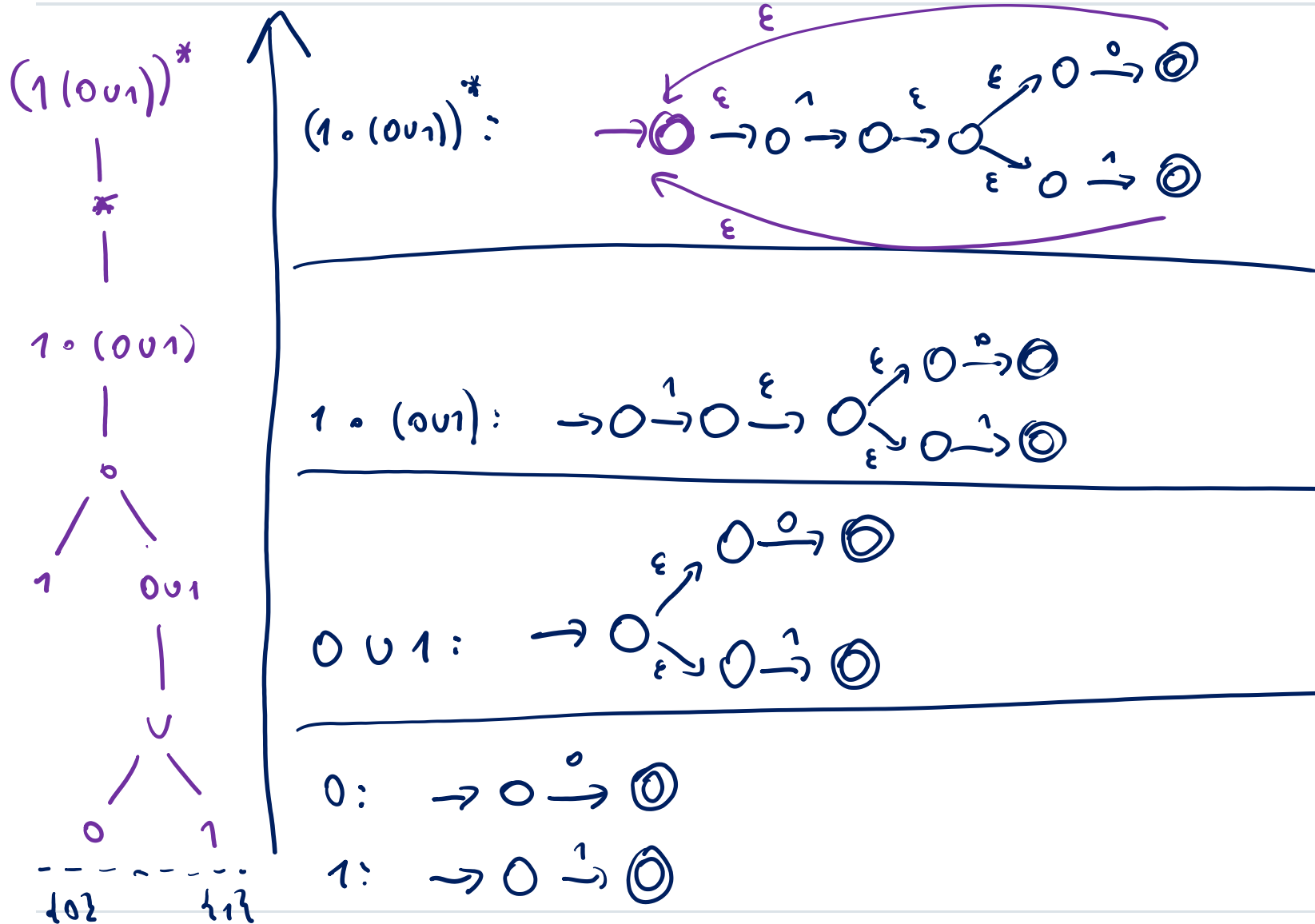
$|R_1| \leq k \quad |R_2| \leq k$



$$|R_1| = k \quad |*| = 1$$



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



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

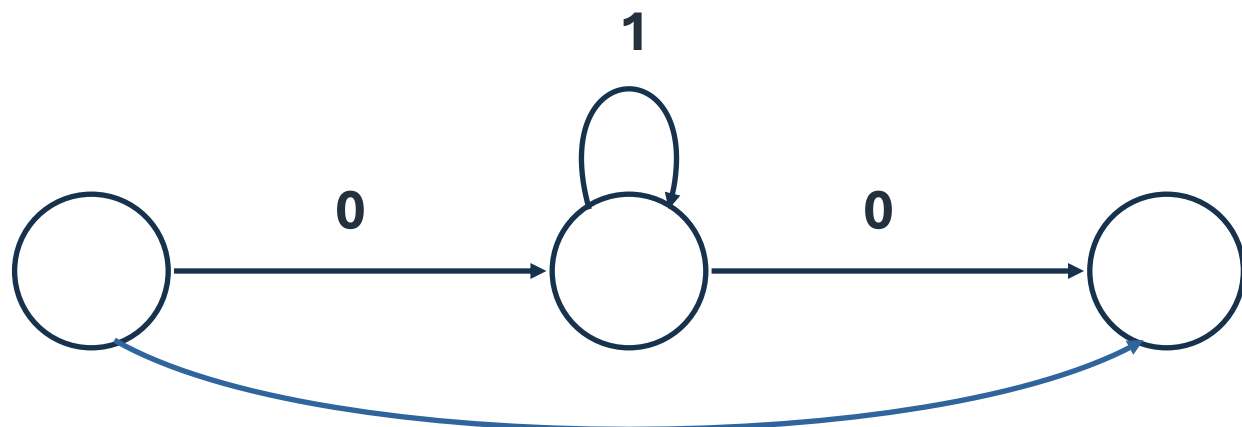
next !

