Lecture 6: Regular Expressions

CSC 320: Foundations of Computer Science

Quinton Yong

quintonyong@uvic.ca

University of Victoria

Regular Expressions

- To prove **if a language is regular**, is there a way to avoid constructing an entire DFA or NFA? Maybe just by describing the language as an **expression**
- Expressions can also be used **describe regular languages** in a **shorter way**, rather than writing full sentences
  - E.g. $(\mathbf{0} \cup \mathbf{1})^*\mathbf{0}$ for the language of all strings over the binary alphabet that end with $\mathbf{0}$
- **Regular Expression:** a language is regular if and only if we can describe it using a regular expression
- What **syntax / operations** should we have to create expressions that describe regular languages?

Regular Expression: Inductive Definition

$\boldsymbol{R}$ is a **regular expression** if R is equal to:

- $\boldsymbol{a}$ for some $\boldsymbol{a} \in \boldsymbol{\Sigma}$

  e.g. $\mathbf{1}$ where $\boldsymbol{\Sigma} = \{\mathbf{0}, \mathbf{1}\}$, the language with only the string $\mathbf{1}$

- $\boldsymbol{\varepsilon}$
- $\emptyset$

- $(\boldsymbol{R_1} \cup \boldsymbol{R_2})$ where $\boldsymbol{R_1}$ and $\boldsymbol{R_2}$ are regular expressions

  e.g. $(\mathbf{0} \cup \mathbf{1})$, the language with strings $\mathbf{0}$ or $\mathbf{1}$

- $(\boldsymbol{R_1}\boldsymbol{R_2})$ where $\boldsymbol{R_1}$ and $\boldsymbol{R_2}$ are regular expressions

  e.g. $(\mathbf{11})$, the language with string $\mathbf{11}$

  e.g. $(\mathbf{11})(\mathbf{0} \cup \mathbf{1})$, the language with strings $\mathbf{110}$ and $\mathbf{111}$

- $(\boldsymbol{R_1}^*)$ where $\boldsymbol{R_1}$ is a regular expression

  e.g. $(\mathbf{1}^*)$, the language with strings of **zero or more** concatenations of $\mathbf{1}$s

  e.g. $(\mathbf{0} \cup \mathbf{1})^*$, the language $\boldsymbol{\Sigma}^*$ where $\boldsymbol{\Sigma} = \{\mathbf{0}, \mathbf{1}\}$

Regular Expression: Conventions and Identities

- Parentheses can be omitted

  e.g. $(\mathbf{1}^*)$ can just be written as $\mathbf{1}^*$

  e.g. $(\mathbf{11})(\mathbf{00})$ can just be written as $\mathbf{1100}$

- If no parentheses, order to evaluate is: **star**, **concatenation**, **union**

  e.g. $\mathbf{1}^*\mathbf{10} \cup \mathbf{1}$ is evaluated as $\left((\mathbf{1}^*)\mathbf{10}\right) \cup \mathbf{1}$

- $\boldsymbol{R}^+ \coloneqq \boldsymbol{R}\boldsymbol{R}^*$

  Basically R^* but without the “zero concatenation” case

  e.g. 1^+ , the language with strings of **one or more** concatenations of $\mathbf{1}$

- $\boldsymbol{R} \cup \emptyset = \boldsymbol{R}$

- $\boldsymbol{R\varepsilon} = \boldsymbol{\varepsilon R} = \boldsymbol{R}$

- $\boldsymbol{R}\emptyset = \emptyset$

Language Recognized by a Regular Expression

Let $\boldsymbol{R_1}$ and $\boldsymbol{R_2}$ be regular expressions.

The language $\boldsymbol{L(R)}$ for a **regular expression** $\boldsymbol{R}$ is defined as:

- If $\boldsymbol{R = a}$, for some $\boldsymbol{a} \in \boldsymbol{\Sigma}$, then $\boldsymbol{L(R) = \{a\}}$
- If $\boldsymbol{R = \varepsilon}$, then $\boldsymbol{L(R) = \{\varepsilon\}}$
- If $\boldsymbol{R} = \emptyset$, then $\boldsymbol{L(R)} = \emptyset$

- If $\boldsymbol{R = (R_1 \cup R_2)}$, then $\boldsymbol{L(R) = L(R_1) \cup L(R_2)}$
- If $\boldsymbol{R = (R_1 R_2)}$, then $\boldsymbol{L(R) = L(R_1) L(R_2)}$
- If $\boldsymbol{R = (R_1{}^*)}$, then $\boldsymbol{L(R) = L(R_1)^*}$

Regular Expression Examples

Describe the languages of the following regular expressions:

Let $\boldsymbol{\Sigma} = \{\boldsymbol{a}, \boldsymbol{b}\}$:

- $\boldsymbol{L(a \cup b) = L(a) \cup L(b) = \{a\} \cup \{b\} = \{a, b\}}$
- $\boldsymbol{L\big(a(a \cup b)\big) = L(a)L(a \cup b) = \{a\}\{a, b\} = \{aa, ab\}}$
- $\boldsymbol{L\big((a \cup b)^*\big) = L(a \cup b)^* = \{a, b\}^* = \Sigma^*}$
- $\boldsymbol{L(a(a \cup b)^*) = L(a)L\big((a \cup b)^*\big) = \{a\}\{a, b\}^* = \{a, aa, ab, \dots\}}$
  - i.e. $\boldsymbol{a}$ followed by anything

Language of regular expressions:

- $\boldsymbol{L(a) = \{a\}}$
- $\boldsymbol{L(\varepsilon) = \{\varepsilon\}}$
- $\boldsymbol{L(\emptyset) = \emptyset}$
- $\boldsymbol{L(R_1 \cup R_2) = L(R_1) \cup L(R_2)}$
- $\boldsymbol{L(R_1R_2) = L(R_1)L(R_2)}$
- $\boldsymbol{L(R^*) = L(R)^*}$

