BU CS 332 – Theory of Computation

Lecture 6:

- More on pumping
- Regular expressions
- Regular expressions = regular languages

Reading:

Sipser Ch 1.3

Ran Canetti

September 22, 2020

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

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

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

$$\underbrace{0^* 1 0^*}$$

Examples

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

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

2. $\{w \mid w$ contains the string 011 at least twice $\}$

$(0 \cup 1)^* \; 011 \; (0 \cup 1)^* \; 0 \qquad (0 \cup 1)^*$

Examples

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

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

2. $\{w \mid w$ contains the string 011 at least twice $\}$

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

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

Examples

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

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

2. $\{w \mid w$ contains the string 011 at least twice $\}$

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

4. $\{w \mid$every odd position of w is $1\}$

$\underline{(1 \cdot (0 \cup 1))^*}$

Additional notation

$(0 \cup 1)$

Σ

$\{0, 1\}$

- For alphabet Σ, the regex Σ represents $L(\Sigma) = \Sigma$
- For regex R, the regex $R^{+} = RR^{*}$

$R^{*} = \{\varepsilon, R^{+}\}$

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: For any regular expression R, L(R) is regular.

Theorem 2: For any regular language L, there is a regular expression R such that L=L(R).

Regular expression -> NFA

Theorem 1: For any regular expression R, L(R) is regular.

Proof: Induction on size of R.

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

Inductive step:

- $R = (R_1 \cup R_2)$

- $R = (R_1 R_2)$

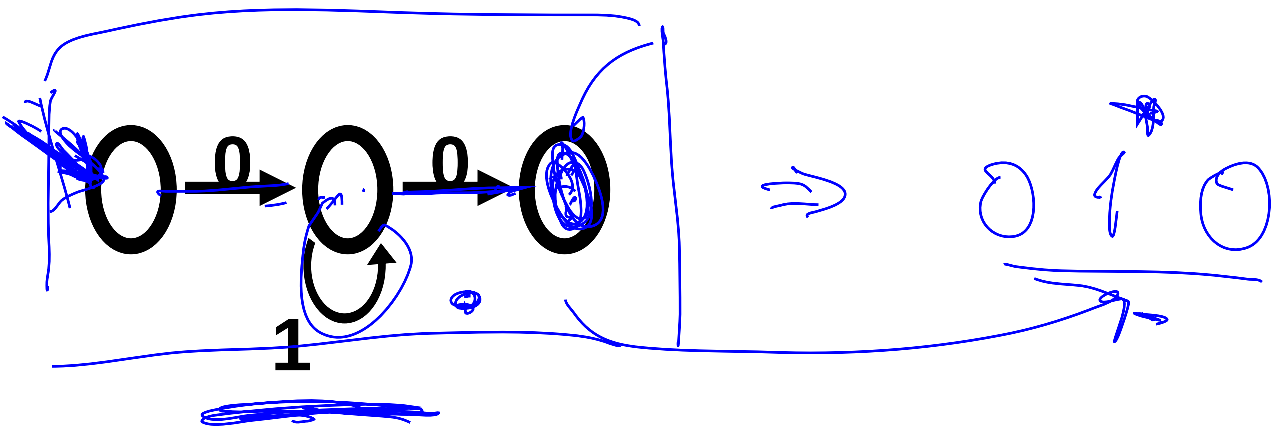
- $R = (R_1^*)$

NFA -> Regular expression

Theorem 2: For any regular language L, there is a regular expression R such that L=L(R).

Proof idea:

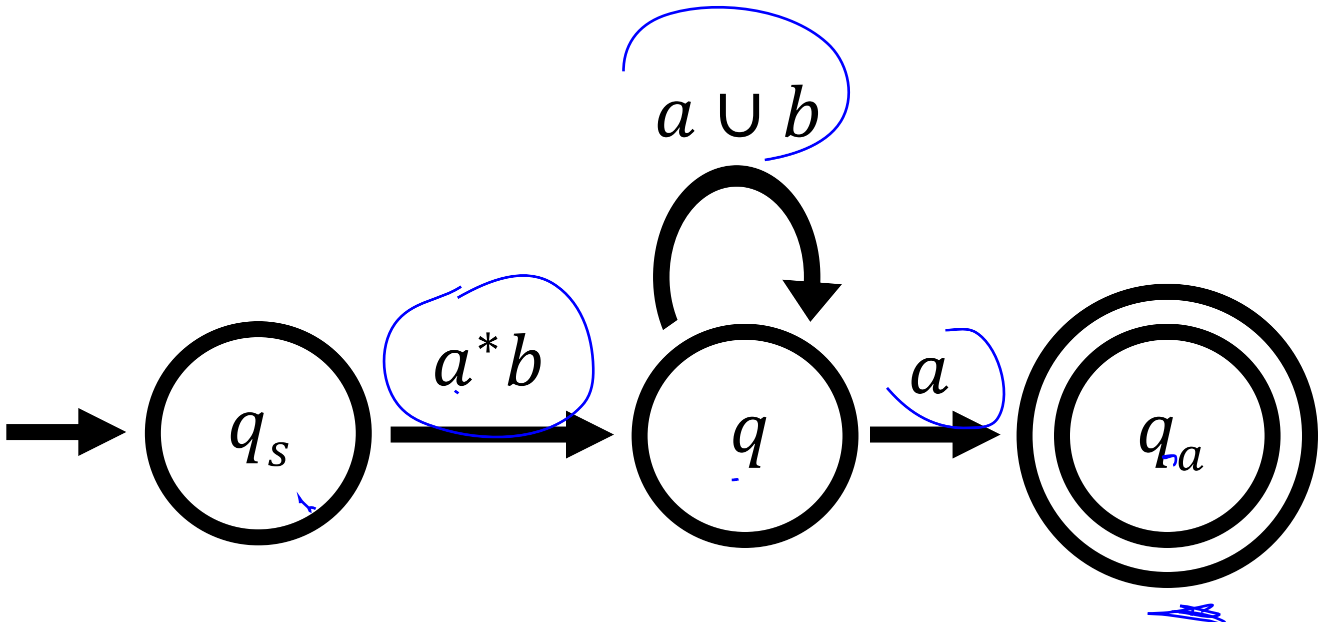
Start from an NFA M for L. Simplify the NFA by "ripping out" states one at a time and replacing them with regexes