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

01...10
↪ 01^*0

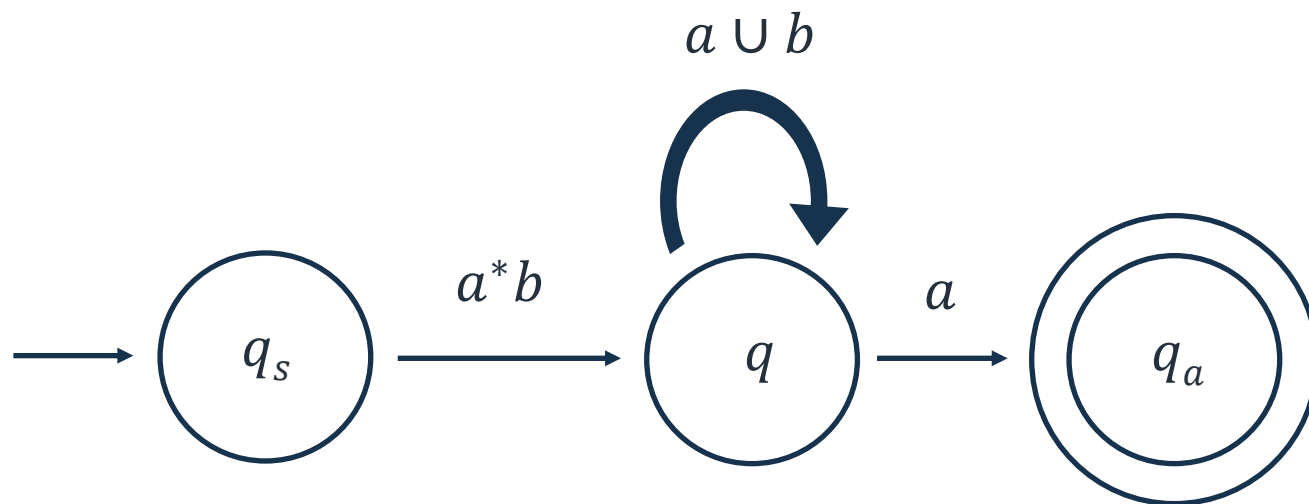


01^*0

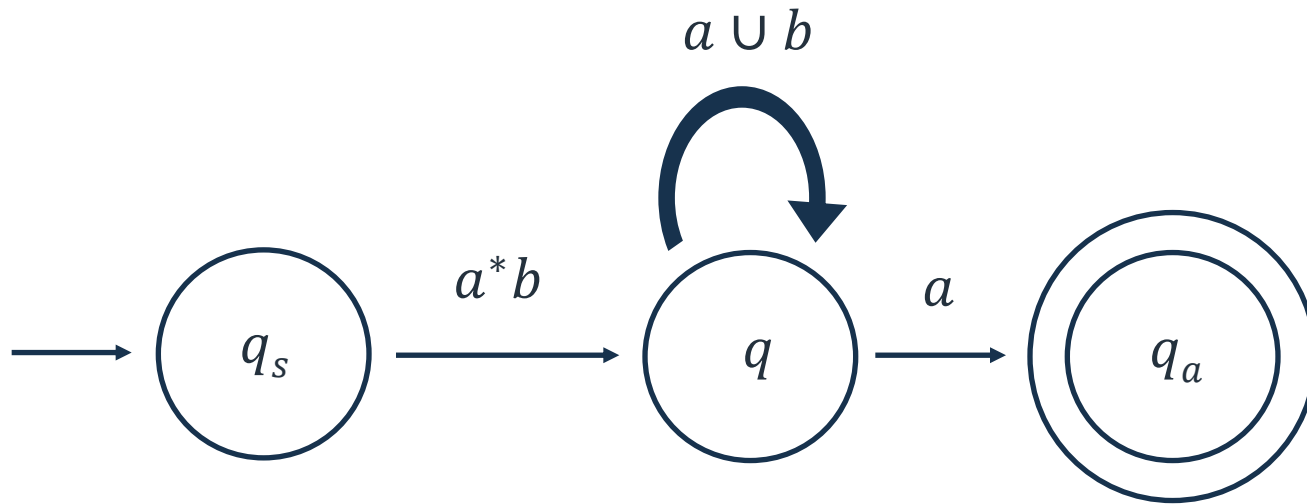
↪ regular expression

Generalized NFAs

- **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



