CSE 135: Introduction to Theory of Computation
Regular Expressions and Regular Languages
(DFA to Regular Expressions)

Sungjin Im

University of California, Merced

02-10-2015
Regular Expressions and Regular Languages
Why do they have such similar names?

Theorem
\[ L \text{ is a regular language if and only if there is a regular expression } R \text{ such that } L(R) = L \]

i.e., Regular expressions have the same “expressive power” as finite automata.

Proof.
▶ Given regular expression \( R \), can construct NFA \( N \) such that \( L(N) = L(R) \)
▶ Given DFA \( M \), will construct regular expression \( R \) such that \( L(M) = L(R) \)
□
Regular Expressions and Regular Languages

Why do they have such similar names?

**Theorem**

$L$ is a regular language if and only if there is a regular expression $R$ such that $L(R) = L$

i.e., Regular expressions have the same “expressive power” as finite automata.

**Proof.**

- Given regular expression $R$, can construct NFA $N$ such that $L(N) = L(R)$
- Given DFA $M$, will construct regular expression $R$ such that $L(M) = L(R)$
DFA to Regular Expression

Given DFA $M$, will construct regular expression $R$ such that $L(M) = L(R)$. 
DFA to Regular Expression

Given DFA $M$, will construct regular expression $R$ such that $L(M) = L(R)$. In two steps:
DFA to Regular Expression

- Given DFA $M$, will construct regular expression $R$ such that $L(M) = L(R)$. In two steps:
  - Construct a “Generalized NFA” (GNFA) $G$ from the DFA $M$
  - And then convert $G$ to a regex $R$
Generalized NFA

- A GNFA is similar to an NFA, but:
Generalized NFA

- A GNFA is similar to an NFA, but:
  - There is a single accept state.
Generalized NFA

- A GNFA is similar to an NFA, but:
  - There is a single accept state.
  - The start state has no incoming transitions, and the accept state has no outgoing transitions.
Generalized NFA

- A GNFA is similar to an NFA, but:
  - There is a single accept state.
  - The start state has no incoming transitions, and the accept state has no outgoing transitions.
    - These are “cosmetic changes”: Any NFA can be converted to an equivalent NFA of this kind.
Generalized NFA

- A GNFA is similar to an NFA, but:
  - There is a single accept state.
  - The start state has no incoming transitions, and the accept state has no outgoing transitions.
    - These are “cosmetic changes”: Any NFA can be converted to an equivalent NFA of this kind.
  - The transitions are labeled not by characters in the alphabet, but by regular expressions.
Generalized NFA

- A GNFA is similar to an NFA, but:
  - There is a single accept state.
  - The start state has no incoming transitions, and the accept state has no outgoing transitions.
    - These are “cosmetic changes”: Any NFA can be converted to an equivalent NFA of this kind.
  - The transitions are labeled not by characters in the alphabet, but by regular expressions.
    - For every pair of states \((q_1, q_2)\), the transition from \(q_1\) to \(q_2\) is labeled by a regular expression \(\rho(q_1, q_2)\).
Generalized NFA

- A GNFA is similar to an NFA, but:
  - There is a single accept state.
  - The start state has no incoming transitions, and the accept state has no outgoing transitions.
    - These are “cosmetic changes”: Any NFA can be converted to an equivalent NFA of this kind.
  - The transitions are labeled not by characters in the alphabet, but by regular expressions.
    - For every pair of states \((q_1, q_2)\), the transition from \(q_1\) to \(q_2\) is labeled by a regular expression \(\rho(q_1, q_2)\).
  - “Generalized NFA” because a normal NFA has transitions labeled by \(\epsilon\), elements in \(\Sigma\) (a union of elements, if multiple edges between a pair of states) and \(\emptyset\) (missing edges).
Generalized NFA

- Transition: GNFA non-deterministically reads a block of characters from the input, chooses an edge from the current state $q_1$ to another state $q_2$, and if the block of symbols matches the regex $\rho(q_1, q_2)$, then moves to $q_2$. 

- Acceptance: $G$ accepts $w$ if there exists some sequence of valid transitions such that on starting from the start state, and after finishing the entire input, $G$ is in the accept state.
Generalized NFA

- Transition: GNFA non-deterministically reads a block of characters from the input, chooses an edge from the current state $q_1$ to another state $q_2$, and if the block of symbols matches the regex $\rho(q_1, q_2)$, then moves to $q_2$.