Equivalence of Regular Expressions

We will show that the set of languages **describable by regular expression** is the same as the **regular languages** (languages representable by DFAs)

Theorem: A language is regular if and only if there exists some regular expression that describes it

Proof consists of two parts:

1. If a language is described by a **regular expression**, then it is **regular** (can be recognized by a DFA / NFA)
2. If a language is **regular** (can be recognized by a DFA / NFA), then it can be described by a **regular expression**

Regular Expression to NFA

1. Given a **regular expression** R we show that there exists a **finite automaton** M with $L(M) = L(R)$

We distinguish the following cases:

1. $R = a\ ,\ \ a \in \Sigma$
2. $R = \varepsilon$
3. $R = \emptyset$

Base cases

4. $R = (R_1 \cup R_2)$, where R_1, R_2 are regular expressions
5. $R = (R_1 R_2)$, where R_1, R_2 are regular expressions
6. $R = (R_1{}^*)$, where R_1 is a regular expression

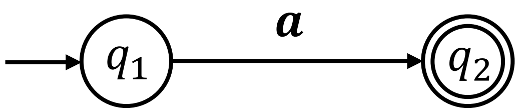
- We show that we can **build NFAs** to recognize case **1** to **3**
- Then, we show that we can **combine those NFAs** to recognize cases **4** to **6**

Regular Expression to NFA

Case 1: $\boldsymbol{R = a}$, $\boldsymbol{a \in \Sigma}$

Show that $\boldsymbol{L(R)}$ is regular:

- $\boldsymbol{L(a) = \{a\}}$



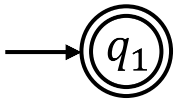
- **NFA** $\boldsymbol{N} = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$ with
  - $\boldsymbol{\delta(q_1, a) = \{q_2\}}$
  - $\boldsymbol{\delta(r, b)} = \emptyset$ for $\boldsymbol{(r, b) \neq (q_1, a)}$ No other transitions

Regular Expression to NFA

Case 2: $\boldsymbol{R = \varepsilon}$

Show that $\boldsymbol{L(R)}$ is regular:

- $\boldsymbol{L(\varepsilon)} = \{\boldsymbol{\varepsilon}\}$



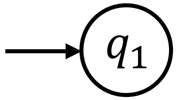
- **NFA** $\boldsymbol{N} = (\{q_1\}, \Sigma, \delta, q_1, \{q_1\})$ with
  - $\boldsymbol{\delta(q_1, b)} = \emptyset$ for any $\boldsymbol{b} \in \boldsymbol{\Sigma} \cup \{\boldsymbol{\varepsilon}\}$

Regular Expression to NFA

Case 3: $R = \emptyset$

Show that $L(R)$ is regular:

- $L(\emptyset) = \emptyset$



- **NFA** $N = (\{q_1\}, \Sigma, \delta, q_1, \emptyset)$ with
  - $\delta(q_1, b) = \emptyset$ for any $b \in \Sigma \cup \{\varepsilon\}$

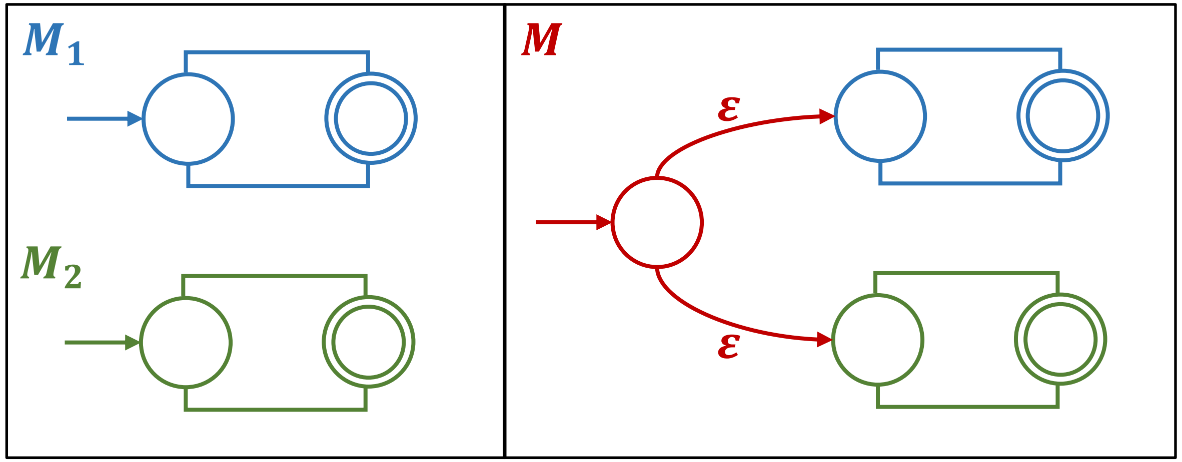
Regular Expression to NFA

Case 4: $R = R_1 \cup R_2$

Show that $L(R)$ is regular:

(**Induction Step**) Assume that $L(R_1)$ and $L(R_2)$ are regular languages

- By definition, $L(R) = L(R_1 \cup R_2) = L(R_1) \cup L(R_2)$
- Since regular languages are closed under union, $L(R)$ is regular



