Regular Expressions

Lecture 7

October 24, 2022

Objectives

By the end of this lecture, you should be able to

- Describe regular languages using regular expressions.
- Construct an NFA equivalent to a given regular expression.
- Construct a regular expression equivalent to a given NFA.

Regular Operations

- Recall that $\cup$, $\circ$, and * are called regular operations.
- We can describe complex regular languages using expressions involving regular operations.
- **Example**
  - $(\{\texttt{0}\} \cup \{\texttt{1}\})^* \circ \{\texttt{0}\}$
  - This is the language consisting of strings of 0s and 1's that end with a 0.
  - As a shorthand, $(\texttt{0} \cup \texttt{1})^*\texttt{0}$
- The last expression is an example of a **regular expression**.
- $L(R)$ denotes the language described by a regular expression R.
- Precedence: $^* > \circ > \cup$

Formal Definition of Regular Expressions

- Let Σ be an alphabet.
- R is a regular expression over Σ if and only if R is
  1. a for some $a \in \Sigma$,
  2. ε,
  3. $\emptyset$,
  4. $(R_1 \cup R_2)$, where R_1 and R_2 are regular expressions,
  5. $(R_1 \circ R_2)$, where R_1 and R_2 are regular expressions, or
  6. (R_1^*), where R_1 is a regular expression.
- **Note:**
  - $L(a) = \{a\}$.
  - $L(\varepsilon) = \{\varepsilon\}$.
  - $L(\Sigma) = \bigcup_{a \in \Sigma} \{a\} = \Sigma$.

Examples

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

1. $L(\texttt{0}^*\texttt{1}\texttt{0}^*) = \{w \mid w$ contains a single 1$\}$.
2. $L(\Sigma^*\texttt{1}\Sigma^*) = \{w \mid w$ contains at least one 1$\}$.
3. $L(\Sigma^*\texttt{001}\Sigma^*) = \{w \mid w$ contains 001 as a substring $\}$.
4. $L((\texttt{01}^+)^*) = \{w \mid w = \varepsilon$ or starts with a $\texttt{0}$ and each $\texttt{0}$ is followed by at least one $\texttt{1}$ $\}$.
5. $L((\Sigma\Sigma\Sigma)^*) = \{w \mid |w|$ is a multiple of 3$\}$.
6. $L(\texttt{1}^*\varnothing) = \varnothing$.
7. $L(\varnothing^*) = \{\varepsilon\}$.

Example

Describe in English the languages denoted by the following regular expressions.

1. $(\mathtt{11} \cup \mathtt{0})^*$
   -

2. $(\mathtt{1} \cup \mathtt{01} \cup \mathtt{001})^*(\varepsilon \cup \mathtt{0} \cup \mathtt{00})$
   -

Example

Describe in English the languages denoted by the following regular expressions.

1. $(\mathtt{11} \cup \mathtt{0})^*$.
   - $\{w|$ such that every maximal substring of $\mathtt{1}$s in w has an even length$\}$.
   - $\{w|$ contains no $\mathtt{0}$s and has an even length or every $\mathtt{0}$ in w is followed and preceded by an even number of $\mathtt{1}$s$\}$
2. $(\mathtt{1} \cup \mathtt{01} \cup \mathtt{001})^*(\varepsilon \cup \mathtt{0} \cup \mathtt{00})$
   - $\{w|$ w has no more than two $\mathtt{0}$s in succession $\}$.

Equivalence with NFA

Theorem (Sipser 1.54) *A language is regular if and only if some regular expression describes it.*

- The proof has two directions.
- The "if" direction is relatively easy. The "only if" direction is a bit involved.
- We will prove each direction as a separate lemma.

From Regular Expressions to NFA

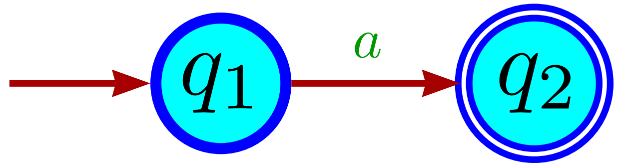
Lemma (Sipser 1.55) *If a language is described by a regular expression, then it is regular.*

Proof

- We will show that, for each possible form of a regular expression, there corresponds an NFA that recognizes the language described by the expression.
- Let R be a regular expression over some alphabet Σ. There are six cases to consider.

Proof: Case 1

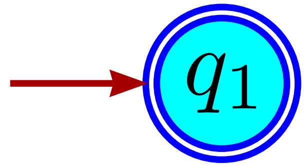
- $R = a$, for some $a \in \Sigma$.
  - Thus, $L(R) = \{a\}$.