Generalized NFA Example



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

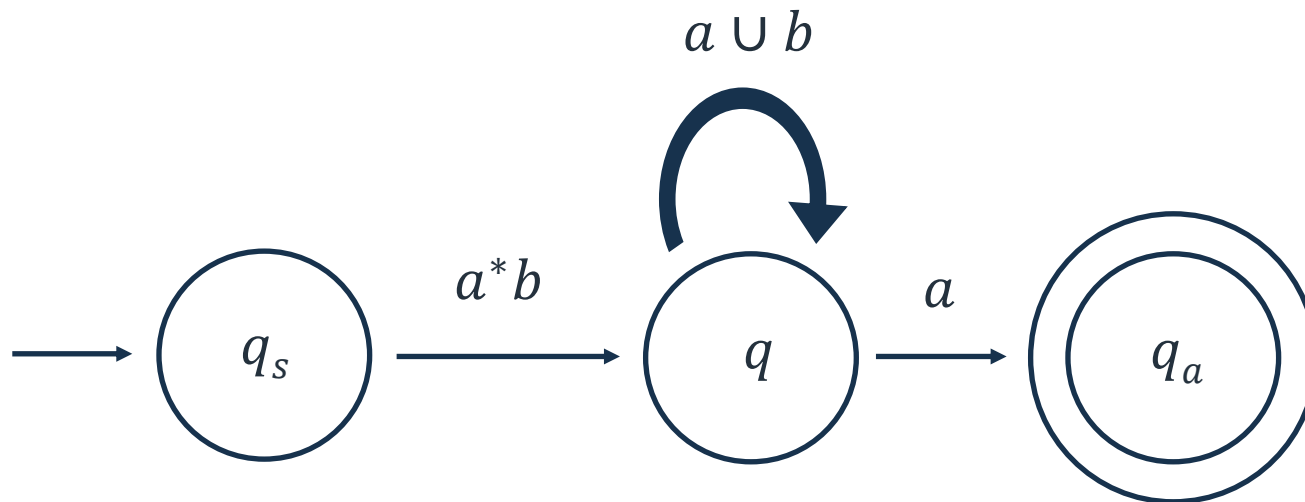
$$R(q_a, q) = \emptyset$$

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

} regular expressions

Question: Which of these strings is accepted?

