Formalizing DFAs

DFAs

A DFA, $M$, is a quintuple, $M = (Q, \Sigma, q_0, \delta, A)$, where

- $Q$ is a finite set of states,
- $\Sigma$ is a finite set of symbols (an alphabet),
- $q_0 \in Q$ is a special start state,
- $A \subseteq Q$ is the set of accepting states,
- $\delta : Q \times \Sigma \rightarrow Q$ is a transition function.
Formalizing DFAs

A DFA, $M$, is a quintuple, $M = (Q, \Sigma, q_0, \delta, A)$, where

- $Q$ is a finite set of states,
- $\Sigma$ is a finite set of symbols (an alphabet),
- $q_0 \in Q$ is a special start state,
- $A \subseteq Q$ is the set of accepting states,
- $\delta : Q \times \Sigma \rightarrow Q$ is a transition function.

$\delta$ maps a state and an input character to another state. Recall, $Q \times \Sigma$ is the set of all ordered pairs $(q, \sigma)$ such that $q \in Q$ and $\sigma \in \Sigma$. 
Formalizing DFAs

<table>
<thead>
<tr>
<th>DFAs</th>
</tr>
</thead>
<tbody>
<tr>
<td>A DFA, $M$, is a quintuple, $M = (Q, \Sigma, q_0, \delta, A)$, where</td>
</tr>
<tr>
<td>- $Q$ is a finite set of states,</td>
</tr>
<tr>
<td>- $\Sigma$ is a finite set of symbols (an alphabet),</td>
</tr>
<tr>
<td>- $q_0 \in Q$ is a special start state,</td>
</tr>
<tr>
<td>- $A \subseteq Q$ is the set of accepting states,</td>
</tr>
<tr>
<td>- $\delta : Q \times \Sigma \rightarrow Q$ is a transition function.</td>
</tr>
</tbody>
</table>

$\delta$ maps a state and an input character to another state.
Recall, $Q \times \Sigma$ is the set of all ordered pairs $(q, \sigma)$ such that $q \in Q$ and $\sigma \in \Sigma$.
$\delta(q, \sigma)$ denotes the state of the computation when you start in state $q$ and the next character is $\sigma$. 

Formalizing DFAs

A DFA, $M$, is a quintuple, $M = (Q, \Sigma, q_0, \delta, A)$, where

- $Q$ is a finite set of states,
- $\Sigma$ is a finite set of symbols (an alphabet),
- $q_0 \in Q$ is a special start state,
- $A \subseteq Q$ is the set of accepting states,
- $\delta : Q \times \Sigma \rightarrow Q$ is a transition function.

$\delta$ maps a state and an input character to another state. Recall, $Q \times \Sigma$ is the set of all ordered pairs $(q, \sigma)$ such that $q \in Q$ and $\sigma \in \Sigma$. $\delta(q, \sigma)$ denotes the state of the computation when you start in state $q$ and the next character is $\sigma$.

Note that $\delta$ is a total function: it is defined for every possible input pair. This assumes a trap state, and assures that the function table for $\delta$ doesn’t have any empty cells.
Example

\[ M = (Q, \Sigma, q_0, \delta, A) \] where \( q_0 \in Q \) is the start state, \( A \subseteq Q \) are the accepting states, \( \delta : Q \times \Sigma \to Q \).

\[ M = (\{q_0, q_1, q_2\}, \{a, b\}, q_0, \delta, \{q_1\}) \], where \( \delta \) is as follows:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td>q1</td>
<td>q2</td>
</tr>
<tr>
<td>q1</td>
<td>q1</td>
<td>q1</td>
</tr>
<tr>
<td>q2</td>
<td>q2</td>
<td>q2</td>
</tr>
</tbody>
</table>
Example

\[ M = (Q, \Sigma, q_0, \delta, A) \]

where \( q_0 \in Q \) is the start state, \( A \subseteq Q \) are the accepting states,
\( \delta : Q \times \Sigma \rightarrow Q \)

\[ M = (\{q_0, q_1, q_2\}, \{a, b\}, q_0, \delta, \{q_1\}) \]

where \( \delta \) is as follows:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td>q1</td>
<td>q2</td>
</tr>
<tr>
<td>q1</td>
<td>q1</td>
<td>q1</td>
</tr>
<tr>
<td>q2</td>
<td>q2</td>
<td>q2</td>
</tr>
</tbody>
</table>