- $N = (\{q_1, q_2\}, \Sigma, \delta, q_1, \{q_2\})$
  - Where $\delta(q_1, a) = \{q_2\}$ and $\delta(r, b) = \emptyset$ for $r \neq q_1$ or $b \neq a$

Proof: Case 2

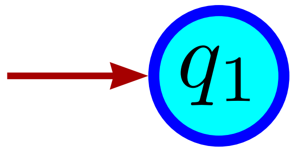
- $R = \varepsilon$.
  - Thus, $L(R) = \{\varepsilon\}$.



- $N = (\{q_1\}, \Sigma, \delta, q_1, \{q_1\})$
  - Where $\delta(r, b) = \varnothing$ for any r and b

Proof: Case 3

- $R = \varnothing$.
  - Thus, $L(R) = \varnothing$.



- $N = (\{q_1\}, \Sigma, \delta, q_1, \varnothing)$
  - Where $\delta(r, b) = \varnothing$ for any r and b

Proof: Cases 4, 5, and 6

4. $R = R_1 \cup R_2$.

5. $R = R_1 \circ R_2$.

6. $R = R_1^*$.

Use the constructions given in the proofs of closure under regular operations.

Example

Draw the state diagram of an NFA that recognizes the language described by the regular expression $(\mathtt{a} \cup \mathtt{b})^*\mathtt{aba}$.

Do it yourself and then check the solution in the text (p. 69).

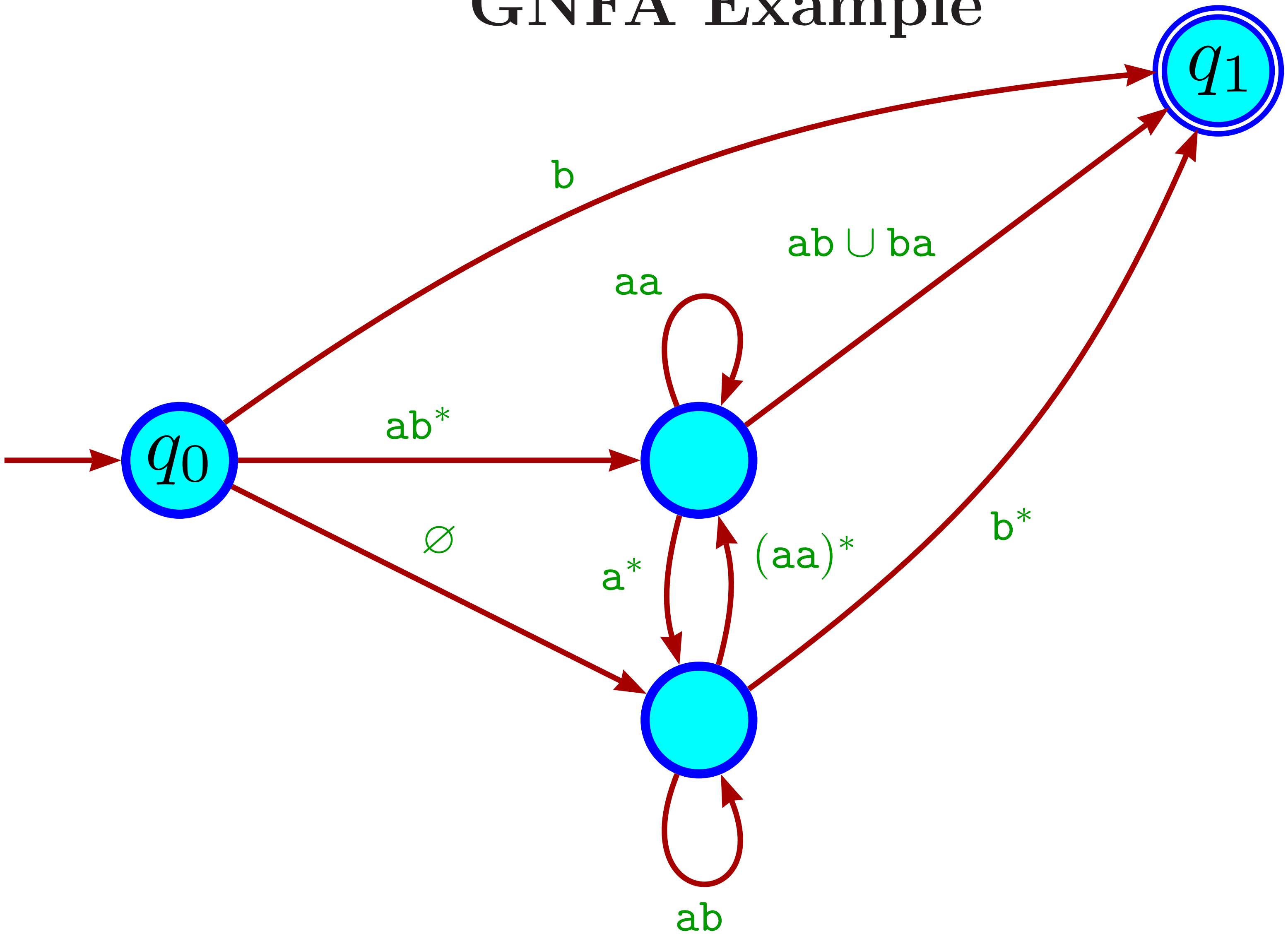
From DFA to Regular Expressions

Lemma (Sipser 1.60) If a language is regular, then it is described by some regular expression.