Which of the following strings is accepted by this GNFA?

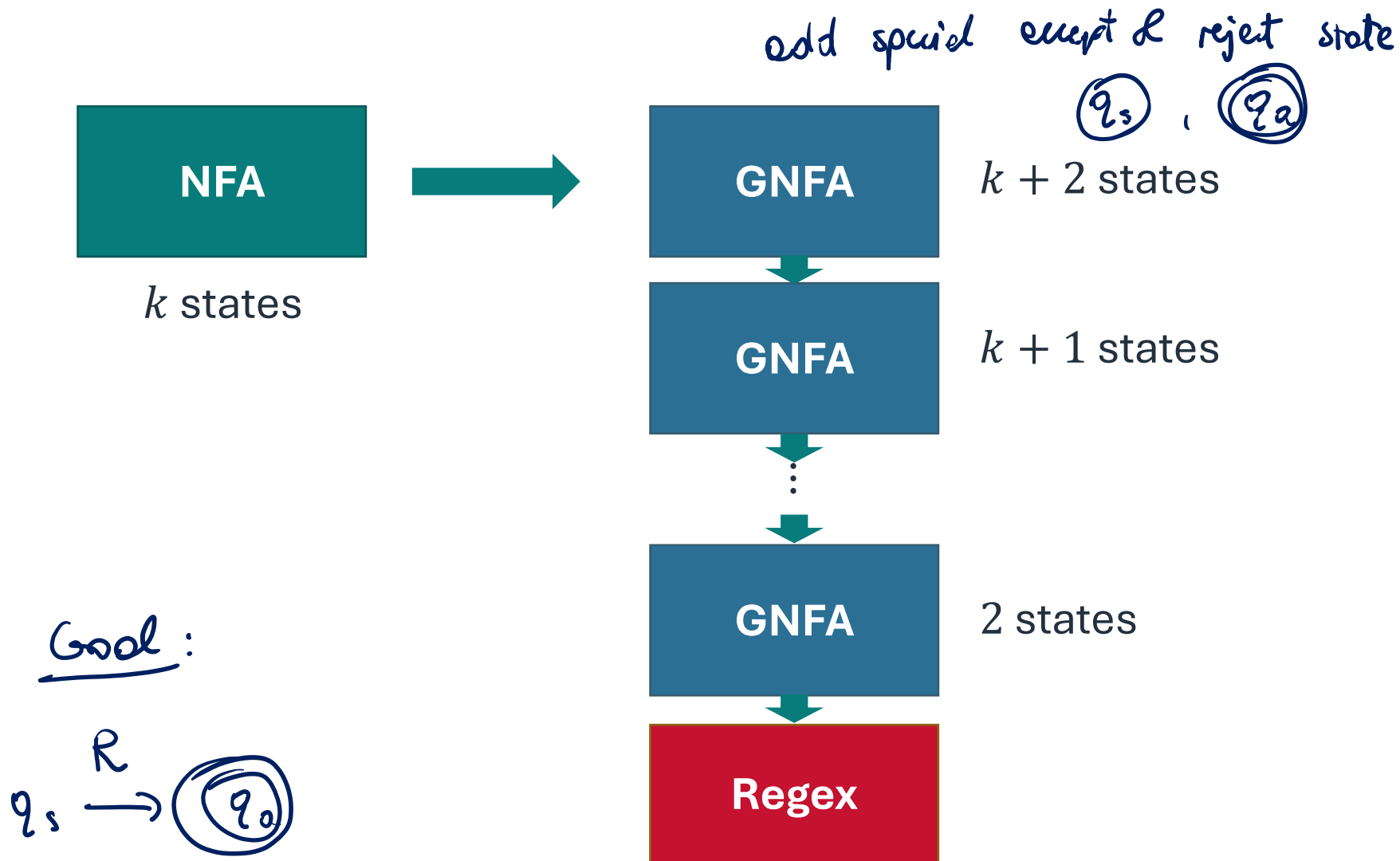


- a) aaa
- b) $aabb$
- c) bbb
- d) bba

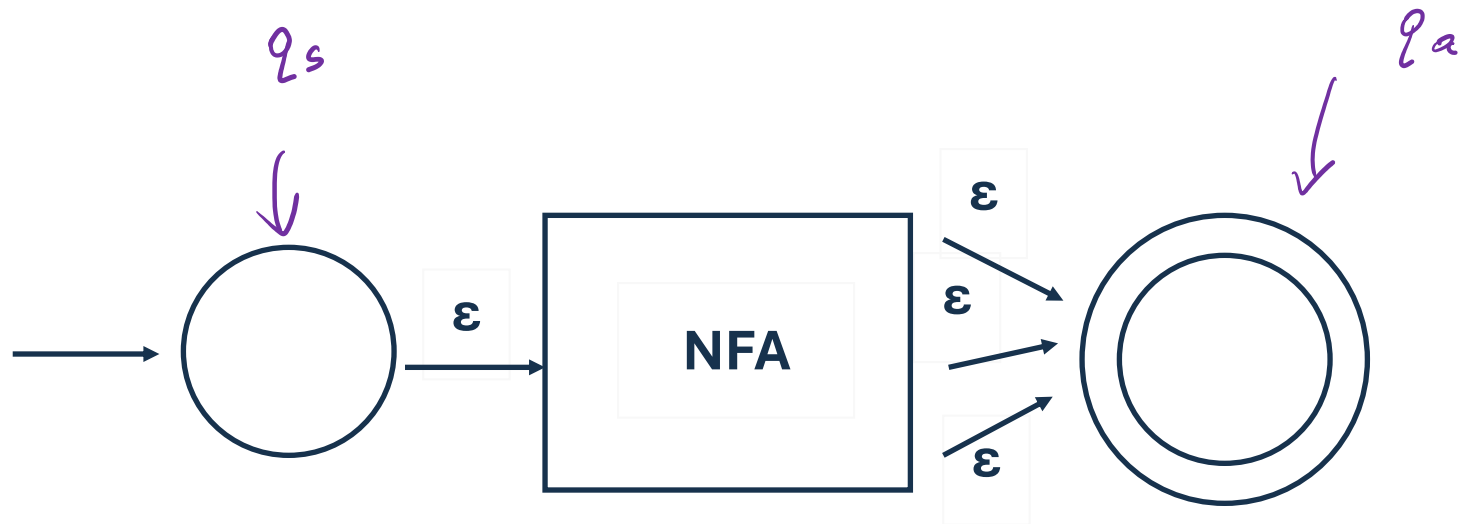