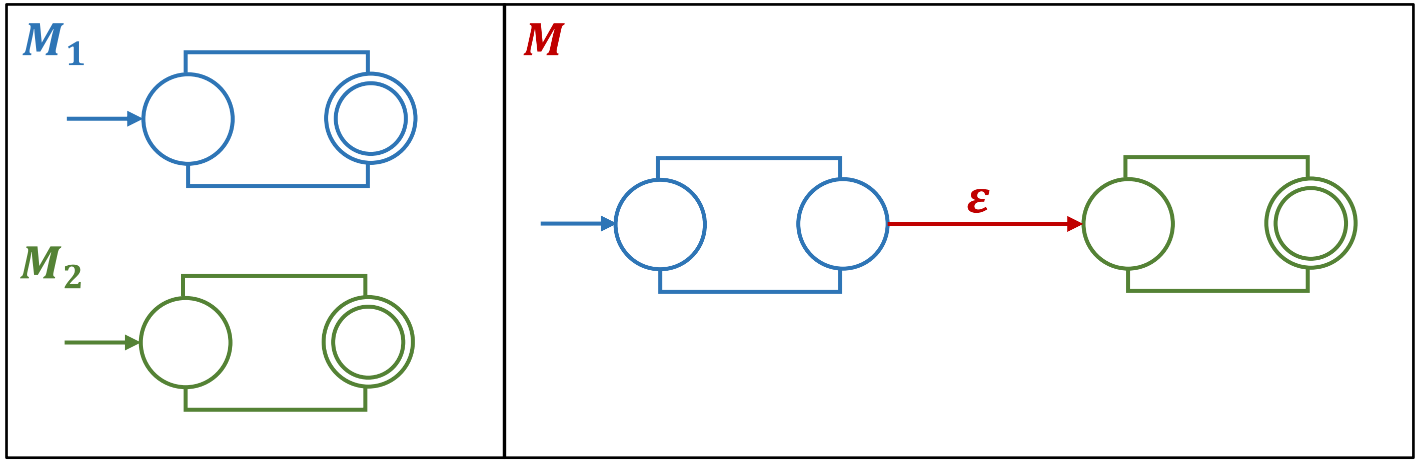
Regular Expression to NFA

Case 5: $R = R_1 R_2$

Show that $L(R)$ is regular:

(**Induction Step**) Assume that $L(R_1)$ and $L(R_2)$ are regular languages

- By definition, $L(R) = L(R_1 R_2) = L(R_1)L(R_2)$
- Since regular languages are closed under concatenation, $L(R)$ is regular



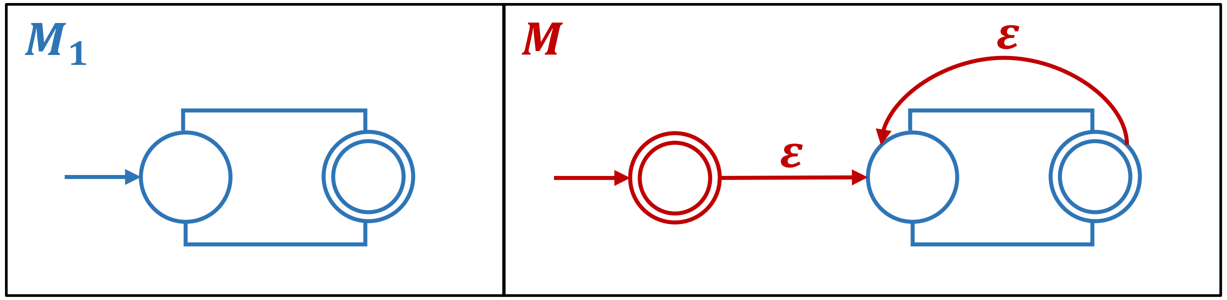
Regular Expression to NFA

<u>**Case 6**: $R = R_1^*$</u>

Show that $L(R)$ is regular:

(**Induction Step**) Assume that $L(R_1)$ is a regular languages

- By definition, $L(R) = L(R_1^*) = L(R_1)^*$
- Since regular languages are closed under Kleene star, $L(R)$ is regular



Equivalence of Regular Expressions

Proof consists of two parts:

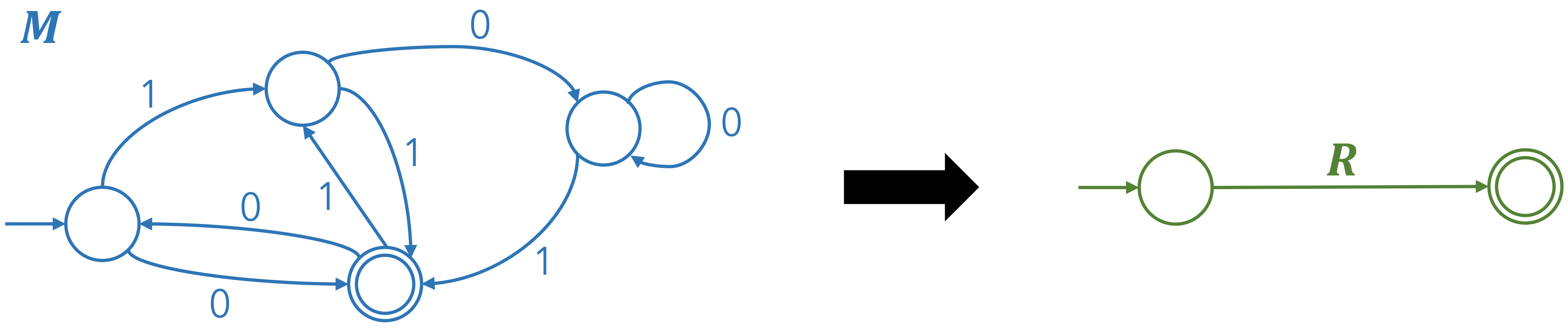
1. If a language is described by a **regular expression**, then it is **regular** (can be recognized by a DFA / NFA)
2. If a language is **regular** (can be recognized by a DFA / NFA), then it can be described by a **regular expression**