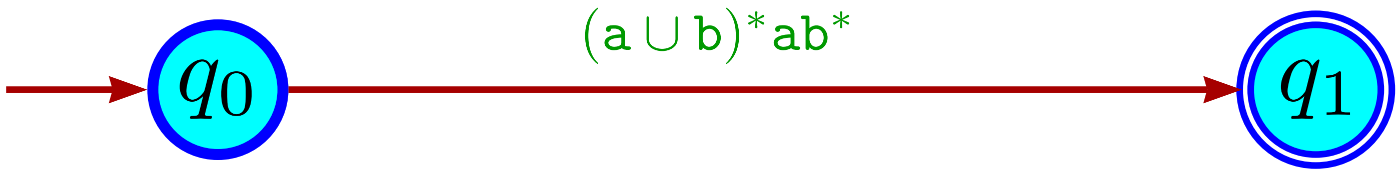
Proof Strategy

- Given a DFA recognizing a regular language L, we will construct a regular expression describing L.
- We will do this by introducing **generalized NFA** (GNFA)—NFA where arcs are labelled by regular expressions.
- We will reduce the DFA into equivalent GNFA by removing one state at a time, adjusting the arcs so that the resulting GNFA recognize the same language.
- When we are left with only the start state and one accept state, we have the target regular expression as the label of the only arc connecting these two states.

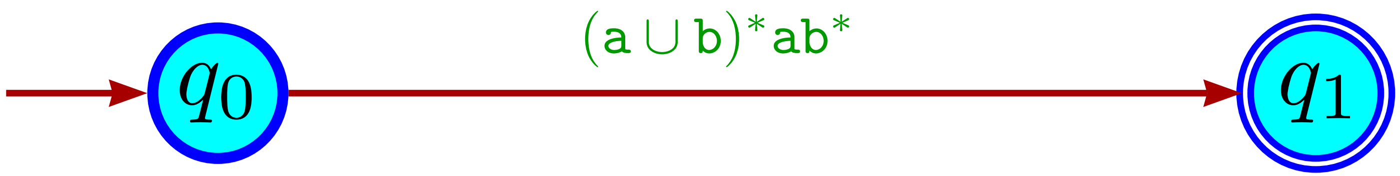
GNFA Example



Another GNFA Example



Another GNFA Example



Note: This GNFA recognizes the language described by the regular expression $(\mathtt{a} \cup \mathtt{b})^*\mathtt{ab}^*$.

Restrictions on GNFA

For convenience, we require GNFA to have the following special features.

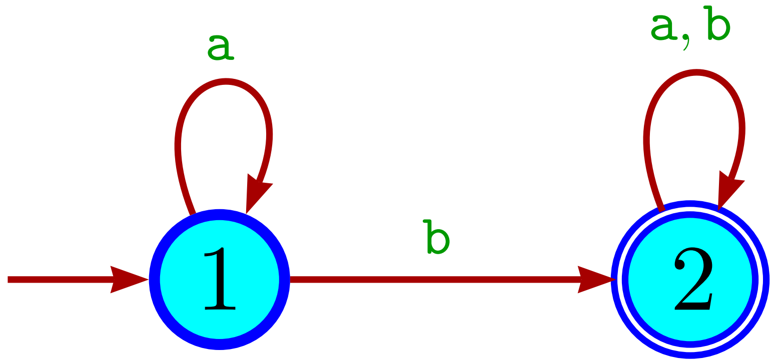
1. The start state has no in-coming transitions.
2. The start state has an out-going transition to every other state.
3. There is one and only one accept state.
4. The accept state has no out-going transitions.
5. The accept state has an in-coming transition from every other state.
6. Except for the start and accept states, every state has one and only one out-going transition to every state (including itself).