$q_s \xrightarrow{b} q \xrightarrow{b} q \xrightarrow{a} q_a$



NFA $\rightarrow$ Regular expression



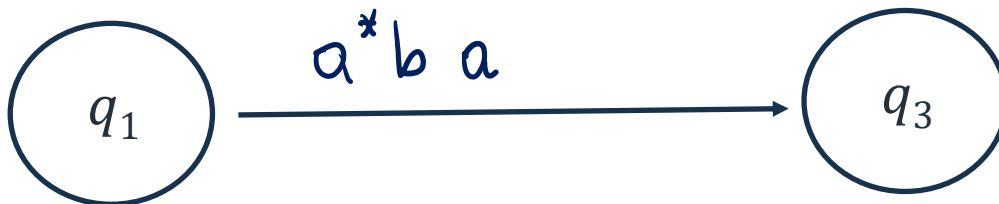
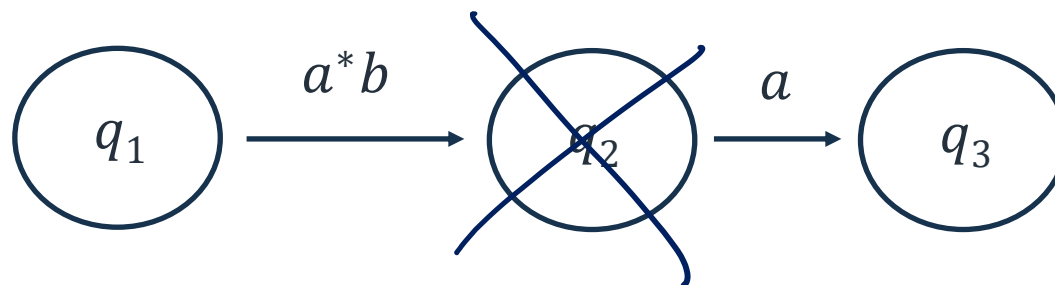
NFA $\rightarrow$ GNFA