- Acceptance: $G$ accepts $w$ if there exists some sequence of valid transitions such that on starting from the start state, and after finishing the entire input, $G$ is in the accept state.
Generalized NFA: Example

Example GNFA $G$

Accepting run of $G$ on 11110100 is
Generalized NFA: Example

Example GNFA $G$

Accepting run of $G$ on 11110100 is

$q_0 \xrightarrow{1} G q_1$
Generalized NFA: Example

Example GNFA $G$

Accepting run of $G$ on 11110100 is

$q_0 \xrightarrow{1} G q_1 \xrightarrow{11} G q_1$
Generalized NFA: Example

Example GNFA $G$

Accepting run of $G$ on 11110100 is

$q_0 \xrightarrow{1} G q_1 \xrightarrow{11} G q_1 \xrightarrow{101} G q_1$
Generalized NFA: Example

Example GNFA $G$

Accepting run of $G$ on 11110100 is

$q_0 \xrightarrow{1} G q_1 \xrightarrow{11} G q_1 \xrightarrow{101} G q_1 \xrightarrow{00} G q_2$
Generalized NFA: Definition

Definition
A generalized nondeterministic finite automaton (GNFA) is $G = (Q, \Sigma, q_0, q_F, \rho)$, where
- $Q$ is the finite set of states
- $\Sigma$ is the finite alphabet
- $q_0 \in Q$ initial state
- $q_F \in Q$, a single accepting state
- $\rho : (Q \setminus \{q_F\}) \times (Q \setminus \{q_0\}) \rightarrow R^\Sigma$, where $R^\Sigma$ is the set of all regular expressions over the alphabet $\Sigma$
Generalized NFA: Definition

Definition
A generalized nondeterministic finite automaton (GNFA) is

\[ G = (Q, \Sigma, q_0, q_F, \rho) \]

where

- \( Q \) is the finite set of states
- \( \Sigma \) is the finite alphabet
- \( q_0 \in Q \) initial state
- \( q_F \in Q \), a single accepting state

\( \rho: (Q \setminus \{q_F\}) \times (Q \setminus \{q_0\}) \rightarrow \Sigma \)
Generalized NFA: Definition

Definition
A generalized nondeterministic finite automaton (GNFA) is 
\[ G = (Q, \Sigma, q_0, q_F, \rho), \]
where
- \( Q \) is the finite set of states
- \( \Sigma \) is the finite alphabet
- \( q_0 \in Q \) initial state
- \( q_F \in Q \), a single accepting state
- \( \rho : (Q \setminus \{q_F\}) \times (Q \setminus \{q_0\}) \rightarrow \mathcal{R}_\Sigma \), where \( \mathcal{R}_\Sigma \) is the set of all regular expressions over the alphabet \( \Sigma \)
Generalized NFA: Definition

Definition
For a GNFA \( M = (Q, \Sigma, q_0, q_F, \rho) \) and string \( w \in \Sigma^* \), we say \( M \) accepts \( w \) iff there exist \( x_1, \ldots, x_t \in \Sigma^* \) and states \( r_0, \ldots, r_t \) such that
Generalized NFA: Definition

Definition
For a GNFA $M = (Q, \Sigma, q_0, q_F, \rho)$ and string $w \in \Sigma^*$, we say $M$ accepts $w$ iff there exist $x_1, \ldots, x_t \in \Sigma^*$ and states $r_0, \ldots, r_t$ such that

$\Rightarrow w = x_1x_2x_3 \cdots x_t$
Generalized NFA: Definition

**Definition**
For a GNFA $M = (Q, \Sigma, q_0, q_F, \rho)$ and string $w \in \Sigma^*$, we say $M$ accepts $w$ iff there exist $x_1, \ldots, x_t \in \Sigma^*$ and states $r_0, \ldots, r_t$ such that

- $w = x_1x_2x_3 \cdots x_t$
- $r_0 = q_0$ and $r_t = q_F$
Definition

For a GNFA $M = (Q, \Sigma, q_0, q_F, \rho)$ and string $w \in \Sigma^*$, we say $M$ accepts $w$ iff there exist $x_1, \ldots, x_t \in \Sigma^*$ and states $r_0, \ldots, r_t$ such that

$\begin{align*}
\text{w} &= x_1 x_2 x_3 \cdots x_t \\
\text{r}_0 &= q_0 \text{ and } \text{r}_t &= q_F \\
\text{for each } i \in [1, t], x_i &\in L(\rho(r_{i-1}, r_i)),
\end{align*}$
A DFA $M = (Q, \Sigma, \delta, q_0, F)$ can be easily converted to an equivalent GNFA $G = (Q', \Sigma, q'_0, q'_F, \rho)$:

- $Q' = Q \cup \{q'_0, q'_F\}$
  where $Q \cap \{q'_0, q'_F\} = \emptyset$
- $\rho(q_1, q_2) = \begin{cases} \epsilon, & \text{if } q_1 = q'_0 \text{ and } q_2 = q_0 \\ \epsilon, & \text{if } q_1 \in F \text{ and } q_2 = q'_F \\ \bigcup \{a \mid \delta(q_1, a) = q_2\}, & \text{otherwise} \end{cases}$
Converting DFA to GNFA

A DFA $M = (Q, \Sigma, \delta, q_0, F)$ can be easily converted to an equivalent GNFA $G = (Q', \Sigma, q'_0, q'_F, \rho)$:

1. $Q' = Q \cup \{q'_0, q'_F\}$ where $Q \cap \{q'_0, q'_F\} = \emptyset$
Converting DFA to GNFA