From NFA to GNFA

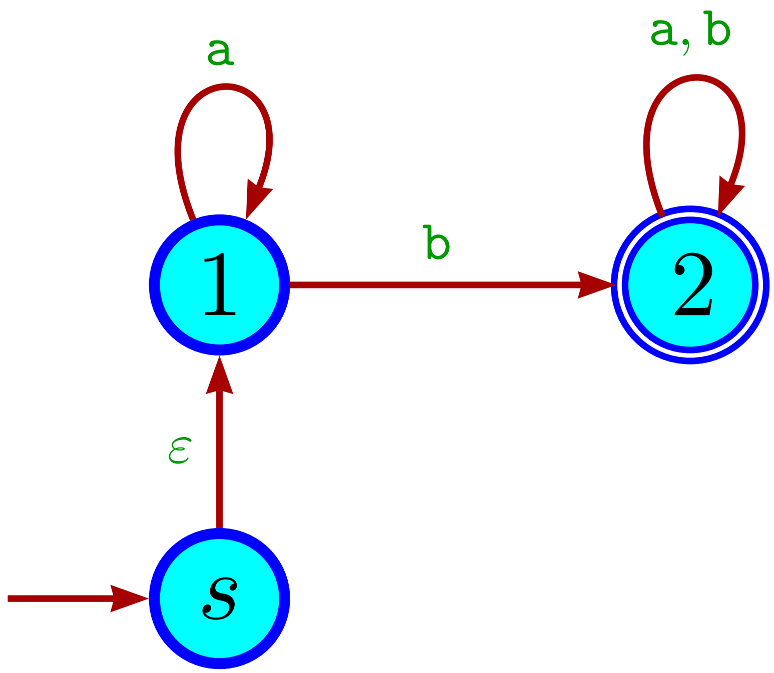
To convert an NFA into a GNFA in the special form:

1. Add a new start state, s, with an ε-transition to the old start state.
2. Add a new accept state, a, with an ε-transition from each of the old accept states.
3. If two states are connected with multiple transitions, replace them by a single transition whose label is the union of the old labels.
4. If two states are not connected, add a transition labelled $\varnothing$ between them (in both directions).
   - A transition labelled $\varnothing$ can never be used.