[Diagram of state transitions]
Example

\[ M = (Q, \Sigma, q_0, \delta, A) \] where \( q_0 \in Q \) is the start state, \( A \subseteq Q \) are the accepting states, \( \delta : Q \times \Sigma \rightarrow Q \).

\[ M = (\{q_0, q_1, q_2\}, \{a, b\}, q_0, \delta, \{q_1\}) \], where \( \delta \) is as follows:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td>q1</td>
<td>q2</td>
</tr>
<tr>
<td>q1</td>
<td>q1</td>
<td>q1</td>
</tr>
<tr>
<td>q2</td>
<td>q2</td>
<td>q2</td>
</tr>
</tbody>
</table>

\[ \delta \]

- \( q_0 \rightarrow q_1 \rightarrow q_1 \rightarrow q_1 \)
- \( q_0 \rightarrow q_0 \rightarrow q_0 \rightarrow q_1 \)
Example

\[ M = (Q, \Sigma, q_0, \delta, A) \] where \( q_0 \in Q \) is the start state, \( A \subseteq Q \) are the accepting states, \( \delta : Q \times \Sigma \rightarrow Q \)

\[ M = (\{q_0, q_1, q_2\}, \{a, b\}, q_0, \delta, \{q_1\}) \] where \( \delta \) is as follows:

\[
\begin{array}{c|cc}
  & a & b \\
\hline
q_0 & q_1 & q_2 \\
q_1 & q_1 & q_1 \\
q_2 & q_2 & q_2 \\
\end{array}
\]
Example

\[ M = (Q, \Sigma, q_0, \delta, A) \] where \( q_0 \in Q \) is the start state, \( A \subseteq Q \) are the accepting states, \( \delta : Q \times \Sigma \to Q \)

\[ M = (\{q_0, q_1, q_2\}, \{a, b\}, q_0, \delta, \{q_1\}) \], where \( \delta \) is as follows:

\[
\begin{array}{ccc}
q_0 & q_1 & q_2 \\
q_0 & q_1 & q_2 \\
q_2 & q_2 & q_2 \\
\end{array}
\]
Example

\[ M = (Q, \Sigma, q_0, \delta, A) \] where \( q_0 \in Q \) is the start state, \( A \subseteq Q \) are the accepting states, \( \delta : Q \times \Sigma \to Q \)

\[ M = (\{q_0, q_1, q_2\}, \{a, b\}, q_0, \delta, \{q_1\}) \], where \( \delta \) is as follows:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td>q1</td>
<td>q2</td>
</tr>
<tr>
<td>q1</td>
<td>q1</td>
<td>q1</td>
</tr>
<tr>
<td>q2</td>
<td>q2</td>
<td>q2</td>
</tr>
</tbody>
</table>
Example

\( M = (Q, \Sigma, q_0, \delta, A) \) where \( q_0 \in Q \) is the start state, \( A \subseteq Q \) are the accepting states, \( \delta : Q \times \Sigma \rightarrow Q \)

\( M = (\{q_0, q_1, q_2\}, \{a, b\}, q_0, \delta, \{q_1\}) \), where \( \delta \) is as follows:

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>q0</td>
<td>q1</td>
<td>q2</td>
</tr>
<tr>
<td>q1</td>
<td>q1</td>
<td>q1</td>
</tr>
<tr>
<td>q2</td>
<td>q2</td>
<td>q2</td>
</tr>
</tbody>
</table>

![Diagram of the automaton](image)
closure of $\delta$

For a given DFA, $M = (Q, \Sigma, q_0, \delta, A)$, $\delta^*$ is a function that takes a state and a string as input, and produces a resulting state. That is, $\delta^* : Q \times \Sigma^* \rightarrow Q$, and

1. For any $q \in Q$, $\delta^*(q, \Lambda) = q$,
2. For any $q \in Q$, any $\sigma \in \Sigma$, and any $x \in \Sigma^*$, $\delta^*(q, x\sigma) = \delta(\delta^*(q, x), \sigma)$
Lemma 9.2

If \( L = \mathcal{L}(G) \) for some deterministic regular grammar \( G \), then there exists a DFA \( M \) such that \( L = \mathcal{L}(M) \).