Hybrid 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

Hybrid NFA Example

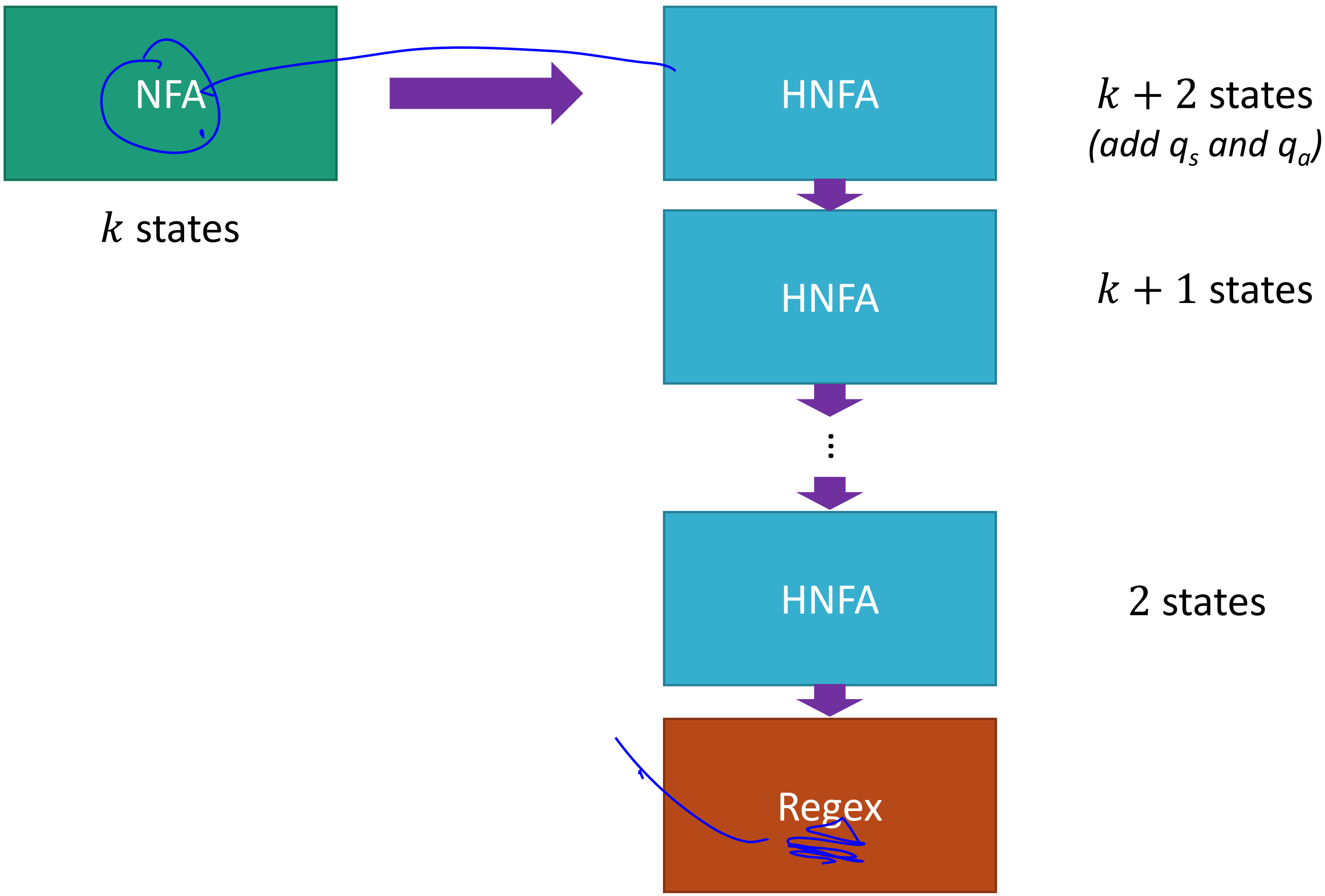


$$R(q_s, q) = a^*b$$

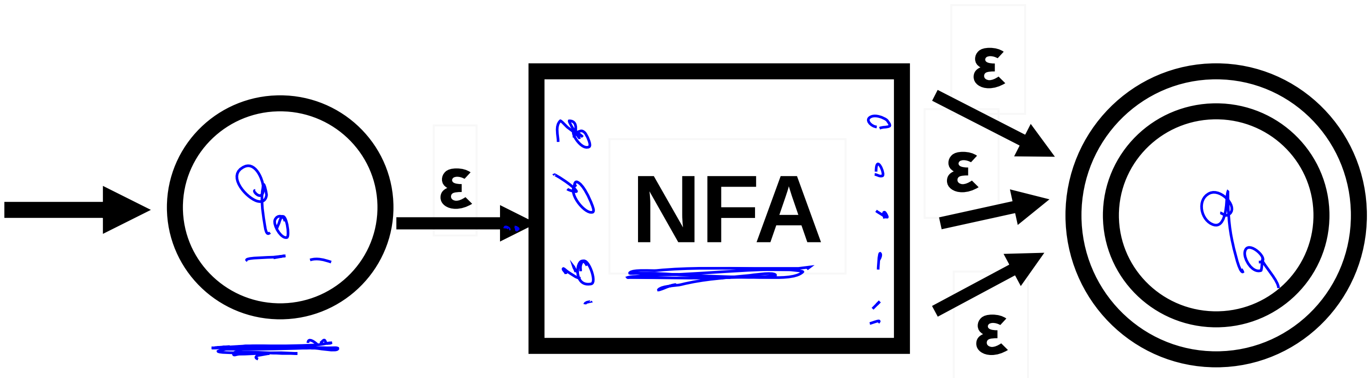
$$R(q, q_a) = a$$

$$R(q, qs) = \phi$$

NFA -> Regular expression



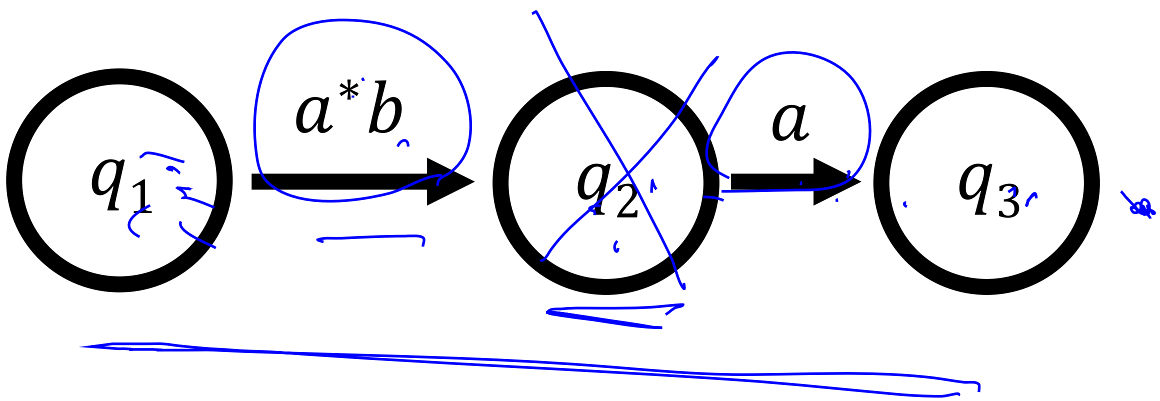