A DFA $M = (Q, \Sigma, \delta, q_0, F)$ can be easily converted to an equivalent GNFA $G = (Q', \Sigma, q'_0, q'_F, \rho)$:

- $Q' = Q \cup \{q'_0, q'_F\}$ where $Q \cap \{q'_0, q'_F\} = \emptyset$

- $\rho(q_1, q_2) = \begin{cases} 
\epsilon, & \text{if } q_1 = q'_0 \text{ and } q_2 = q_0 \\
\epsilon, & \text{if } q_1 \in F \text{ and } q_2 = q'_F \\
\bigcup \{a|\delta(q_1, a) = q_2\} a & \text{otherwise}
\end{cases}$
Converting DFA to GNFA

A DFA $M = (Q, \Sigma, \delta, q_0, F)$ can be easily converted to an equivalent GNFA $G = (Q', \Sigma, q'_0, q'_F, \rho)$:

1. $Q' = Q \cup \{q'_0, q'_F\}$ where $Q \cap \{q'_0, q'_F\} = \emptyset$

2. $\rho(q_1, q_2) = \begin{cases} 
\epsilon, & \text{if } q_1 = q'_0 \text{ and } q_2 = q_0 \\
\epsilon, & \text{if } q_1 \in F \text{ and } q_2 = q'_F \\
\bigcup \{a | \delta(q_1, a) = q_2\} & \text{otherwise}
\end{cases}$

Prove: $L(G) = L(M)$. 
Suppose $G$ is a GNFA with only two states, $q_0$ and $q_F$.

Then $L(R) = L(G)$ where $R = \rho(q_0, q_F)$.

How about $G$ with three states?

Plan: Reduce any GNFA $G$ with $k > 2$ states to an equivalent GFA with $k - 1$ states.
GNFA to Regex

- Suppose $G$ is a GNFA with only two states, $q_0$ and $q_F$. 

- How about $G$ with three states?

Plan: Reduce any GNFA $G$ with $k > 2$ states to an equivalent GFA with $k - 1$ states.
GNFA to Regex

- Suppose $G$ is a GNFA with only two states, $q_0$ and $q_F$.
- Then $L(R) = L(G)$ where $R = \rho(q_0, q_F)$. 

Plan: Reduce any GNFA $G$ with $k > 2$ states to an equivalent GFA with $k - 1$ states.
GNFA to Regex

- Suppose $G$ is a GNFA with only two states, $q_0$ and $q_F$.
- Then $L(R) = L(G)$ where $R = \rho(q_0, q_F)$.
- How about $G$ with three states?
Suppose $G$ is a GNFA with only two states, $q_0$ and $q_F$.
Then $L(R) = L(G)$ where $R = \rho(q_0, q_F)$.
How about $G$ with three states?

Plan: Reduce any GNFA $G$ with $k > 2$ states to an equivalent GFA with $k - 1$ states.
GNFA to Regex

- Suppose $G$ is a GNFA with only two states, $q_0$ and $q_F$.
- Then $L(R) = L(G)$ where $R = \rho(q_0, q_F)$.
- How about $G$ with three states?
GNFA to Regex

- Suppose $G$ is a GNFA with only two states, $q_0$ and $q_F$.
- Then $L(R) = L(G)$ where $R = \rho(q_0, q_F)$.
- How about $G$ with three states?

- Plan: Reduce any GNFA $G$ with $k > 2$ states to an equivalent GFA with $k - 1$ states.
Definition (Deleting a GNFA State)

Given GNFA $G = (Q, \Sigma, q_0, q_F, \rho)$ with $|Q| > 2$, and any state $q^* \in Q \setminus \{q_0, q_F\}$, define GNFA $\text{rip}(G, q^*) = (Q', \Sigma, q_0, q_F, \rho')$ as follows:

\[ Q' = Q \setminus \{q^*\} \]
\[ \rho'(q_1, q_2) = (R_1 R_2 R_3) \cup R_4 \]

where $R_1 = \rho(q_1, q^*)$, $R_2 = \rho(q^*, q^*)$, $R_3 = \rho(q^*, q_2)$ and $R_4 = \rho(q_1, q_2)$. 

Claim. For any $q^* \in Q \setminus \{q_0, q_F\}$, $G$ and $\text{rip}(G, q^*)$ are equivalent.
Definition (Deleting a GNFA State)

Given GNFA $G = (Q, \Sigma, q_0, q_F, \rho)$ with $|Q| > 2$, and any state $q^* \in Q \setminus \{q_0, q_F\}$, define GNFA $\text{rip}(G, q^*) = (Q', \Sigma, q_0, q_F, \rho')$ as follows:

$\quad Q' = Q \setminus \{q^*\}$. 