DFA to Regular Expression

2. If a language is **regular** (can be recognized by a DFA / NFA), then it can be described by a **regular expression**

We **convert a DFA** $\boldsymbol{M}$ which recognizes the language into a **regular expression** $\boldsymbol{R}$

- **Transform** M into a **generalized NFA** (**GNFA**): a hybrid between an automaton and a regular expression
- **Shrink the GNFA** until we obtain the regular expression $\boldsymbol{R}$ which recognizes the same language as $\boldsymbol{M}$



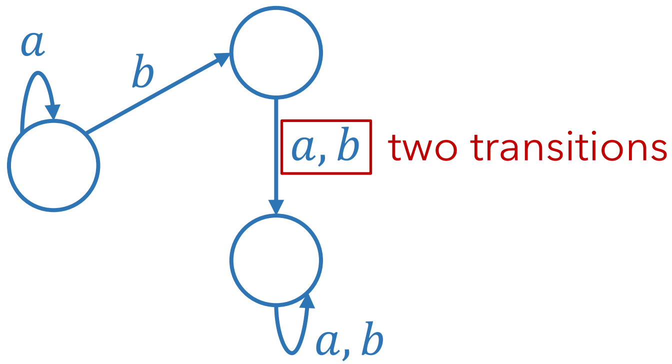
Generalized NFA (GNFA)

A **GNFA** $\boldsymbol{G}$ is almost like an NFA except:

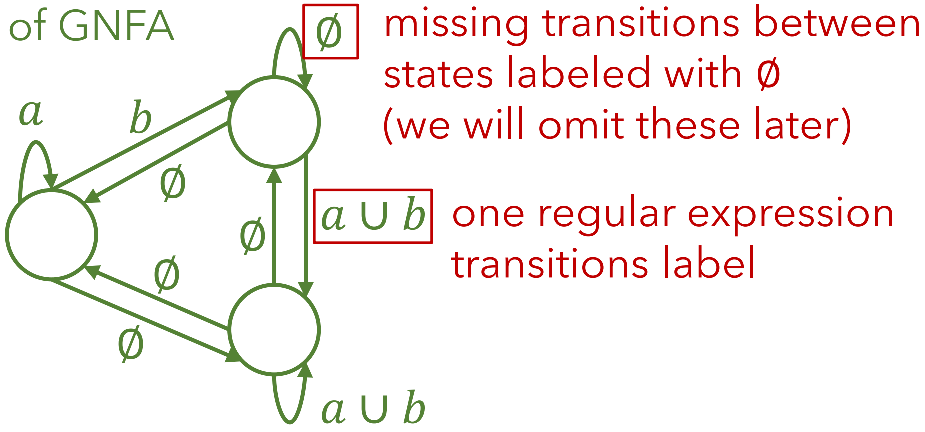
- $\boldsymbol{G}$ has exactly **one start state**
- $\boldsymbol{G}$ has exactly **one accept state**

- Transitions are labeled with **regular expressions**
- Exactly **one transition** from **every state** to **every other state** (and **self loops**)

part of DFA



part of GNFA



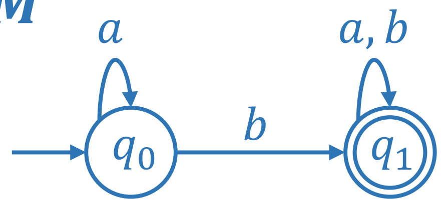
- **Exceptions**: No transitions **to start state**, no transitions **from accept state**

DFA to GNFA

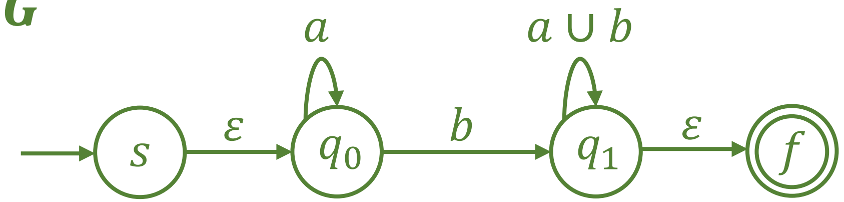
Given **DFA** $M = (Q, \Sigma, \delta, q_0, F)$, create **GNFA** G as follows:

- Add **new start state** s and **ε-transition** from s to q_0
- G needs only one accept state:
  - Add **new accept state** f and **ε-transitions** from all states in F to f
  - Change $M's$ accept states into non-accept states in G
- Transform label on each transition into **regular expression**
  - Single symbols stay the same
  - **Combine** multiple transitions into **one transition** (e.g. a, b turns into $a \cup b$)
- Between pairs of states with no transition, add transition with label $\emptyset$

M



G



(∅-transitions are omitted)

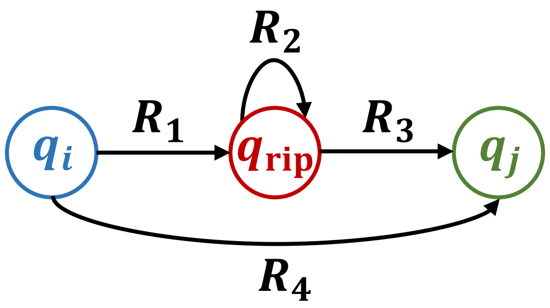
GNFA to Regular Expression

- We gradually shrink $\boldsymbol{G}$ by **removing states**
- Replace removed states with a more **complex regular expression** representing "strings created" by going through that state
- Finally, the only states left are $\boldsymbol{s}$ and $\boldsymbol{f}$ with a **single regular expression $\boldsymbol{R}$** transition