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

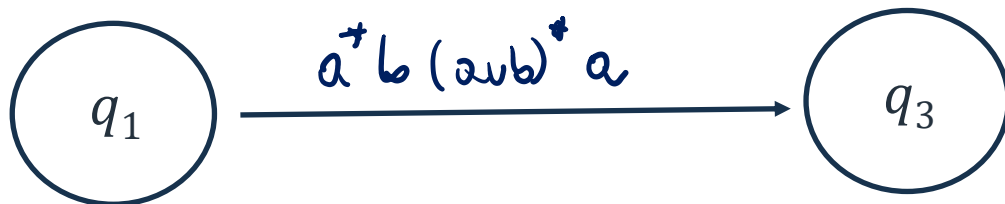
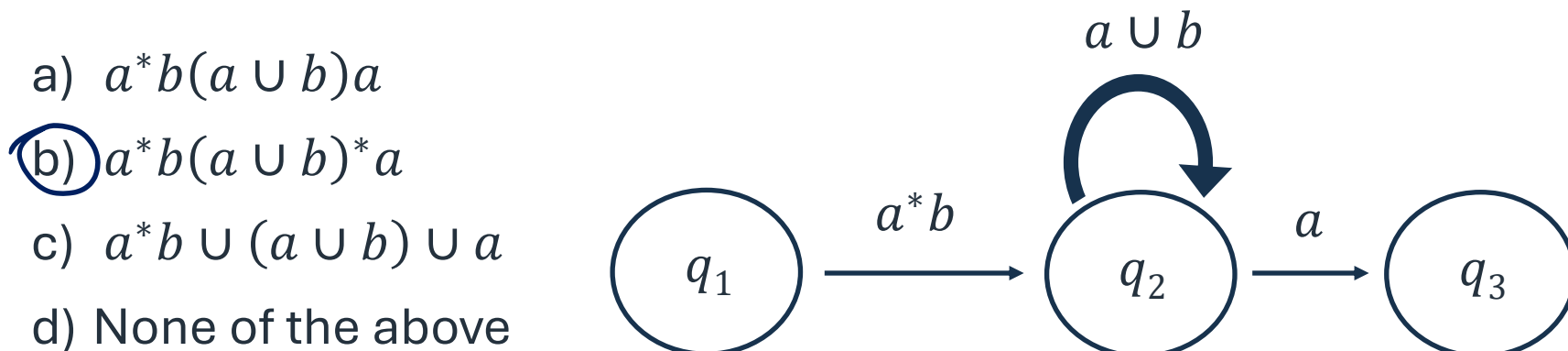
GNFA $\rightarrow$ Regular expression