Claim. For any $q^* \in Q \setminus \{q_0, q_F\}$, $G$ and $\text{rip}(G, q^*)$ are equivalent.
Definition (Deleting a GNFA State)

Given GNFA $G = (Q, \Sigma, q_0, q_F, \rho)$ with $|Q| > 2$, and any state $q^* \in Q \setminus \{q_0, q_F\}$, define GNFA $\text{rip}(G, q^*) = (Q', \Sigma, q_0, q_F, \rho')$ as follows:

- $Q' = Q \setminus \{q^*\}$.
- For any $(q_1, q_2) \in Q' \setminus \{q_F\} \times Q' \setminus \{q_0\}$ (possibly $q_1 = q_2$), let

$$\rho'(q_1, q_2) = (R_1 R_2^* R_3) \cup R_4,$$

where $R_1 = \rho(q_1, q^*)$, $R_2 = \rho(q^*, q^*)$, $R_3 = \rho(q^*, q_2)$ and $R_4 = \rho(q_1, q_2)$. 

Claim. For any $q^* \in Q \setminus \{q_0, q_F\}$, $G$ and $\text{rip}(G, q^*)$ are equivalent.
GNFA to Regex: From $k$ states to $k - 1$ states

Definition (Deleting a GNFA State)

Given GNFA $G = (Q, \Sigma, q_0, q_F, \rho)$ with $|Q| > 2$, and any state $q^* \in Q \setminus \{q_0, q_F\}$, define GNFA $\text{rip}(G, q^*) = (Q', \Sigma, q_0, q_F, \rho')$ as follows:

- $Q' = Q \setminus \{q^*\}$.
- For any $(q_1, q_2) \in Q' \setminus \{q_F\} \times Q' \setminus \{q_0\}$ (possibly $q_1 = q_2$), let

  $$\rho'(q_1, q_2) = (R_1 R_* R_3) \cup R_4,$$

  where $R_1 = \rho(q_1, q^*)$, $R_2 = \rho(q^*, q^*)$, $R_3 = \rho(q^*, q_2)$ and $R_4 = \rho(q_1, q_2)$.

Claim. For any $q^* \in Q \setminus \{q_0, q_F\}$, $G$ and $\text{rip}(G, q^*)$ are equivalent.
GNFA to Regex: From $k$ states to $k - 1$ states

$w \in L(G) \implies w \in L(G')$

Proof.
GNFA to Regex: From \( k \) states to \( k - 1 \) states

\[ w \in L(G) \implies w \in L(G') \]

Proof.

\[ \begin{align*}
\triangleright w \in L(G) & \implies w = x_1x_2x_3 \cdots x_t, \text{ and a sequence of states } \\
q_0 = r_0, r_1, \ldots, r_t = q_F \text{ s.t. } x_i \in L(\rho(r_{i-1}, r_i)).
\end{align*} \]
Proof.

- $w \in L(G) \implies w = x_1x_2x_3 \cdots x_t$, and a sequence of states $q_0 = r_0, r_1, \ldots, r_t = q_F$ s.t. $x_i \in L(\rho(r_{i-1}, r_i))$.
- Let $(q_0 = s_0, \ldots, s_d = q_F)$ be the subsequence of states obtained by deleting all occurrences of $q^*$.
GNFA to Regex: From $k$ states to $k - 1$ states

$w \in L(G) \implies w \in L(G')$

Proof.

- $w \in L(G) \implies w = x_1x_2x_3\cdots x_t$, and a sequence of states $q_0 = r_0, r_1, \ldots, r_t = q_F$ s.t. $x_i \in L(\rho(r_{i-1}, r_i))$.

- Let $(q_0 = s_0, \ldots, s_d = q_F)$ be the subsequence of states obtained by deleting all occurrences of $q^*$.

- For any run of $q^*$ — i.e., an interval $[a, b]$ s.t. $r_{a-1} \neq q^* = r_a = \ldots = r_{b-1} \neq r_b$ — let $x_{[a,b]} = x_a \cdots x_b$. 
GNFA to Regex: From $k$ states to $k - 1$ states
\[ w \in L(G) \implies w \in L(G') \]

Proof.

- \( w \in L(G) \implies w = x_1 x_2 x_3 \cdots x_t \), and a sequence of states \( q_0 = r_0, r_1, \ldots, r_t = q_F \) s.t. \( x_i \in L(\rho(r_{i-1}, r_i)) \).
- Let \( (q_0 = s_0, \ldots, s_d = q_F) \) be the subsequence of states obtained by deleting all occurrences of \( q^* \).
- For any run of \( q^* \) — i.e., an interval \([a, b]\) s.t. \( r_{a-1} \neq q^* = r_a = \ldots = r_{b-1} \neq r_b \) — let \( x_{[a,b]} = x_a \cdots x_b \).
- If \( s_{j-1} = r_{a-1} \) and \( s_j = r_b \), then \( x_{[a,b]} \in L(\rho'(s_{j-1}, s_j)) \).
GNFA to Regex: From $k$ states to $k-1$ states

$w \in L(G) \implies w \in L(G')$

Proof.