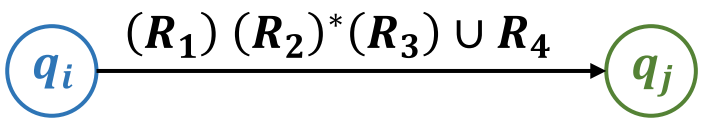
Removing a State from GNFA

Remove states from G (not s or f) one by one, while not changing the **language**

- Consider state to be removed: q_{rip}
- When removed, transitions through q_{rip} must be replaced

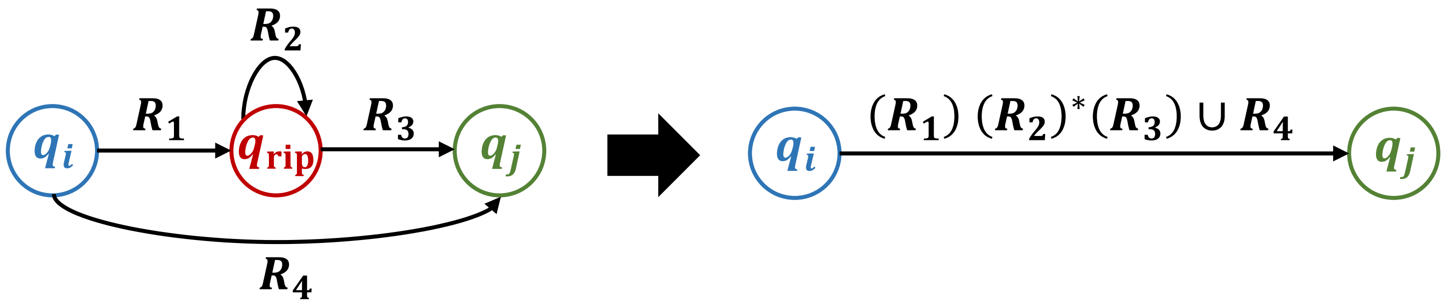


- The strings obtained from some q_i going through q_{rip} to some q_j look like: R_1, concatenated with zero or more repeats of R_2, concatenated with R_3
- So, transitions going through q_{rip} must be replaced by:



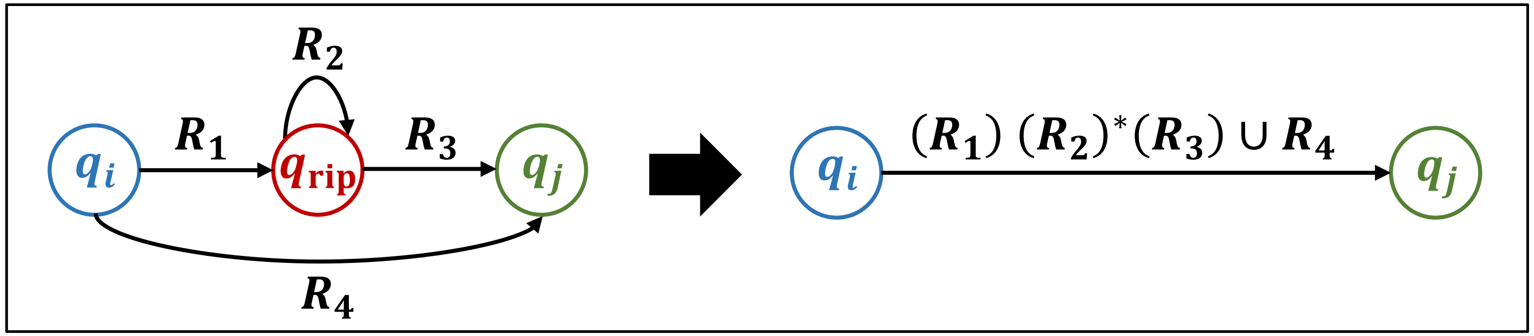
GNFA to Regular Expression

- Let $G = (Q \cup \{s, f\}, \Sigma, \delta, s, f)$ be a **GNFA** (converted from **DFA** $M = (Q, \Sigma, \delta, q_0, F)$)
- Choose a state q_{rip} and turn G to $G' = (Q', \Sigma, \delta', s, f)$ with $Q' = Q \cup \{s, f\} \setminus \{q_{rip}\}$ and update δ to δ' as follows:
  - Let $\mathbf{reg}(q_i, q_j)$ denote the regular expression transition between q_i and q_j
  - For each $q_i, q_j \in Q'$, set $\mathbf{reg}(q_i, q_j) \coloneqq (R_1)(R_2)^*(R_3) \cup (R_4)$ where
    - $R_1 = \mathbf{reg}(q_i, q_{\text{rip}})$
    - $R_2 = \mathbf{reg}(q_{\text{rip}}, q_{\text{rip}})$
    - $R_3 = \mathbf{reg}(q_{\text{rip}}, q_j)$
    - $R_4 = \mathbf{reg}(q_i, q_j)$

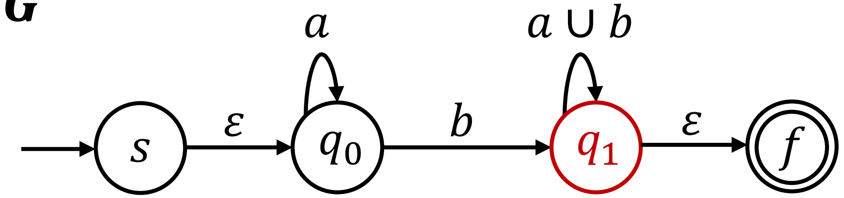


- **Note:** even though conversion says **for each** $q_i, q_j \in Q'$, we really only have to do this for each q_i, q_j where q_i can get to q_j through q_{rip}