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



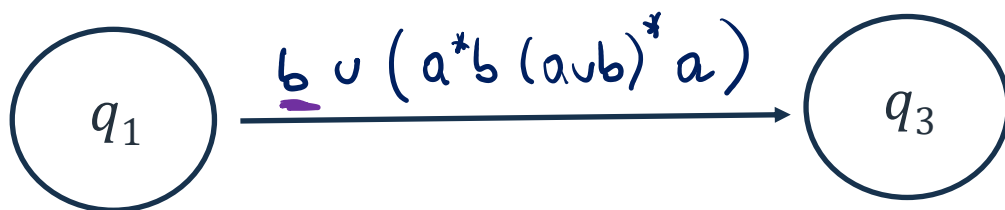
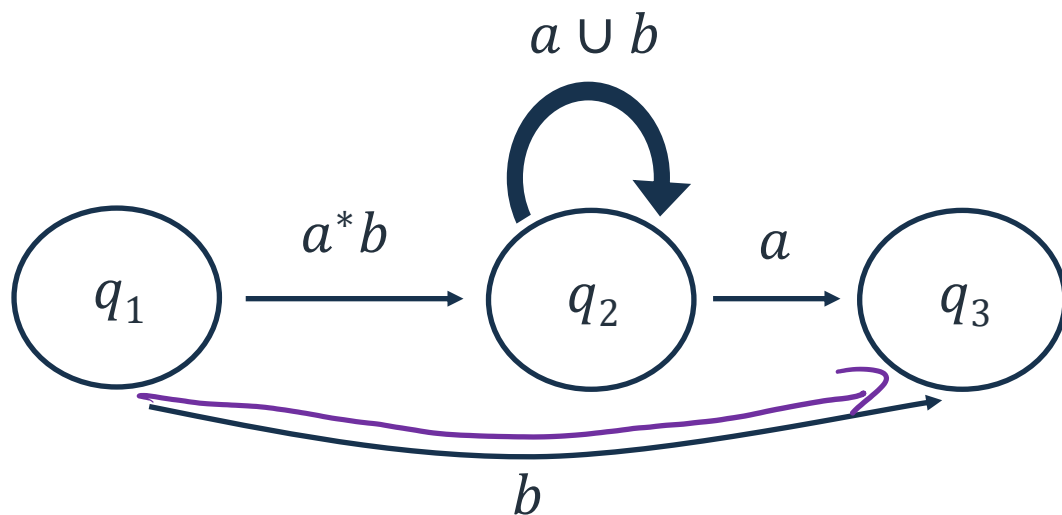
Question: GNFA $\rightarrow$ Regular expression

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



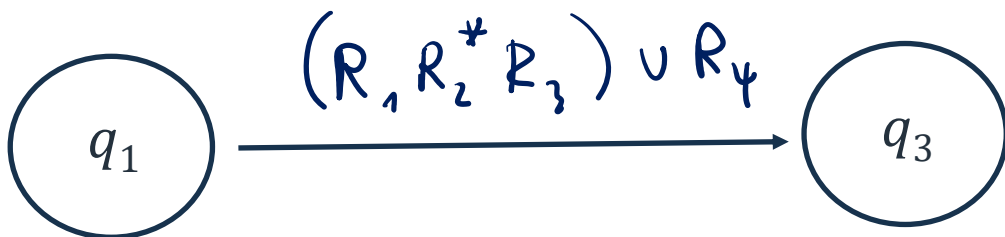
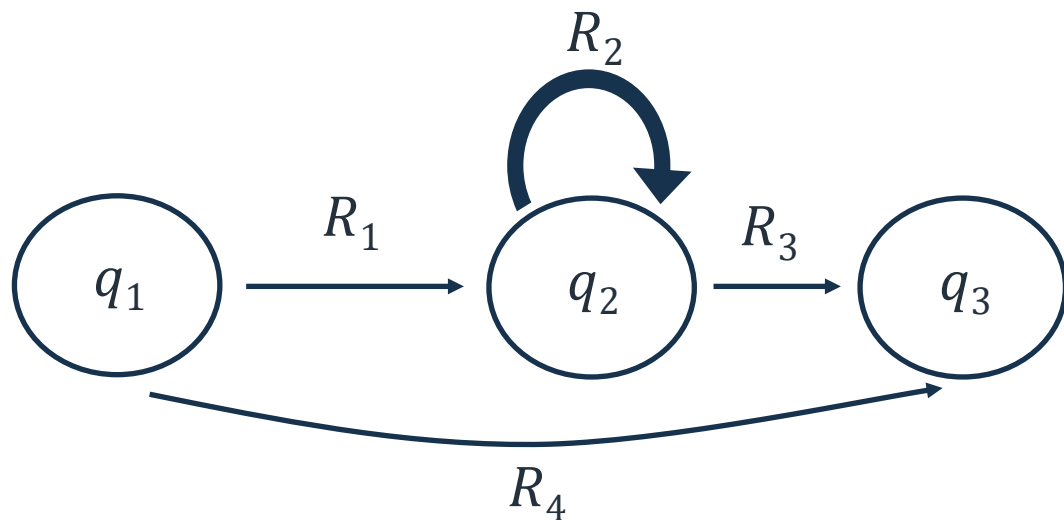
GNFA $\rightarrow$ Regular expression