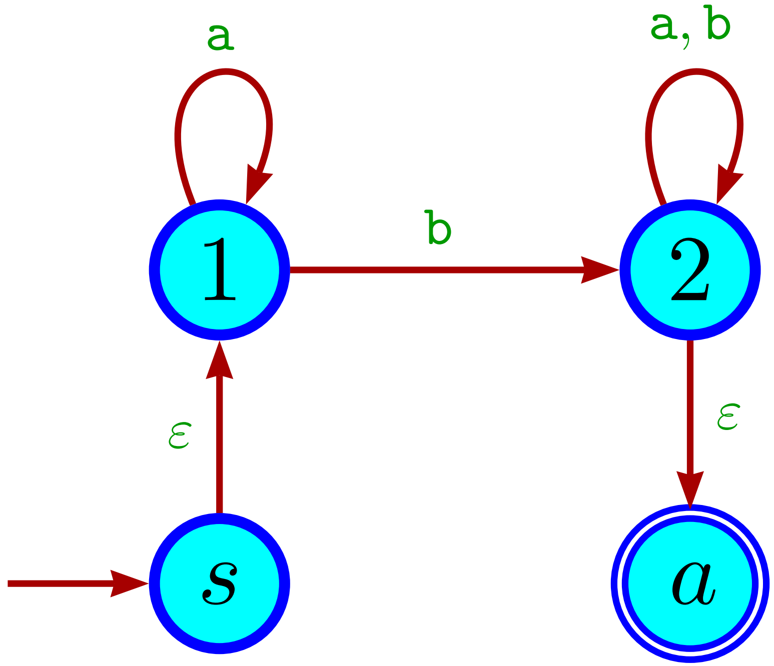
Example



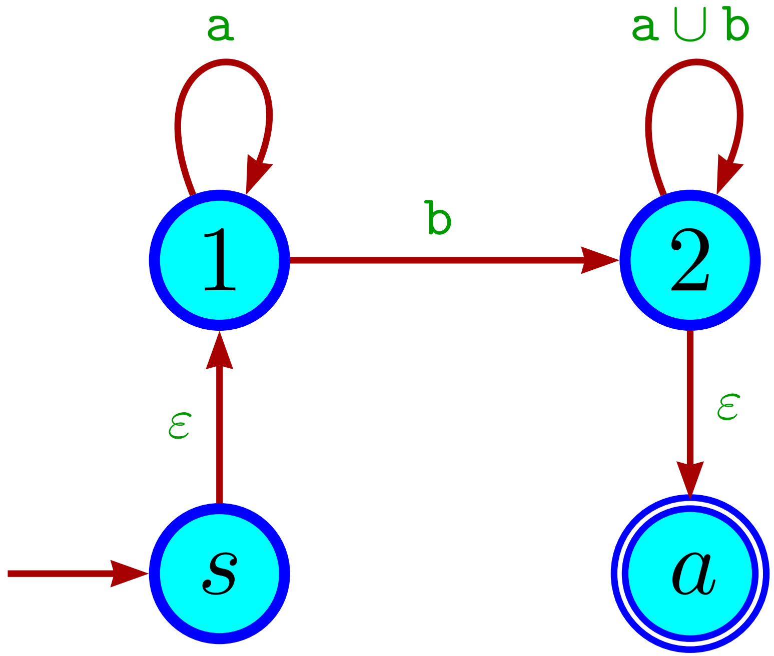
Example



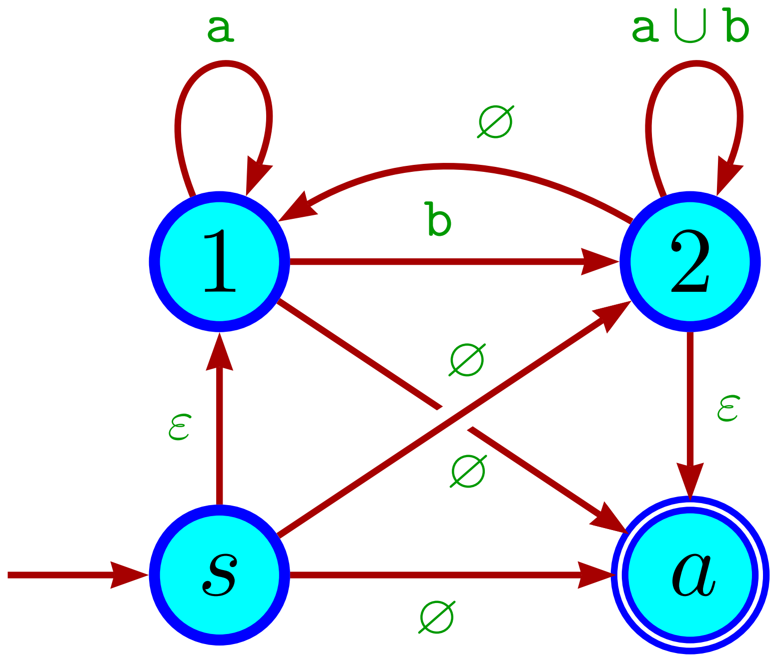
Example



Example



Example



We typically omit the $\emptyset$-transitions in state diagrams. (They still exist though.)

From GNFA to Regular Expressions

To convert a GNFA with k states into a regular expression:

1. If $k = 2$, there are only the states s and a. The label of the transition from s to a is the regular expression equivalent to the GNFA.

2. If $k > 2$ (note that k is at least 2), then construct an equivalent GNFA with $k - 1$ states.

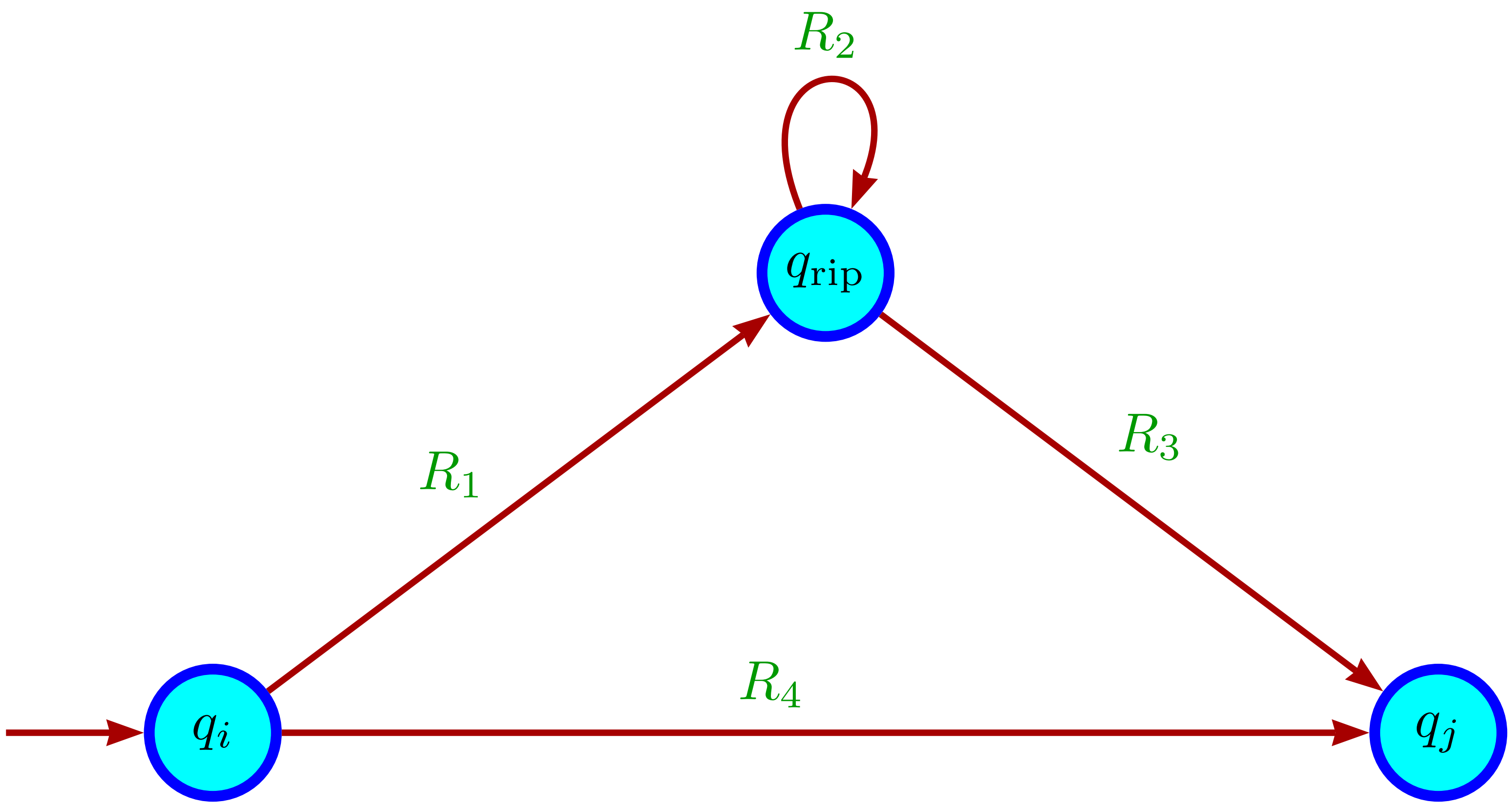
   (a) Choose a state, q_{rip}, other than s and a.