NFA -> HNFA

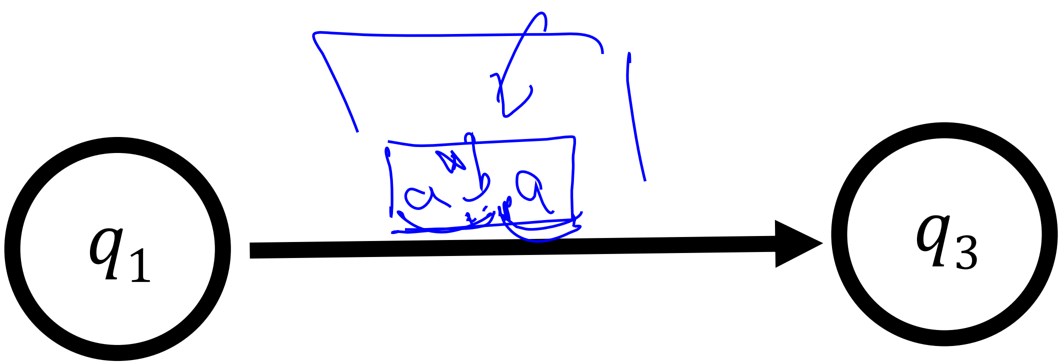


- Add a new start state with no incoming arrows.
- Make a unique accept state with no outgoing arrows.

HNFA -> Regular expression

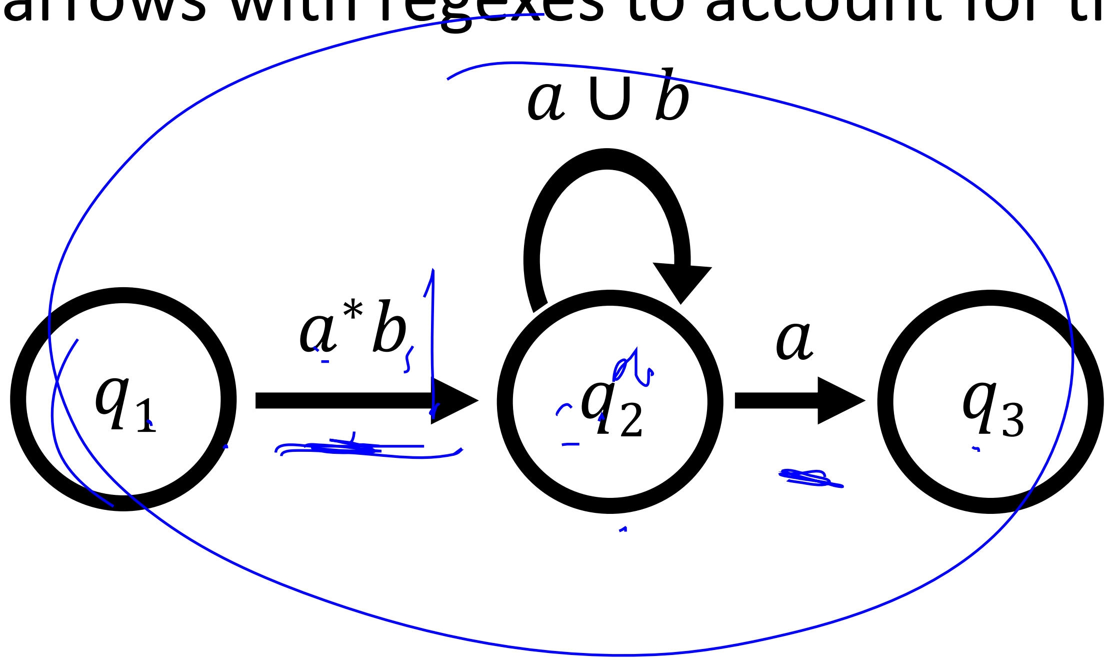
Idea: While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state