- $w \in L(G) \implies w = x_1x_2x_3 \cdots x_t$, and a sequence of states $q_0 = r_0, r_1, \ldots, r_t = q_F$ s.t. $x_i \in L(\rho(r_{i-1}, r_i))$.
- Let $(q_0 = s_0, \ldots, s_d = q_F)$ be the subsequence of states obtained by deleting all occurrences of $q^*$.
- For any run of $q^*$ — i.e., an interval $[a, b]$ s.t. $r_{a-1} \neq q^* = r_a = \ldots = r_{b-1} \neq r_b$ — let $x_{[a,b]} = x_a \cdots x_b$.
- If $s_{j-1} = r_{a-1}$ and $s_j = r_b$, then $x_{[a,b]} \in L(\rho'(s_{j-1}, s_j))$
  - Let $R_1 = \rho(s_{j-1}, q^*), R_2 = \rho(q^*, q^*), R_3 = \rho(q^*, s_j)$ and $R_4 = \rho(s_{j-1}, s_j)$. Then $\rho'(s_{j-1}, s_j) = R_4 \cup (R_1R_2^*R_3)$. 

\[
\implies w \in L(G) \implies w \in L(G')
\]
GNFA to Regex: From $k$ states to $k-1$ states

\[ w \in L(G) \implies w \in L(G') \]

Proof.

- $w \in L(G) \implies w = x_1x_2x_3 \cdots x_t$, and a sequence of states \( q_0 = r_0, r_1, \ldots, r_t = q_F \) s.t. \( x_i \in L(\rho(r_{i-1}, r_i)) \).
- Let \((q_0 = s_0, \ldots, s_d = q_F)\) be the subsequence of states obtained by deleting all occurrences of \( q^* \).
- For any run of \( q^* \) — i.e., an interval \([a, b]\) s.t. \( r_{a-1} \neq q^* = r_a = \ldots = r_{b-1} \neq r_b \) — let \( x_{[a,b]} = x_a \cdots x_b \).
- If \( s_{j-1} = r_{a-1} \) and \( s_j = r_b \), then \( x_{[a,b]} \in L(\rho'(s_{j-1}, s_j)) \)
  - Let \( R_1 = \rho(s_{j-1}, q^*), R_2 = \rho(q^*, q^*), R_3 = \rho(q^*, s_j) \) and \( R_4 = \rho(s_{j-1}, s_j) \). Then \( \rho'(s_{j-1}, s_j) = R_4 \cup (R_1R_2^*R_3) \).
- Case \( a = b \). \((s_{j-1}, s_j) = (r_{b-1}, r_b)\) and \( x_{[a,b]} = x_b \in L(R_4) \).
GNFA to Regex: From $k$ states to $k - 1$ states

$w \in L(G) \implies w \in L(G')$

Proof.

- $w \in L(G) \implies w = x_1 x_2 x_3 \cdots x_t$, and a sequence of states
  $q_0 = r_0, r_1, \ldots, r_t = q_F$ s.t. $x_i \in L(\rho(r_{i-1}, r_i))$.

- Let $(q_0 = s_0, \ldots, s_d = q_F)$ be the subsequence of states
  obtained by deleting all occurrences of $q^*$.

- For any run of $q^*$ — i.e., an interval $[a, b]$ s.t.
  $r_{a-1} \neq q^* = r_a = \ldots = r_{b-1} \neq r_b$ — let $x_{[a,b]} = x_a \cdots x_b$.

- If $s_{j-1} = r_{a-1}$ and $s_j = r_b$, then $x_{[a,b]} \in L(\rho'(s_{j-1}, s_j))$
  - Let $R_1 = \rho(s_{j-1}, q^*), R_2 = \rho(q^*, q^*), R_3 = \rho(q^*, s_j)$ and
    $R_4 = \rho(s_{j-1}, s_j)$. Then $\rho'(s_{j-1}, s_j) = R_4 \cup (R_1 R_2^* R_3)$.
  - Case $a = b$. $(s_{j-1}, s_j) = (r_{b-1}, r_b)$ and $x_{[a,b]} = x_b \in L(R_4)$.
  - Case $a = b + 1 + u$. $x_a \in L(R_1), x_{a+1}, \ldots, x_{b-1} \in L(R_2)$ and
    $x_b \in L(R_3)$. So $x_{[a,b]} \in L(R_1 R_2^u R_3)$. 

\[\text{...}\]
GNFA to Regex: From $k$ states to $k-1$ states

$w \in L(G) \implies w \in L(G')$

Proof.

- $w \in L(G)$ $\implies$ $w = x_1 x_2 x_3 \cdots x_t$, and a sequence of states $q_0 = r_0, r_1, \ldots, r_t = q_F$ s.t. $x_i \in L(\rho(r_{i-1}, r_i))$.
- Let $(q_0 = s_0, \ldots, s_d = q_F)$ be the subsequence of states obtained by deleting all occurrences of $q^*$.
- For any run of $q^*$ — i.e., an interval $[a, b]$ s.t. $r_{a-1} \neq q^* = r_a = \ldots = r_{b-1} \neq r_b$ — let $x_{[a,b]} = x_a \cdots x_b$.
- If $s_{j-1} = r_{a-1}$ and $s_j = r_b$, then $x_{[a,b]} \in L(\rho'(s_{j-1}, s_j))$
- Let $y_1, \ldots, y_d$ be the sequence of blocks of the form $x_{[a,b]}$. 
GNFA to Regex: From \( k \) states to \( k - 1 \) states

\[ w \in L(G) \implies w \in L(G') \]

Proof.