   (b) Rip it out of the automaton.

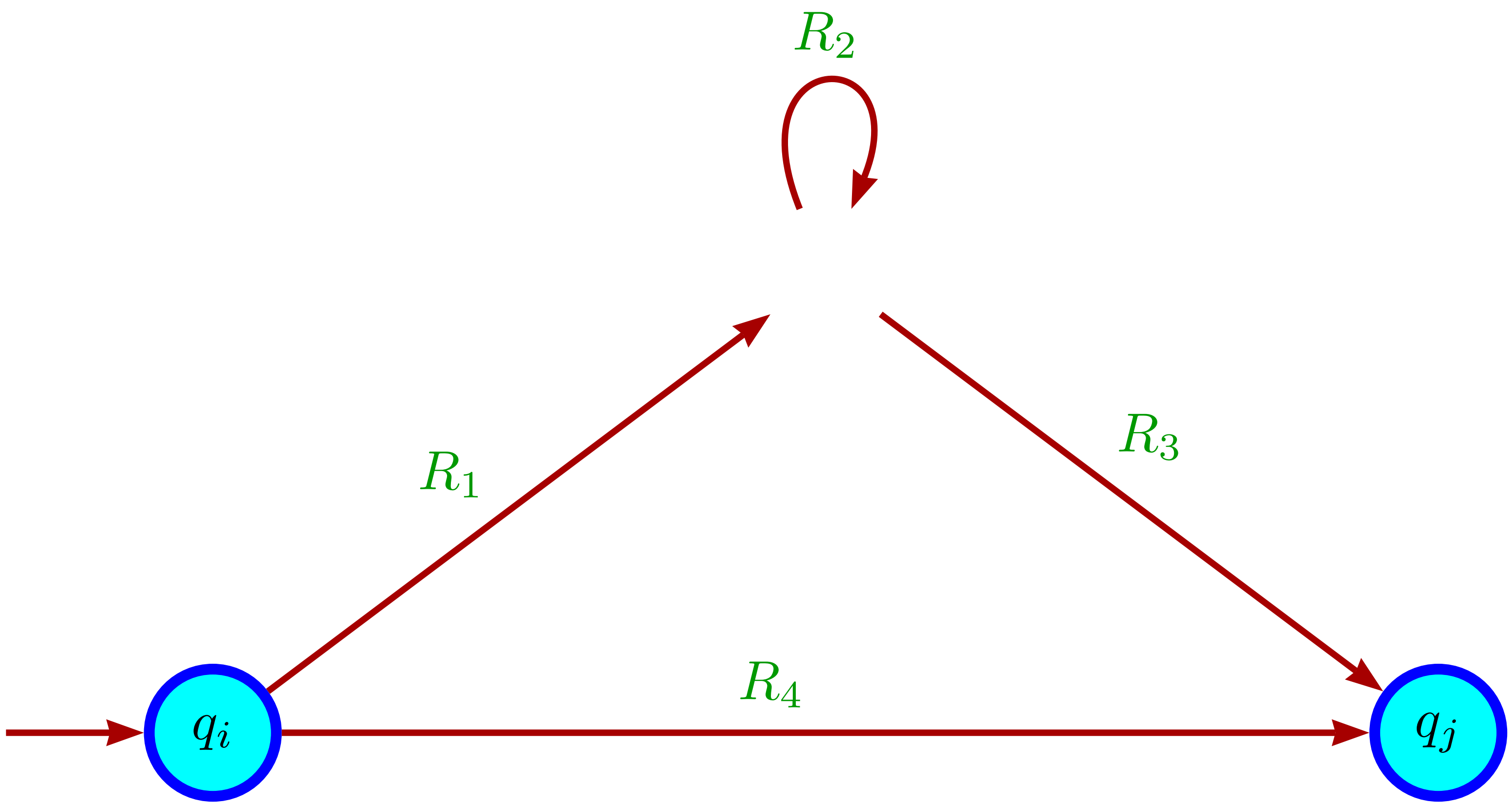
   (c) Reconnect the loose transitions appropriately.