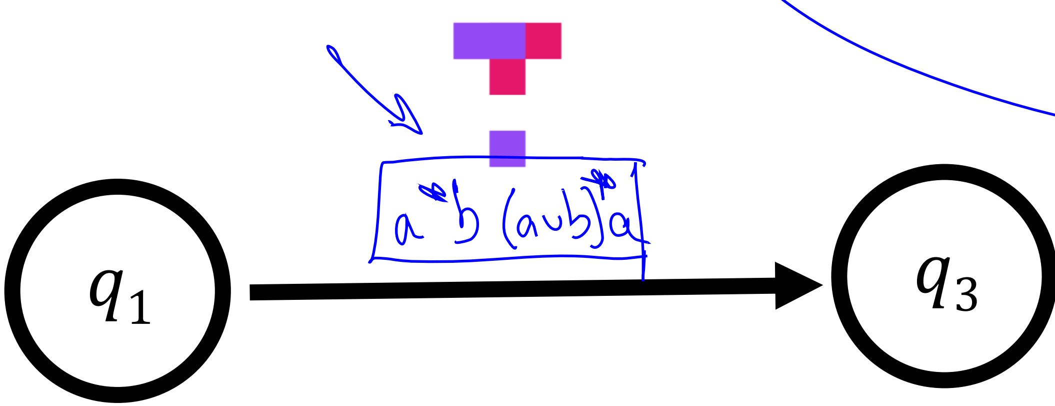




HNFA -> Regular expression

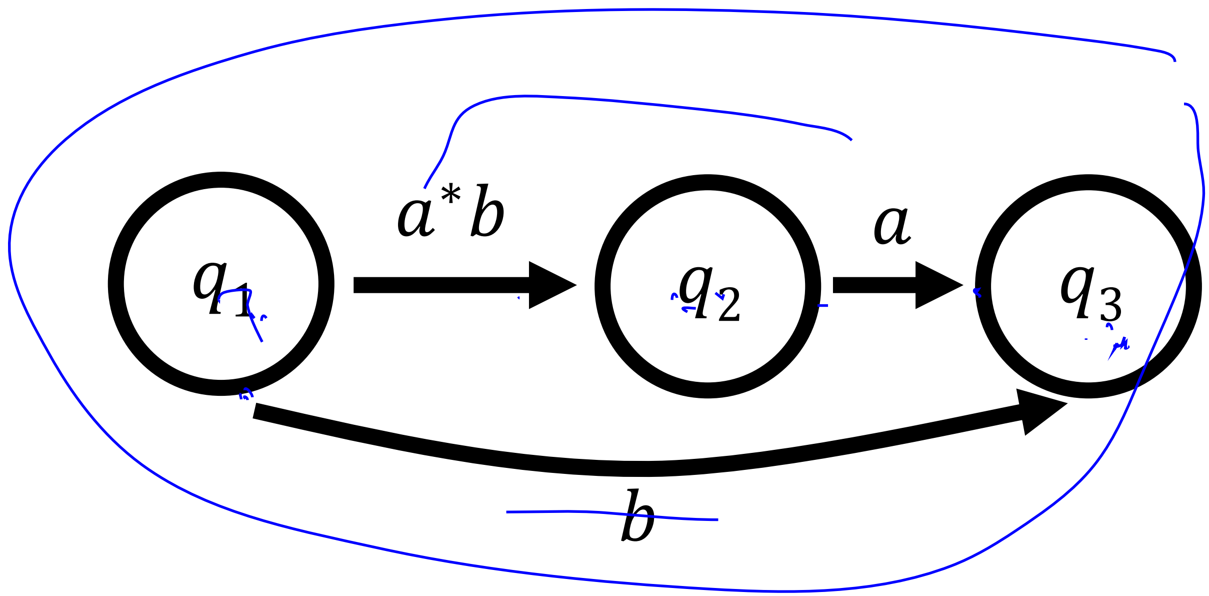
Idea: While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state