\( w \in L(G) \implies w = x_1x_2x_3 \cdots x_t \), and a sequence of states \( q_0 = r_0, r_1, \ldots, r_t = q_F \) s.t. \( x_i \in L(\rho(r_{i-1}, r_i)) \).

Let \( (q_0 = s_0, \ldots, s_d = q_F) \) be the subsequence of states obtained by deleting all occurrences of \( q^* \).

For any run of \( q^* \) — i.e., an interval \([a, b]\) s.t. \( r_{a-1} \neq q^* = r_a = \ldots = r_{b-1} \neq r_b \) — let \( x_{[a,b]} = x_a \cdots x_b \).

If \( s_{j-1} = r_{a-1} \) and \( s_j = r_b \), then \( x_{[a,b]} \in L(\rho'(s_{j-1}, s_j)) \)

Let \( y_1, \ldots, y_d \) be the sequence of blocks of the form \( x_{[a,b]} \).

Then \( w = y_1 \cdots y_d \) and \( y_j \in L(\rho'(s_{j-1}, s_j)) \).
GNFA to Regex: From $k$ states to $k - 1$ states

$w \in L(G) \implies w \in L(G')$

Proof.

- $w \in L(G) \implies w = x_1 x_2 x_3 \cdots x_t$, and a sequence of states $q_0 = r_0, r_1, \ldots, r_t = q_F$ s.t. $x_i \in L(\rho(r_{i-1}, r_i))$.
- Let $(q_0 = s_0, \ldots, s_d = q_F)$ be the subsequence of states obtained by deleting all occurrences of $q^*$.
- For any run of $q^*$ — i.e., an interval $[a, b]$ s.t.
  $r_{a-1} \neq q^* = r_a = \ldots = r_{b-1} \neq r_b$ — let $x_{[a,b]} = x_a \cdots x_b$.
- If $s_{j-1} = r_{a-1}$ and $s_j = r_b$, then $x_{[a,b]} \in L(\rho'(s_{j-1}, s_j))$
- Let $y_1, \ldots, y_d$ be the sequence of blocks of the form $x_{[a,b]}$.
- Then $w = y_1 \cdots y_d$ and $y_j \in L(\rho'(s_{j-1}, s_j))$.

i.e., $w \in L(G) \implies w \in L(G')$.  

$\implies$
Proof (contd).

$w \in L(G') \implies w \in L(G)$

Case $y_j \in L(R_4)$. Retain the block $y_j$ and retain $s_j - 1$ and $s_j$ as adjacent states.

Case $y_j \in L(R_1 R_2^* R_3)$. $y_j = z_0 \cdots z_u + 1$ where $z_0 \in L(R_1)$, $z_1, \ldots, z_u \in L(R_2)$ and $z_{u+1} = L(R_3)$ (for some finite $u$). Insert $u + 1$ copies of $q^*$ between $s_j - 1$ and $s_j$. Divide $y_j$ into $u + 2$ blocks ($z_0, \ldots, z_{u+1}$). □

(See notes for a formal argument.)
GNFA to Regex: From $k$ states to $k - 1$ states

$w \in L(G') \implies w \in L(G)$

Proof (contd).

- $w \in L(G') \implies w = y_1 \cdots y_d$ and a sequence of states $q_0 = s_0, \ldots, s_d = q_F$ s.t. $y_j \in L(\rho'(s_{j-1}, s_j))$
GNFA to Regex: From $k$ states to $k - 1$ states

$w \in L(G') \implies w \in L(G)$

Proof (contd).

- $w \in L(G') \implies w = y_1 \cdots y_d$ and a sequence of states $q_0 = s_0, \ldots, s_d = q_F$ s.t. $y_j \in L(\rho'(s_{j-1}, s_j)) = L((\rho(s_{j-1}, q^*)\rho(q^*, q^*)^*\rho(q^*, r_i)) \cup \rho(s_{j-1}, s_j))$
GNFA to Regex: From $k$ states to $k - 1$ states

$w \in L(G') \implies w \in L(G)$

Proof (contd).

- $w \in L(G') \implies w = y_1 \cdots y_d$ and a sequence of states $q_0 = s_0, \ldots, s_d = q_F$ s.t. $y_j \in L(\rho'(s_{j-1}, s_j)) = L((\rho(s_{j-1}, q^*)\rho(q^*, q^*)^*\rho(q^*, r_i)) \cup \rho(s_{j-1}, s_j)) = L(R_1 R_2^* R_3) \cup L(R_4).$
GNFA to Regex: From $k$ states to $k - 1$ states

$w \in L(G') \implies w \in L(G)$

Proof (contd).