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

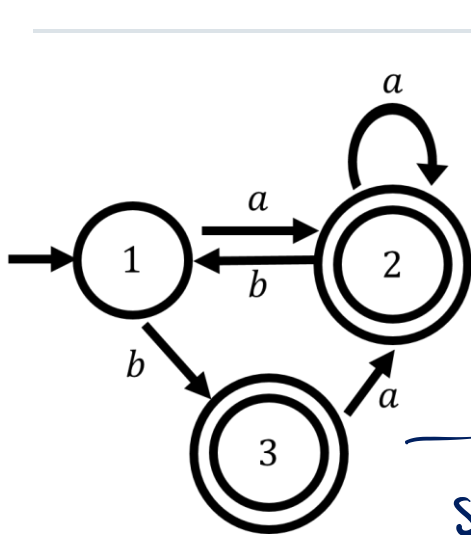


GNFA $\rightarrow$ Regular expression

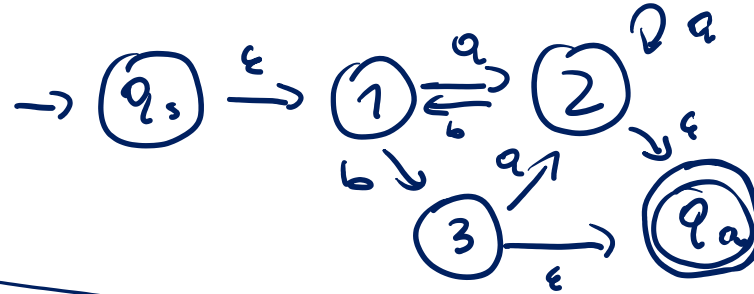
Idea: While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state



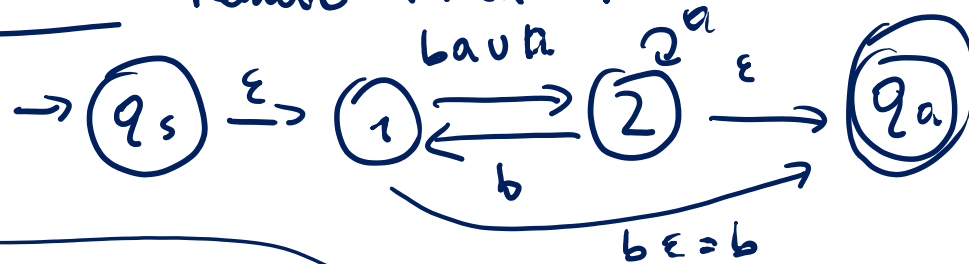
Example



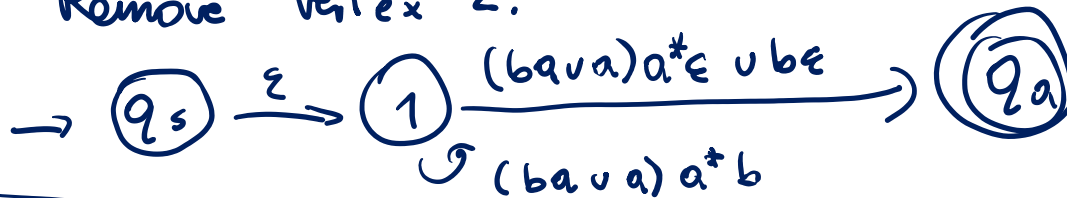
Step 1: Add q_s and q_a



Step 2: Remove vertex 3.

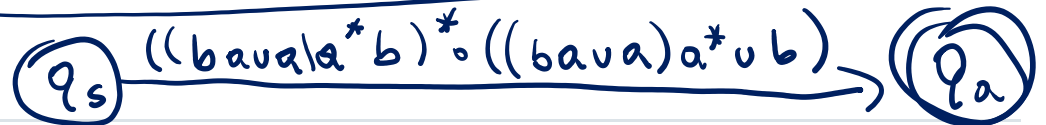


Step 3: Remove vertex 2.



Final
repx

Step 4: Remove vertex 1.



Quiz time!