Let \( G = (V, \Sigma, S, P) \). We define DFA \( M = (Q, \Sigma, q_0, \delta, A) \) as follows.

\[ Q = V, \quad q_0 = S, \quad A = \{ p \mid p \rightarrow \Lambda \in P \}, \]

and, for every \( p \in Q, \sigma \in \Sigma \),
\[ \delta(p, \sigma) = q \text{ if } p \rightarrow \sigma q \in P. \]

If \( \delta \) is not a complete function, we can add a trap state to \( Q \).

Claim: For every \( x \in \Sigma^* \) and \( A \in V \), \( S^* \Rightarrow xA \) iff \( \delta^*(q_0, x) = A \).

Proof is by induction on the length of \( x \). When \( |x| = 0 \), note that \( S \) can generate \( A \) (because \( G \) is regular), and \( \delta(q_0, \Lambda) = S \).

Suppose the claim holds for all \( x \) of length \( k \geq 0 \).

\[ (S^* \Rightarrow xA) \rightarrow (\delta^*(q_0, x) = A) \]

Suppose that in the \( k+1 \)st step of a derivation of string \( yb \) of length \( k+1 \), we use the rule \( A \rightarrow bC \): we have \( S^* \Rightarrow yA \Rightarrow ybC \).

By the inductive hypothesis, we have \( \delta^*(q_0, y) = A \).

By the way we constructed \( M \), we have \( \delta(A, b) = C \).

Therefore:
\[ \delta^*(q_0, yb) = C. \]
Lemma 9.2

If $L = L(G)$ for some deterministic regular grammar $G$, then there exists a DFA $M$ such that $L = L(M)$.

Let $G = (V, \Sigma, S, P)$. We define DFA $M = (Q, \Sigma, q_0, \delta, A)$ as follows.
Lemma 9.2

If \( L = \mathcal{L}(G) \) for some deterministic regular grammar \( G \), then there exists a DFA \( M \) such that \( L = \mathcal{L}(M) \).

Let \( G = (V, \Sigma, S, P) \). We define DFA \( M = (Q, \Sigma, q_0, \delta, A) \) as follows.

\( Q = V, \)
\( q_0 = S, \)
\( A = \{ p \mid p \to \Lambda \in P \}, \)
\( \delta \).
Lemma 9.2

If $L = L(G)$ for some deterministic regular grammar $G$, then there exists a DFA $M$ such that $L = L(M)$.

Let $G = (V, \Sigma, S, P)$. We define DFA $M = (Q, \Sigma, q_0, \delta, A)$ as follows.

$Q = V$,
$q_0 = S$,
$A = \{p \mid p \rightarrow \Lambda \in P\}$, and,
for every $p \in Q, \sigma \in \Sigma$, $\delta(p, \sigma) = q$ if $p \rightarrow \sigma q \in P$. 
Lemma 9.2

If \( L = \mathcal{L}(G) \) for some deterministic regular grammar \( G \), then there exists a DFA \( M \) such that \( L = \mathcal{L}(M) \).

Let \( G = (V, \Sigma, S, P) \). We define DFA \( M = (Q, \Sigma, q_0, \delta, A) \) as follows.

\( Q = V \),
\( q_0 = S \),
\( A = \{ p \mid p \rightarrow \Lambda \in P \} \), and,
for every \( p \in Q, \sigma \in \Sigma, \delta(p, \sigma) = q \) if \( p \rightarrow \sigma q \in P \).
If \( \delta \) is not a complete function, we can add a trap state to \( Q \).
Lemma 9.2

If \( L = \mathcal{L}(G) \) for some deterministic regular grammar \( G \), then there exists a DFA \( M \) such that \( L = \mathcal{L}(M) \).

Let \( G = (V, \Sigma, S, P) \). We define DFA \( M = (Q, \Sigma, q_0, \delta, A) \) as follows.

\( Q = V \),
\( q_0 = S \),
\( A = \{ p \mid p \rightarrow \Lambda \in P \} \), and,
for every \( p \in Q, \sigma \in \Sigma, \delta(p, \sigma) = q \) if \( p \rightarrow \sigma q \in P \).

If \( \delta \) is not a complete function, we can add a trap state to \( Q \).

Claim: For every \( x \in \Sigma^* \) and \( A \in V \), \( S \xrightarrow{*} xA \) iff \( \delta^*(q_0, x) = A \).
Lemma 9.2

If $L = L(G)$ for some deterministic regular grammar $G$, then there exists a DFA $M$ such that $L = L(M)$.