- $w \in L(G') \implies w = y_1 \cdots y_d$ and a sequence of states $q_0 = s_0, \ldots, s_d = q_F$ s.t. $y_j \in L(\rho'(s_{j-1}, s_j)) = L((\rho(s_{j-1}, q^*)\rho(q^*, q^*)^*\rho(q^*, r_i)) \cup \rho(s_{j-1}, s_j)) = L(R_1R_2^*R_3) \cup L(R_4)$.

- To build a sequence of blocks $x_1, \ldots, x_t$ and a sequence of states $q_0 = r_0, \ldots, r_t = q_F$ to show $w \in L(G)$:
GNFA to Regex: From $k$ states to $k - 1$ states

$w \in L(G') \implies w \in L(G)$

Proof (contd).

- $w \in L(G') \implies w = y_1 \cdots y_d$ and a sequence of states $q_0 = s_0, \ldots, s_d = q_F$ s.t. $y_j \in L(\rho'(s_{j-1}, s_j)) = L((\rho(s_{j-1}, q^*)\rho(q^*, q^*)^*\rho(q^*, r_i)) \cup \rho(s_{j-1}, s_j)) = L(R_1R_2^*R_3) \cup L(R_4)$.

- To build a sequence of blocks $x_1, \ldots, x_t$ and a sequence of states $q_0 = r_0, \ldots, r_t = q_F$ to show $w \in L(G)$:
  
  - Case $y_j \in L(R_4)$. Retain the block $y_j$ and retain $s_{j-1}$ and $s_j$ as adjacent states.
GNFA to Regex: From $k$ states to $k - 1$ states

$w \in L(G') \implies w \in L(G)$

Proof (contd).

- $w \in L(G') \implies w = y_1 \cdots y_d$ and a sequence of states
  $q_0 = s_0, \ldots, s_d = q_F$ s.t.
  $y_j \in L(\rho'(s_{j-1}, s_j)) = L((\rho(s_{j-1}, q^*)\rho(q^*, q^*)^{*}\rho(q^*, r_i)) \cup \rho(s_{j-1}, s_j)) = L(R_1 R_2^* R_3) \cup L(R_4)$.

- To build a sequence of blocks $x_1, \ldots, x_t$ and a sequence of states $q_0 = r_0, \ldots, r_t = q_F$ to show $w \in L(G)$:
  - Case $y_j \in L(R_4)$. Retain the block $y_j$ and retain $s_{j-1}$ and $s_j$ as adjacent states.
  - Case $y_j \in L(R_1 R_2^* R_3)$. $y_j = z_0 \cdots z_{u+1}$ where $z_0 \in L(R_1)$, $z_1, \ldots, z_u \in L(R_2)$ and $z_{u+1} = L(R_3)$ (for some finite $u$). Insert $u + 1$ copies of $q^*$ between $s_{j-1}$ and $s_j$. Divide $y_j$ into $u + 2$ blocks ($z_0, \ldots, z_{u+1}$). □
GNFA to Regex: From $k$ states to $k - 1$ states

$w \in L(G') \implies w \in L(G)$

Proof (contd).

- $w \in L(G') \implies w = y_1 \cdots y_d$ and a sequence of states $q_0 = s_0, \ldots, s_d = q_F$ s.t. $y_j \in L(\rho'(s_{j-1}, s_j)) = L((\rho(s_{j-1}, q^*) \rho(q^*, q^*)^* \rho(q^*, r_i)) \cup \rho(s_{j-1}, s_j)) = L(R_1 R_2^* R_3) \cup L(R_4)$.

- To build a sequence of blocks $x_1, \ldots, x_t$ and a sequence of states $q_0 = r_0, \ldots, r_t = q_F$ to show $w \in L(G)$:
  - **Case $y_j \in L(R_4)$**: Retain the block $y_j$ and retain $s_{j-1}$ and $s_j$ as adjacent states.
  - **Case $y_j \in L(R_1 R_2^* R_3)$**: $y_j = z_0 \cdots z_{u+1}$ where $z_0 \in L(R_1)$, $z_1, \ldots, z_u \in L(R_2)$ and $z_{u+1} = L(R_3)$ (for some finite $u$). Insert $u + 1$ copies of $q^*$ between $s_{j-1}$ and $s_j$. Divide $y_j$ into $u + 2$ blocks $(z_0, \ldots, z_{u+1})$. □

(See notes for a formal argument.)
DFA to Regex: Summary

Lemma

For every DFA $M$, there is a regular expression $R$ such that $L(M) = L(R)$.
Lemma

For every DFA $M$, there is a regular expression $R$ such that $L(M) = L(R)$.  

- Any DFA can be converted into an equivalent GNFA. So let $G$ be a GNFA s.t. $L(M) = L(G)$. 

Lemma

For every DFA $M$, there is a regular expression $R$ such that $L(M) = L(R)$.