   - For each pair of states, q_i and q_j, change the label of the transition to one that would take the GNFA from q_i to q_j directly or via q_{rip}

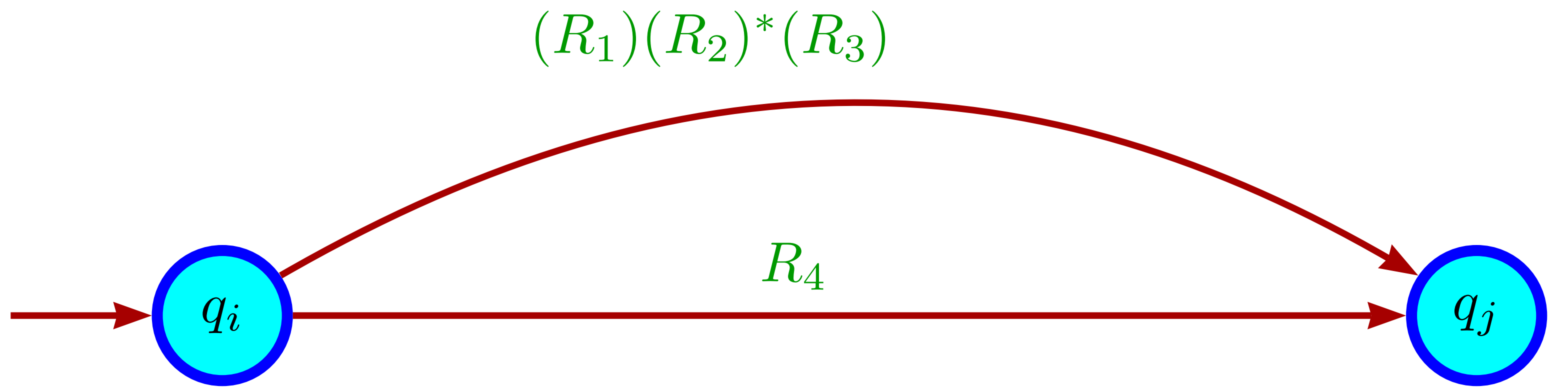
Example



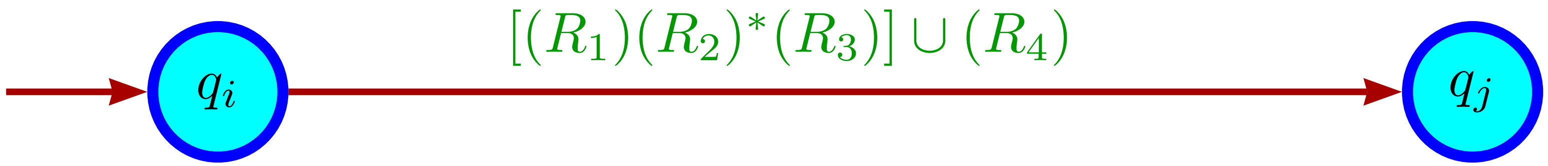
Example



Example



Example



Formal Definition of GNFA

- A GNFA is a 5-tuple $(Q, \Sigma, \delta, s, a)$ where
  1. Q is a finite set of states,
  2. Σ is an alphabet,
  3. $\delta : (Q - \{a\}) \times (Q - \{s\}) \longrightarrow \mathcal{R}$ is the transition function, $\mathcal{R}$ being the set of all regular expressions over Σ,
  4. $s \in Q$ is the start state, and
  5. $a \in Q$ is the accept state.