GNFA to Regular Expression Example

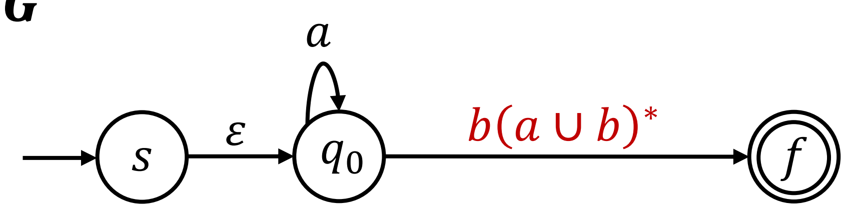


G

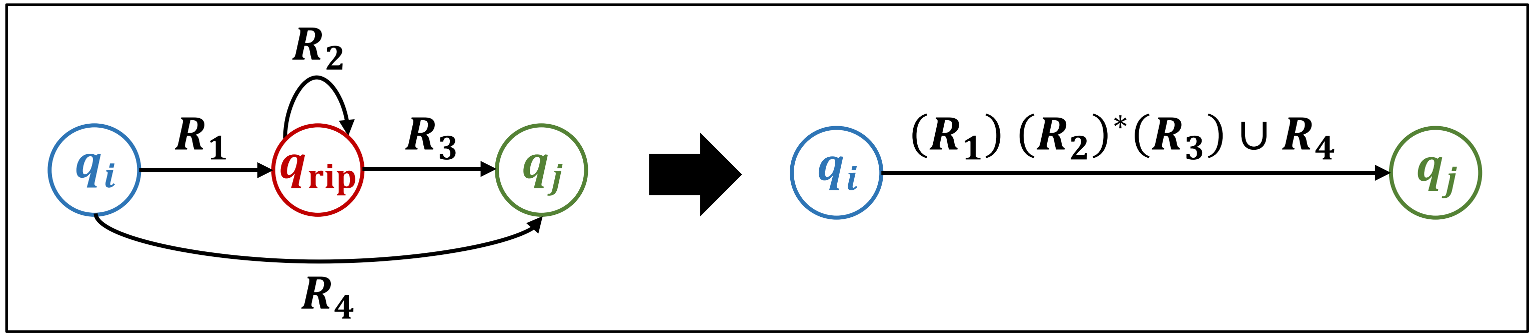


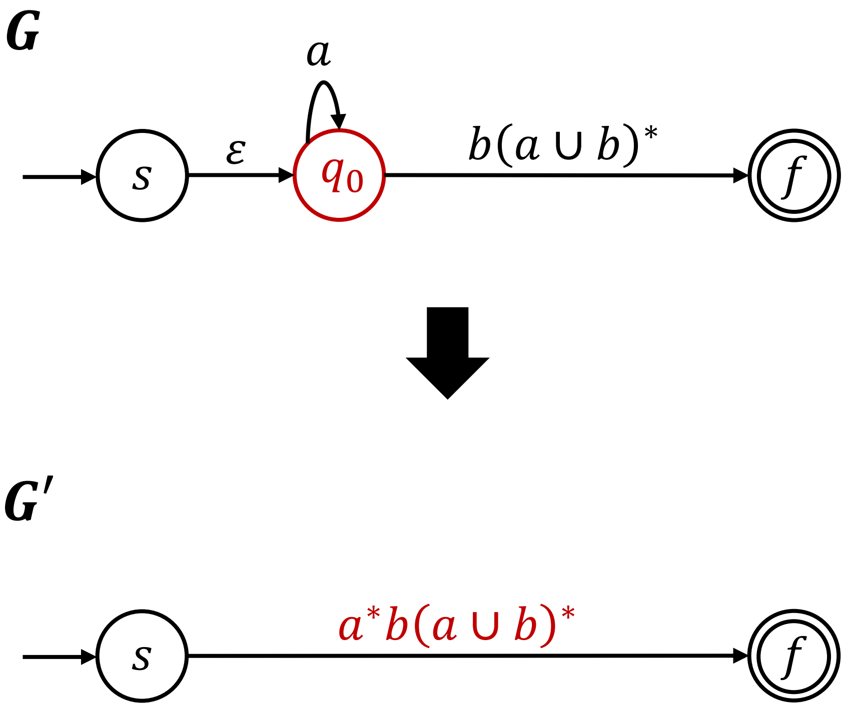
$q_{\text{rip}} = q_1$

G'



GNFA to Regular Expression Example

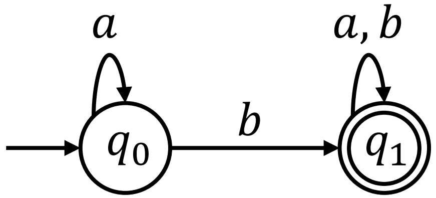




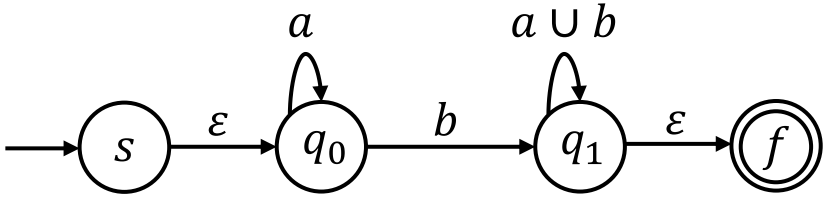
$q_{\text{rip}} = q_0$

DFA to GNFA to Regular Expression Example

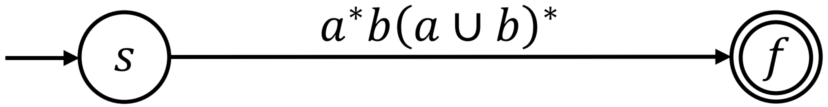
DFA M



GNFA G



Simplified GNFA G



Regular Expression R

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

Equivalence of Regular Expressions

Theorem: A language is regular if and only if there exists some regular expression that describes it