Let $G = (V, \Sigma, S, P)$. We define DFA $M = (Q, \Sigma, q_0, \delta, A)$ as follows.

$Q = V$,  
$q_0 = S$,  
$A = \{p \mid p \rightarrow \Lambda \in P\}$, and,  
for every $p \in Q, \sigma \in \Sigma$, $\delta(p, \sigma) = q$ if $p \rightarrow \sigma q \in P$.

If $\delta$ is not a complete function, we can add a trap state to $Q$.

Claim: For every $x \in \Sigma^*$ and $A \in V$, $S \Rightarrow^* xA$ iff $\delta^*(q_0, x) = A$.

Proof is by induction on the length of $x$. When $|x| = 0$, note that $S$ can generate $A$ iff $A = S$ (because $G$ is regular), and $\delta(q_0, \Lambda) = S$. 


Lemma 9.2

If $L = \mathcal{L}(G)$ for some deterministic regular grammar $G$, then there exists a DFA $M$ such that $L = \mathcal{L}(M)$.

Let $G = (V, \Sigma, S, P)$. We define DFA $M = (Q, \Sigma, q_0, \delta, A)$ as follows.

- $Q = V$,
- $q_0 = S$,
- $A = \{p \mid p \rightarrow \Lambda \in P\}$, and,
- for every $p \in Q$, $\sigma \in \Sigma$, $\delta(p, \sigma) = q$ if $p \rightarrow \sigma q \in P$.

If $\delta$ is not a complete function, we can add a trap state to $Q$.

Claim: For every $x \in \Sigma^*$ and $A \in V$, $S \overset{*}{\Rightarrow} xA$ iff $\delta^*(q_0, x) = A$.

Proof is by induction on the length of $x$. When $|x| = 0$, note that $S$ can generate $A$ iff $A = S$ (because $G$ is regular), and $\delta(q_0, \Lambda) = S$.

Suppose the claim holds for all $x$ of length $k \geq 0$. 

. 
Lemma 9.2

If \( L = \mathcal{L}(G) \) for some deterministic regular grammar \( G \), then there exists a DFA \( M \) such that \( L = \mathcal{L}(M) \).

Let \( G = (V, \Sigma, S, P) \). We define DFA \( M = (Q, \Sigma, q_0, \delta, A) \) as follows.

\[
Q = V, \\
q_0 = S, \\
A = \{ p \mid p \rightarrow \Lambda \in P \}, \text{ and,} \\
\text{for every } p \in Q, \sigma \in \Sigma, \, \delta(p, \sigma) = q \text{ if } p \rightarrow \sigma q \in P.
\]

If \( \delta \) is not a complete function, we can add a trap state to \( Q \).

Claim: For every \( x \in \Sigma^* \) and \( A \in V \), \( S \Rightarrow^* xA \) iff \( \delta^*(q_0, x) = A \).

Proof is by induction on the length of \( x \). When \( |x| = 0 \), note that \( S \) can generate \( A \) iff \( A = S \) (because \( G \) is regular), and \( \delta(q_0, \Lambda) = S \).

Suppose the claim holds for all \( x \) of length \( k \geq 0 \).

\[
(S \Rightarrow^* xA) \rightarrow (\delta^*(q_0, x) = A)
\]

Suppose that in the \( k + 1 \)st step of a derivation of string \( yb \) of length \( k + 1 \), we use the rule \( A \rightarrow bC \):

we have \( S \Rightarrow^* yA \Rightarrow ybC \).
Lemma 9.2

If \( L = L(G) \) for some deterministic regular grammar \( G \), then there exists a DFA \( M \) such that \( L = L(M) \).

Let \( G = (V, \Sigma, S, P) \). We define DFA \( M = (Q, \Sigma, q_0, \delta, A) \) as follows.

\[ Q = V, \]
\[ q_0 = S, \]
\[ A = \{ p \mid p \rightarrow \Lambda \in P \}, \]

and, for every \( p \in Q, \sigma \in \Sigma, \delta(p, \sigma) = q \) if \( p \rightarrow \sigma q \in P \).

If \( \delta \) is not a complete function, we can add a trap state to \( Q \).

Claim: For every \( x \in \Sigma^* \) and \( A \in V \), \( S \Rightarrow^*xA \) iff \( \delta^*(q_0, x) = A \).