The Language of a GNFA

A GNFA G accepts a string $w \in \Sigma^*$ if

- $w = w_1 w_2 \cdots w_k$, where each w_i is in Σ^*, and
- there is a sequence of states $q_0, q_1, \ldots, q_k$ such that
  1. $q_0 = s$,
  2. $q_k = a$, and
  3. for each i $(1 \leq i \leq k)$, $w_i \in L(R_i)$, where $R_i = \delta(q_{i-1}, q_i)$.
- $L(G) = \{w |\ G \text{ accepts } w\}$.
- **Note:** If G has only the two states s and a, then $L(G) = L(R)$, where $R = \delta(s, a)$.

From NFA to Regular Expressions

- Let L be a regular language.
- We need to show that there is a regular expression R, such that $L(R) = L$.
- Let M be an NFA that recognizes L (i.e., $L(M) = L$).
- We prove the existence of R by constructing a two-state GNFA equivalent to M.
- First, transform M into a GNFA G (with the special features discussed previously).
- Then, proceed to reducing G into a two-state GNFA.

GNFA Reduction

Convert(G) $\{G = (Q, \Sigma, \delta, s, a)\}$

$k \leftarrow$ number of states of G
if $k = 2$ **then**
  Return $\delta(s, a)$
else
  Select any state $q_{\text{rip}} \in Q - \{s, a\}$
  $Q' \leftarrow Q - \{q_{\text{rip}}\}$
  for all $q_i \in Q' - \{a\}$ **do**
    for all $q_j \in Q' - \{s\}$ **do**
      $R_1 \leftarrow \delta(q_i, q_{\text{rip}})$
      $R_2 \leftarrow \delta(q_{\text{rip}}, q_{\text{rip}})$
      $R_3 \leftarrow \delta(q_{\text{rip}}, q_j)$
      $R_4 \leftarrow \delta(q_i, q_j)$
      $\delta'(q_i, q_j) \leftarrow [(R_1)(R_2)^*(R_3)] \cup (R_4)$
  $G' \leftarrow (Q', \Sigma, \delta', s, a)$
  Return **Convert**(G')

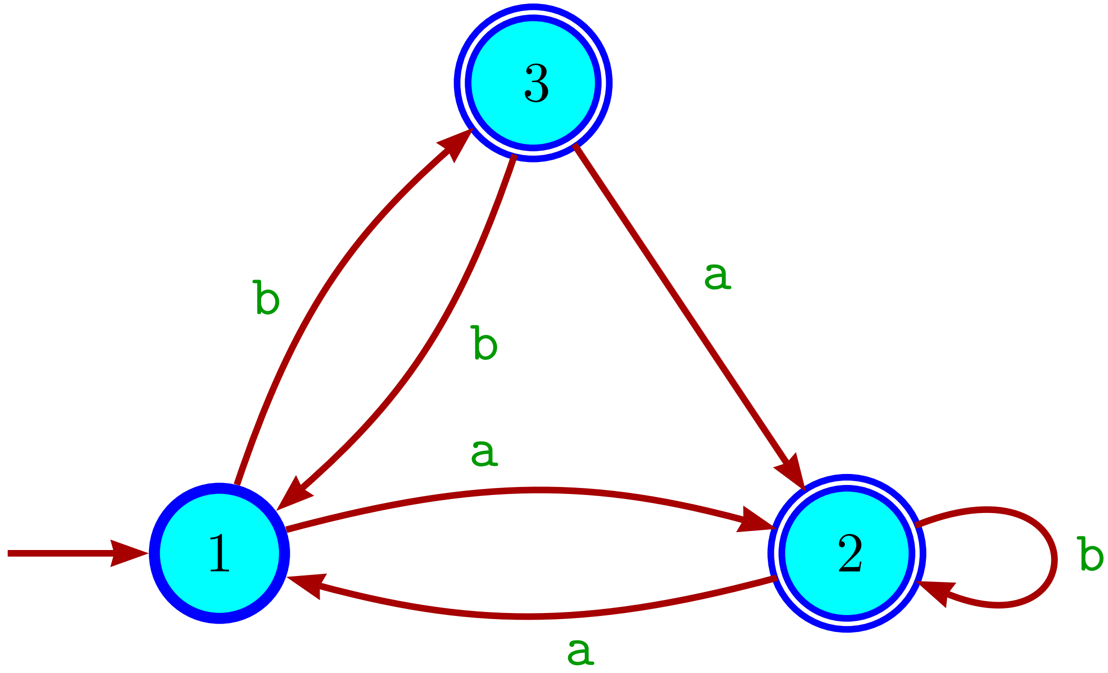
One More Step . . .

Claim (Sipser 1.65) *For any GNFA G,* Convert*(G) is equivalent to G.*

- Check the proof on page 74 of the text.

Example

Find a regular expression equivalent to the following DFA.



Do it yourself.

Next time

- The Pumping Lemma.

Points to take home

- Regular expressions.
- From regular expressions to NFA.
- From NFA to regular expressions.