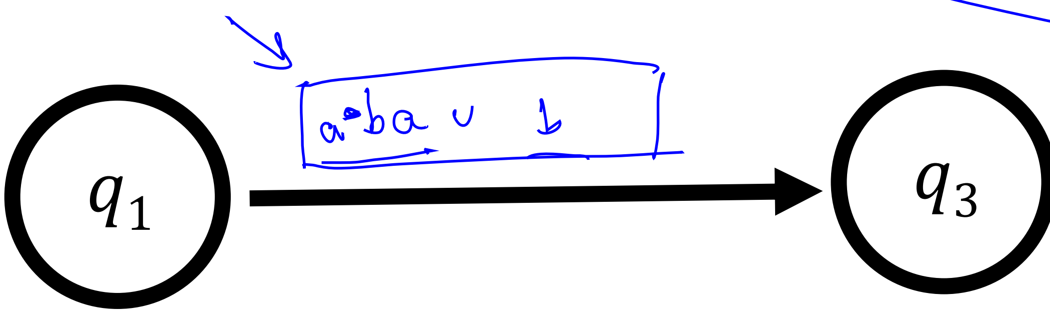




GNFA -> Regular expression

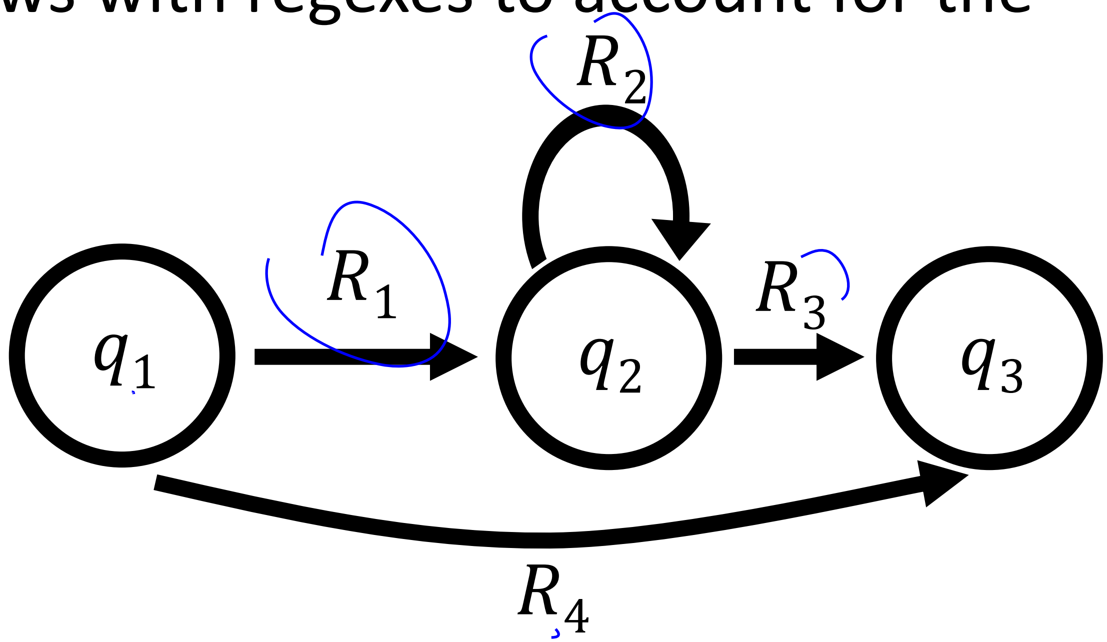
Idea: While the machine has more than 2 states, rip one out and relabel the arrows with regexes to account for the missing state

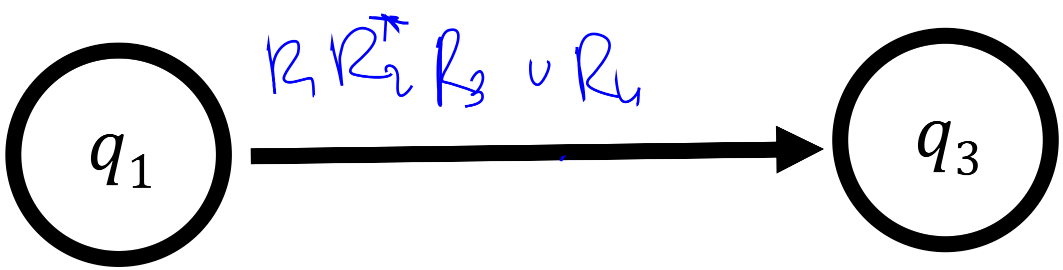




GNFA -> Regular expression

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





Example