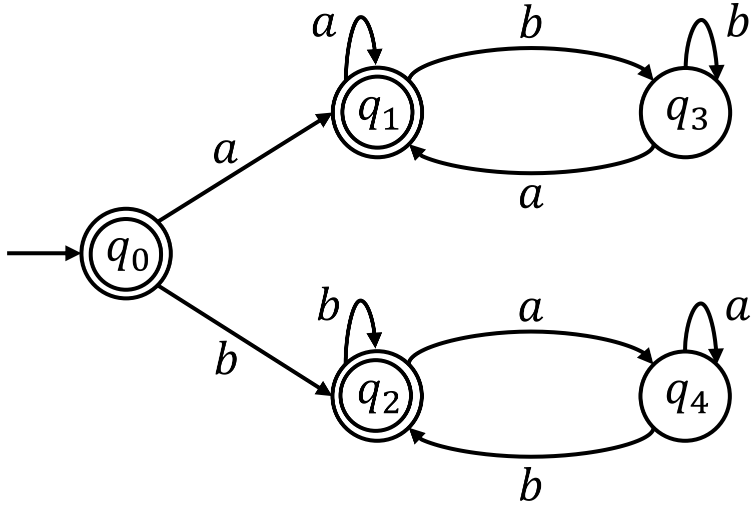
Proof consists of two parts:

1. If a language is described by a **regular expression**, then it is **regular** (can be recognized by a DFA / NFA)
2. If a language is **regular** (can be recognized by a DFA / NFA), then it can be described by a **regular expression**

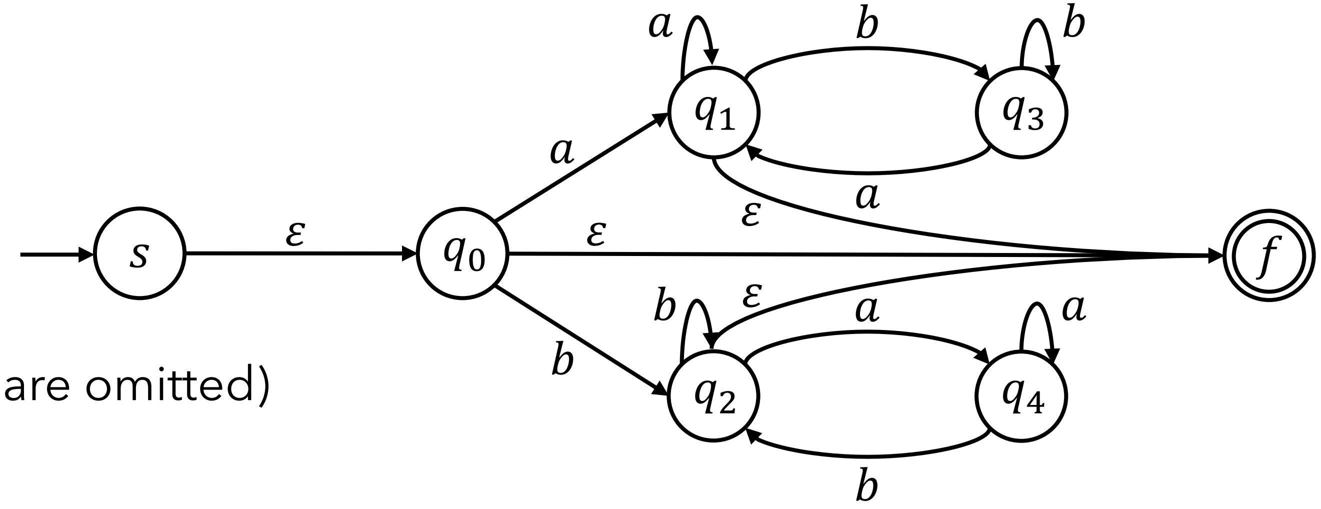
DFA to Regular Expression Example 2

Describe the language recognized by the DFA $\boldsymbol{M}$ as a regular expression $\boldsymbol{R}$

DFA M



GNFA G



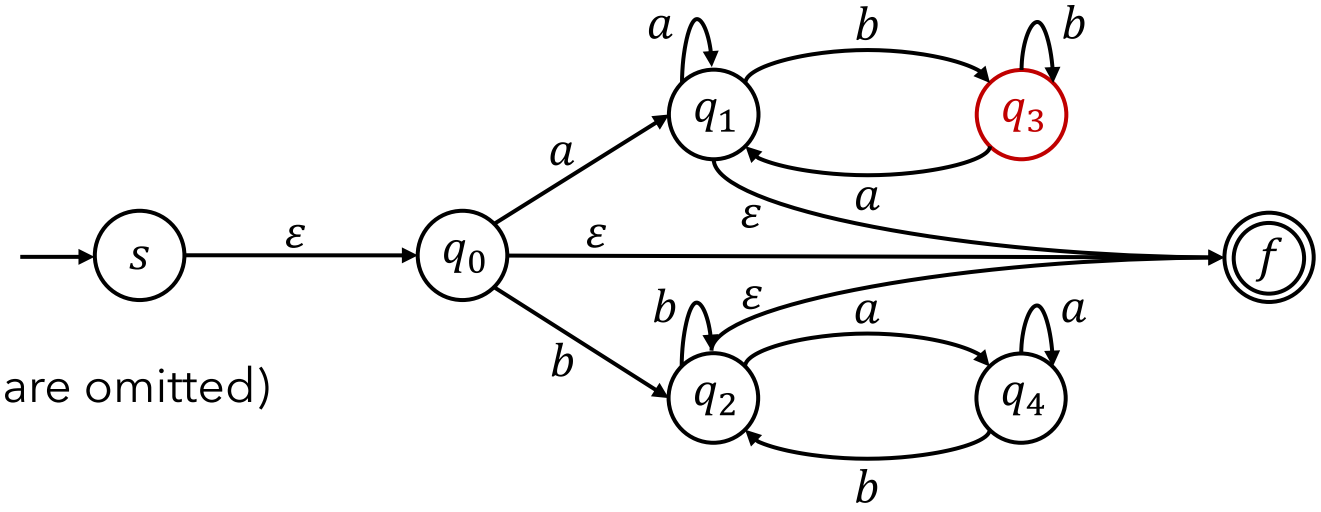
(∅-transitions are omitted)