Proof is by induction on the length of \( x \). When \( |x| = 0 \), note that \( S \) can generate \( A \) iff \( A = S \) (because \( G \) is regular), and \( \delta(q_0, \Lambda) = S \).

Suppose the claim holds for all \( x \) of length \( k \geq 0 \).

\[ (S \Rightarrow^*xA) \rightarrow (\delta^*(q_0, x) = A) \]

Suppose that in the \( k+1 \)st step of a derivation of string \( yb \) of length \( k + 1 \), we use the rule \( A \rightarrow bC \):

we have \( S \Rightarrow^*yA \Rightarrow^*ybC \).

By the inductive hypothesis, we have \( \delta^*(q_0, y) = A \).
Lemma 9.2

If $L = \mathcal{L}(G)$ for some deterministic regular grammar $G$, then there exists a DFA $M$ such that $L = \mathcal{L}(M)$.

Let $G = (V, \Sigma, S, P)$. We define DFA $M = (Q, \Sigma, q_0, \delta, A)$ as follows.

$Q = V, \quad q_0 = S, \quad A = \{p \mid p \rightarrow \Lambda \in P\}$, and,

for every $p \in Q, \sigma \in \Sigma$, $\delta(p, \sigma) = q$ if $p \rightarrow \sigma q \in P$.

If $\delta$ is not a complete function, we can add a trap state to $Q$.

Claim: For every $x \in \Sigma^*$ and $A \in V$, $S \Rightarrow^* xA$ iff $\delta^*(q_0, x) = A$.

Proof is by induction on the length of $x$. When $|x| = 0$, note that $S$ can generate $A$ iff $A = S$ (because $G$ is regular), and $\delta(q_0, \Lambda) = S$.

Suppose the claim holds for all $x$ of length $k \geq 0$.

$$(S \Rightarrow^* xA) \rightarrow (\delta^*(q_0, x) = A)$$

Suppose that in the $k + 1$st step of a derivation of string $yb$ of length $k + 1$, we use the rule $A \rightarrow bC$:

we have $S \Rightarrow^* yA \Rightarrow ybC$.

By the inductive hypothesis, we have $\delta^*(q_0, y) = A$.

By the way we constructed $M$, we have $\delta(A, b) = C$.

Therefore: $\delta^*(q_0, yb) = C$. 

\[\text{DFA from RG}\]
Lemma 9.2

If \( L = \mathcal{L}(G) \) for some deterministic regular grammar \( G \), then there exists a DFA \( M \) such that \( L = \mathcal{L}(M) \).

Let \( G = (V, \Sigma, S, P) \). We define DFA \( M = (Q, \Sigma, q_0, \delta, A) \) as follows.

- \( Q = V \),
- \( q_0 = S \),
- \( A = \{ p \mid p \rightarrow \Lambda \in P \} \), and,
- for every \( p \in Q, \sigma \in \Sigma, \delta(p, \sigma) = q \) if \( p \rightarrow \sigma q \in P \).

If \( \delta \) is not a complete function, we can add a trap state to \( Q \).

Claim: For every \( x \in \Sigma^* \) and \( A \in V \), \( S \overset{*}{\Rightarrow} xA \) iff \( \delta^*(q_0, x) = A \).

Proof is by induction on the length of \( x \). When \( |x| = 0 \), note that \( S \) can generate \( A \) iff \( A = S \) (because \( G \) is regular), and \( \delta(q_0, \Lambda) = S \).

Suppose the claim holds for all \( x \) of length \( k \geq 0 \).

\((\delta^*(q_0, x) = A) \rightarrow (S \overset{*}{\Rightarrow} xA)\)
Lemma 9.2

If \( L = \mathcal{L}(G) \) for some deterministic regular grammar \( G \), then there exists a DFA \( M \) such that \( L = \mathcal{L}(M) \).

Let \( G = (V, \Sigma, S, P) \). We define DFA \( M = (Q, \Sigma, q_0, \delta, A) \) as follows.

- \( Q = V \),
- \( q_0 = S \),
- \( A = \{ p \mid p \rightarrow \Lambda \in P \} \), and,
- for every \( p \in Q, \sigma \in \Sigma, \delta(p, \sigma) = q \) if \( p \rightarrow \sigma q \in P \).

If \( \delta \) is not a complete function, we can add a trap state to \( Q \).