- Any DFA can be converted into an equivalent GNFA. So let $G$ be a GNFA s.t. $L(M) = L(G)$.
- For any GNFA $G = (Q, \Sigma, q_0, q_F, \rho)$ with $|Q| > 2$, for any $q^* \in Q \setminus \{q_0, q_F\}$, $G$ and $\text{rip}(G, q^*)$ are equivalent.
DFA to Regex: Summary

**Lemma**

*For every DFA $M$, there is a regular expression $R$ such that $L(M) = L(R)$.***

- Any DFA can be converted into an equivalent GNFA. So let $G$ be a GNFA s.t. $L(M) = L(G)$.
- For any GNFA $G = (Q, \Sigma, q_0, q_F, \rho)$ with $|Q| > 2$, for any $q^* \in Q \setminus \{q_0, q_F\}$, $G$ and $\text{rip}(G, q^*)$ are equivalent. $\text{rip}(G, q^*)$ has one fewer state than $G$. 
DFA to Regex: Summary

Lemma
For every DFA $M$, there is a regular expression $R$ such that $L(M) = L(R)$.

- Any DFA can be converted into an equivalent GNFA. So let $G$ be a GNFA s.t. $L(M) = L(G)$.
- For any GNFA $G = (Q, \Sigma, q_0, q_F, \rho)$ with $|Q| > 2$, for any $q^* \in Q \setminus \{q_0, q_F\}$, $G$ and $\text{rip}(G, q^*)$ are equivalent. $\text{rip}(G, q^*)$ has one fewer state than $G$.
- So given $G$, by applying rip repeatedly (choosing $q^*$ arbitrarily each time), we can get a GNFA $G'$ with two states s.t. $L(G) = L(G')$. 
Lemma

For every DFA $M$, there is a regular expression $R$ such that $L(M) = L(R)$.

- Any DFA can be converted into an equivalent GNFA. So let $G$ be a GNFA s.t. $L(M) = L(G)$.
- For any GNFA $G = (Q, \Sigma, q_0, q_F, \rho)$ with $|Q| > 2$, for any $q^* \in Q \setminus \{q_0, q_F\}$, $G$ and $\text{rip}(G, q^*)$ are equivalent. $\text{rip}(G, q^*)$ has one fewer state than $G$.
- So given $G$, by applying $\text{rip}$ repeatedly (choosing $q^*$ arbitrarily each time), we can get a GNFA $G'$ with two states s.t. $L(G) = L(G')$. Formally, by induction on the number of states in $G$. 
Lemma

For every DFA $M$, there is a regular expression $R$ such that $L(M) = L(R)$.

- Any DFA can be converted into an equivalent GNFA. So let $G$ be a GNFA s.t. $L(M) = L(G)$.
- For any GNFA $G = (Q, \Sigma, q_0, q_F, \rho)$ with $|Q| > 2$, for any $q^* \in Q \setminus \{q_0, q_F\}$, $G$ and $\text{rip}(G, q^*)$ are equivalent. $\text{rip}(G, q^*)$ has one fewer state than $G$.
- So given $G$, by applying $\text{rip}$ repeatedly (choosing $q^*$ arbitrarily each time), we can get a GNFA $G'$ with two states s.t. $L(G) = L(G')$. Formally, by induction on the number of states in $G$.
- For a 2-state GNFA $G'$, $L(G') = L(R)$, where $R = \rho(q_0, q_F)$.
DFA to Regex: Example
DFA to Regex: Example

\begin{itemize}
\item $q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow q_F$
\item $q_0 \rightarrow q_1$ on $\epsilon$
\item $q_1 \rightarrow q_1$ on $0$
\item $q_1 \rightarrow q_2$ on $1$
\item $q_2 \rightarrow q_2$ on $\epsilon$
\item $q_2 \rightarrow q_1$ on $\epsilon$
\item $q_2 \rightarrow q_F$ on $\emptyset$
\item $q_F \rightarrow q_F$ on $\emptyset$
\end{itemize}
DFA to Regex: Example
DFA to Regex: Example
DFA to Regex: Example

$q_0 \xrightarrow{\epsilon 0^*1} q_2 \xrightarrow{\epsilon} q_F \\
0 \cup (10^*1)$
DFA to Regex: Example

\[ q_0 \xrightarrow{\varepsilon 0^* 1} q_2 \xrightarrow{\varepsilon} q_F \]

\[ 0 \cup (10^* 1) \]
DFA to Regex: Example

\[ q_0 \xrightarrow{0*1(0 \cup (10*1))^*} q_F \]
DFA to Regex: Example (2)
DFA to Regex: Example (2)
DFA to Regex: Example (2)
DFA to Regex: Example (2)

\[ a(aa \cup b)^* \]

\[ a(aa \cup b)^*ab \cup b \]

\[ (ba \cup a)(aa \cup b)^* \cup \varepsilon \]

\[ (ba \cup a)(aa \cup b)^*ab \cup bb \]
DFA to Regex: Example (2)

\[(a(aa\cup b)^*ab\cup b)((ba\cup a)(aa\cup b)^*ab\cup bb)^*((ba\cup a)(aa\cup b)^*\cup \varepsilon)\cup a(aa\cup b)^*\]