DFA to Regular Expression Example 2

Describe the language recognized by the DFA $\boldsymbol{M}$ as a regular expression $\boldsymbol{R}$

GNFA G

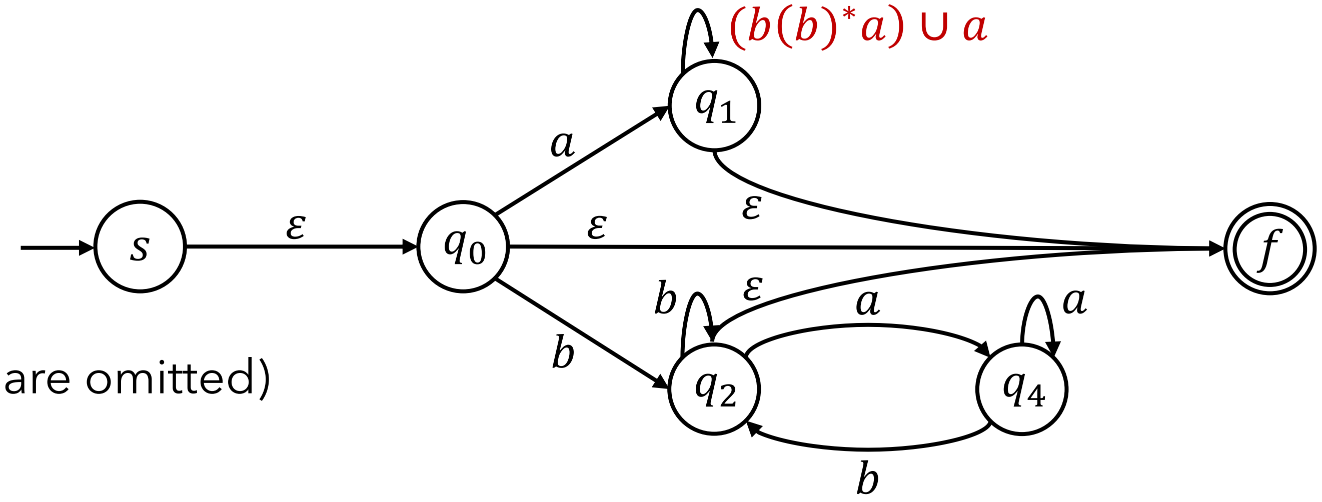
(∅-transitions are omitted)



$\boldsymbol{q}_{\mathbf{rip}} = q_3$

GNFA G'

(∅-transitions are omitted)

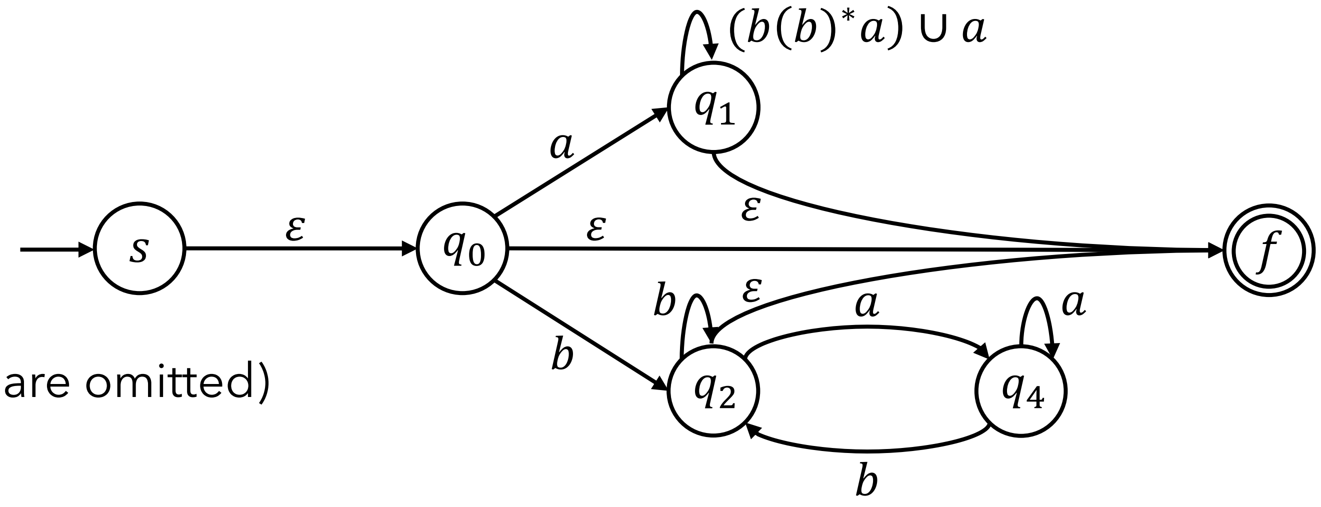


DFA to Regular Expression Example 2

Describe the language recognized by the DFA $\boldsymbol{M}$ as a regular expression $\boldsymbol{R}$

GNFA $\boldsymbol{G}$

(∅-transitions are omitted)



The remaining steps are left for you as an exercise…