Claim: For every \( x \in \Sigma^* \) and \( A \in V \), \( S \overset{x}{\Rightarrow} xA \) iff \( \delta^*(q_0, x) = A \).

Proof is by induction on the length of \( x \). When \( |x| = 0 \), note that \( S \) can generate \( A \) iff \( A = S \) (because \( G \) is regular), and \( \delta(q_0, \Lambda) = S \).

Suppose the claim holds for all \( x \) of length \( k \geq 0 \).

\[(\delta^*(q_0, x) = A) \rightarrow (S \overset{x}{\Rightarrow} xA)\]

Suppose that on some input \( yb \) of length \( k + 1 \), the \( k + 1 \)st transition in \( M \) was \( \delta(A, b) = C \).
Lemma 9.2

If $L = L(G)$ for some deterministic regular grammar $G$, then there exists a DFA $M$ such that $L = L(M)$.

Let $G = (V, \Sigma, S, P)$. We define DFA $M = (Q, \Sigma, q_0, \delta, A)$ as follows.

$Q = V$,
$q_0 = S$,
$A = \{ p \mid p \rightarrow \Lambda \in P \}$, and,
for every $p \in Q, \sigma \in \Sigma$, $\delta(p, \sigma) = q$ if $p \rightarrow \sigma q \in P$.

If $\delta$ is not a complete function, we can add a trap state to $Q$.

Claim: For every $x \in \Sigma^*$ and $A \in V$, $S \xrightarrow{*} xA$ iff $\delta^*(q_0, x) = A$.

Proof is by induction on the length of $x$. When $|x| = 0$, note that $S$ can generate $A$ iff $A = S$ (because $G$ is regular), and $\delta(q_0, \Lambda) = S$.

Suppose the claim holds for all $x$ of length $k \geq 0$.

$(\delta^*(q_0, x) = A) \rightarrow (S \xrightarrow{*} xA)$

Suppose that on some input $yb$ of length $k + 1$, the $k + 1$st transition in $M$ was $\delta(A, b) = A$.

We have: $\delta^*(q_0, y) = A$. 
Lemma 9.2

If \( L = \mathcal{L}(G) \) for some deterministic regular grammar \( G \), then there exists a DFA \( M \) such that \( L = \mathcal{L}(M) \).

Let \( G = (V, \Sigma, S, P) \). We define DFA \( M = (Q, \Sigma, q_0, \delta, A) \) as follows.

- \( Q = V \),
- \( q_0 = S \),
- \( A = \{ p \mid p \rightarrow \Lambda \in P \} \), and,
- for every \( p \in Q, \sigma \in \Sigma, \delta(p, \sigma) = q \) if \( p \rightarrow \sigma q \in P \).

If \( \delta \) is not a complete function, we can add a trap state to \( Q \).

Claim: For every \( x \in \Sigma^* \) and \( A \in V \), \( S \Rightarrow^* xA \) iff \( \delta^*(q_0, x) = A \).

Proof is by induction on the length of \( x \). When \( |x| = 0 \), note that \( S \) can generate \( A \) iff \( A = S \) (because \( G \) is regular), and \( \delta(q_0, \Lambda) = S \).

Suppose the claim holds for all \( x \) of length \( k \geq 0 \).

\((\delta^*(q_0, x) = A) \rightarrow (S \Rightarrow^* xA)\)

Suppose that on some input \( yb \) of length \( k + 1 \), the \( k + 1 \)st transition in \( M \) was \( \delta(A, b) = C \).

We have: \( \delta^*(q_0, y) = A \).

By the inductive hypothesis, we have \( S \Rightarrow^* yA \).
Lemma 9.2

If \( L = \mathcal{L}(G) \) for some deterministic regular grammar \( G \), then there exists a DFA \( M \) such that \( L = \mathcal{L}(M) \).

Let \( G = (V, \Sigma, S, P) \). We define DFA \( M = (Q, \Sigma, q_0, \delta, A) \) as follows.

- \( Q = V \),
- \( q_0 = S \),
- \( A = \{ p \mid p \rightarrow \Lambda \in P \} \), and,
- for every \( p \in Q, \sigma \in \Sigma, \delta(p, \sigma) = q \) if \( p \rightarrow \sigma q \in P \).

If \( \delta \) is not a complete function, we can add a trap state to \( Q \).

Claim: For every \( x \in \Sigma^* \) and \( A \in V \), \( S \xrightarrow{*} xA \) iff \( \delta^*(q_0, x) = A \).

Proof is by induction on the length of \( x \). When \(|x| = 0\), note that \( S \) can generate \( A \) iff \( A = S \) (because \( G \) is regular), and \( \delta(q_0, \Lambda) = S \).

Suppose the claim holds for all \( x \) of length \( k \geq 0 \).

\((\delta^*(q_0, x) = A) \rightarrow (S \xrightarrow{*} xA)\)

Suppose that on some input \( yb \) of length \( k + 1 \), the \( k + 1 \)st transition in \( M \) was \( \delta(A, b) = C \).

We have: \( \delta^*(q_0, y) = A \).

By the inductive hypothesis, we have \( S \xrightarrow{*} yA \).

By the way we defined \( M \), there must be a rule \( A \rightarrow bC \).
Lemma 9.2

If \( L = \mathcal{L}(G) \) for some deterministic regular grammar \( G \), then there exists a DFA \( M \) such that \( L = \mathcal{L}(M) \).

Let \( G = (V, \Sigma, S, P) \). We define DFA \( M = (Q, \Sigma, q_0, \delta, A) \) as follows.
\[
Q = V, \\
q_0 = S, \\
A = \{p \mid p \rightarrow \Lambda \in P\}, \\
\text{and,}
\]
for every \( p \in Q, \sigma \in \Sigma, \delta(p, \sigma) = q \) if \( p \rightarrow \sigma q \in P \).
If \( \delta \) is not a complete function, we can add a trap state to \( Q \).

Claim: For every \( x \in \Sigma^* \) and \( A \in V \), \( S \xrightarrow{*} xA \) iff \( \delta^*(q_0, x) = A \).
Proof is by induction on the length of \( x \). When \( |x| = 0 \), note that \( S \) can generate \( A \) iff \( A = S \) (because \( G \) is regular), and \( \delta(q_0, \Lambda) = S \).

Suppose the claim holds for all \( x \) of length \( k \geq 0 \).
\[(\delta^*(q_0, x) = A) \rightarrow (S \xrightarrow{*} xA)\]
Suppose that on some input \( yb \) of length \( k + 1 \), the \( k + 1 \)st transition in \( M \) was \( \delta(A, b) = C \).
We have: \( \delta^*(q_0, y) = A \).
By the inductive hypothesis, we have \( S \xrightarrow{*} yA \)
By the way we defined \( M \), there must be a rule \( A \rightarrow bC \).
Therefore, \( S \xrightarrow{*} ybC \).
Lemma 9.2

If \( L = \mathcal{L}(G) \) for some deterministic regular grammar \( G \), then there exists a DFA \( M \) such that \( L = \mathcal{L}(M) \).

Let \( G = (V, \Sigma, S, P) \). We define DFA \( M = (Q, \Sigma, q_0, \delta, A) \) as follows.

\[ Q = V, \]
\[ q_0 = S, \]
\[ A = \{ p \mid p \rightarrow \Lambda \in P \}, \]
and,

for every \( p \in Q, \sigma \in \Sigma, \delta(p, \sigma) = q \) if \( p \rightarrow \sigma q \in P \).

If \( \delta \) is not a complete function, we can add a trap state to \( Q \).

Claim: For every \( x \in \Sigma^* \) and \( A \in V, S \xrightarrow{*} xA \) iff \( \delta^*(q_0, x) = A \).

Proof is by induction on the length of \( x \). When \( |x| = 0 \), note that \( S \) can generate \( A \) iff \( A = S \) (because \( G \) is regular), and \( \delta(q_0, \Lambda) = S \).

Suppose the claim holds for all \( x \) of length \( k \geq 0 \).

\[ (\delta^*(q_0, x) = A) \rightarrow (S \xrightarrow{*} xA) \]

Suppose that on some input \( yb \) of length \( k + 1 \), the \( k + 1 \)st transition in \( M \) was \( \delta(A, b) = C \).

We have: \( \delta^*(q_0, y) = A \).

By the inductive hypothesis, we have \( S \xrightarrow{*} yA \)

By the way we defined \( M \), there must be a rule \( A \rightarrow bC \).

Therefore, \( S \xrightarrow{*} ybC \).

(Why does this suffice for the proof of the Lemma?)
Example: DFA from $G$

Consider the grammar $G = (\{C, D, E, F\}, \{a, b\}, C, P)$, where

$P = \{C \rightarrow aD; D \rightarrow aC; E \rightarrow aF; F \rightarrow aE; C \rightarrow bE; D \rightarrow bF; E \rightarrow bC; F \rightarrow bD; F \rightarrow \Lambda\}$
Example: DFA from $G$

Consider the grammar $G = (\{C, D, E, F\}, \{a, b\}, C, P)$, where

$P = \{C \rightarrow aD; D \rightarrow aC; E \rightarrow aF; F \rightarrow aE; C \rightarrow bE; D \rightarrow bF; E \rightarrow bC; F \rightarrow bD; F \rightarrow \Lambda\}$
Example: DFA from $G$

Consider the grammar $G = (\{C, D, E, F\}, \{a, b\}, C, P)$, where

$P = \{C \rightarrow aD; D \rightarrow aC; E \rightarrow aF; F \rightarrow aE; C \rightarrow bE; D \rightarrow bF; E \rightarrow bC; F \rightarrow bD; F \rightarrow \Lambda\}$
Example: DFA from $G$

Consider the grammar $G = (\{C, D, E, F\}, \{a, b\}, C, P)$, where

$P = \{C \rightarrow aD; D \rightarrow aC; E \rightarrow aF; F \rightarrow aE; C \rightarrow bE; D \rightarrow bF; E \rightarrow bC; F \rightarrow bD; F \rightarrow \Lambda\}$
Example: DFA from $G$

Consider the grammar $G = (\{C, D, E, F\}, \{a, b\}, C, P)$, where

$P = \{C \rightarrow aD; D \rightarrow aC; E \rightarrow aF; F \rightarrow aE; C \rightarrow bE; D \rightarrow bF; E \rightarrow bC; F \rightarrow bD; F \rightarrow \Lambda\}$
Example: DFA from $G$

Consider the grammar $G = (\{C, D, E, F\}, \{a, b\}, C, P)$, where

$P = \{C \rightarrow aD; D \rightarrow aC; E \rightarrow aF; F \rightarrow aE; C \rightarrow bE; D \rightarrow bF; E \rightarrow bC; F \rightarrow bD; F \rightarrow \Lambda\}$
Example: DFA from $G$

Consider the grammar $G = (\{C, D, E, F\}, \{a, b\}, C, P)$, where

$P = \{C \rightarrow aD; D \rightarrow aC; E \rightarrow aF; F \rightarrow aE; C \rightarrow bE; D \rightarrow bF; E \rightarrow bC; F \rightarrow bD; F \rightarrow \Lambda\}$
Example: DFA from $G$

Consider the grammar $G = (\{C, D, E, F\}, \{a, b\}, C, P)$, where

$P = \{C \rightarrow aD; D \rightarrow aC; E \rightarrow aF; F \rightarrow aE; C \rightarrow bE; D \rightarrow bF; E \rightarrow bC; F \rightarrow bD; F \rightarrow \Lambda\}$
Example: DFA from $G$

Consider the grammar $G = (\{C, D, E, F\}, \{a, b\}, C, P)$, where $P = \{C \to aD; D \to aC; E \to aF; F \to aE; C \to bE; D \to bF; E \to bC; F \to bD; F \to \Lambda\}$

For example: consider $\delta^*(C, aab) = E$, and $C \Rightarrow aD \Rightarrow aaC \Rightarrow aabE$. 
Lemma 9.3

If $L = \mathcal{L}(M)$ for some deterministic finite automata $M$, then there exists a deterministic regular grammar $G$ such that $L = \mathcal{L}(G)$. 

The same algorithm works in reverse!

Theorem 9.1

$L$ is a regular language if and only if there exists a deterministic finite automata $M$ such that $L = \mathcal{L}(M)$.
Lemma 9.3

If $L = L(M)$ for some deterministic finite automata $M$, then there exists a deterministic regular grammar $G$ such that $L = L(G)$.

The same algorithm works in reverse!
Lemma 9.3

If $L = \mathcal{L}(M)$ for some deterministic finite automata $M$, then there exists a deterministic regular grammar $G$ such that $L = \mathcal{L}(G)$.

The same algorithm works in reverse!

Theorem 9.1

$L$ is a regular language if and only if there exists a deterministic finite automata $M$ such that $L = \mathcal{L}(M)$. 