Reverse of a Regular Language
Theorem:

The reverse $L^R$ of a regular language $L$ is a regular language.

Proof idea:

Construct NFA that accepts $L^R$:

invert the transitions of the NFA that accepts $L$
Proof

Since $L$ is regular, there is NFA that accepts $L$

Example:

$L = ab^* + ba$
Invert Transitions
Make old initial state a final state
Add a new initial state
Resulting machine accepts $L^R$

$L^R$ is regular

$L = ab^* + ba$

$L^R = b^* a + ab$
Linear Grammars
Linear Grammars

Grammars with at most one variable at the right side of a production

Examples:

\[ S \rightarrow aSb \] \[ S \rightarrow Ab \] 
\[ S \rightarrow \lambda \] \[ A \rightarrow aAb \] 
\[ A \rightarrow \lambda \]
A Non-Linear Grammar

Grammar \( G : \)

\[ S \rightarrow SS \]

\[ S \rightarrow \lambda \]

\[ S \rightarrow aSb \]

\[ S \rightarrow bSa \]

\[ L(G) = \{ w : n_a(w) = n_b(w) \} \]
Another Linear Grammar

Grammar \( G : \)

\[ S \rightarrow A \]

\[ A \rightarrow aB \mid \lambda \]

\[ B \rightarrow Ab \]

\[ L(G) = \{ a^n b^n : n \geq 0 \} \]
Right-Linear Grammars

All productions have form: \[ A \rightarrow xB \]

or

\[ A \rightarrow x \]

Example:

\[ S \rightarrow abS \]

\[ S \rightarrow a \]
Left-Linear Grammars

All productions have form:  \[ A \rightarrow Bx \]

or

\[ A \rightarrow x \]

Example:

\[ S \rightarrow Aab \]

\[ A \rightarrow Aab \mid B \]

\[ B \rightarrow a \]
Regular Grammars
Regular Grammars

A regular grammar is any right-linear or left-linear grammar

Examples:

\[ G_1 \]

\[ S \rightarrow abS \]

\[ S \rightarrow a \]

\[ G_2 \]

\[ S \rightarrow Aab \]

\[ A \rightarrow Aab | B \]

\[ B \rightarrow a \]
Observation

Regular grammars generate regular languages

Examples:

\[ G_1 \]
\[ S \rightarrow abS \]
\[ S \rightarrow a \]

\[ L(G_1) = (ab)^*a \]

\[ G_2 \]
\[ S \rightarrow Aab \]
\[ A \rightarrow Aab \mid B \]
\[ B \rightarrow a \]

\[ L(G_2) = aab(ab)^* \]
Regular Grammars
Generate
Regular Languages
Theorem

\[
\{ \text{Languages Generated by Regular Grammars} \} = \{ \text{Regular Languages} \}
\]
Theorem - Part 1

\[
\left\{ \text{Languages Generated by Regular Grammars} \right\} \subseteq \left\{ \text{Regular Languages} \right\}
\]

Any regular grammar generates a regular language
Theorem - Part 2

\[
\left\{ \begin{array}{c}
\text{Languages} \\
\text{Generated by} \\
\text{Regular Grammars}
\end{array} \right\} \supseteq \left\{ \begin{array}{c}
\text{Regular} \\
\text{Languages}
\end{array} \right\}
\]

Any regular language is generated by a regular grammar
Proof - Part 1

\{ Languages Generated by Regular Grammars \} \subseteq \{ Regular Languages \}

The language $L(G)$ generated by any regular grammar $G$ is regular.
The case of Right-Linear Grammars

Let $G$ be a right-linear grammar

We will prove: $L(G)$ is regular

Proof idea: We will construct NFA $M$ with $L(M) = L(G)$
Grammar $G$ is right-linear

Example:

\[ S \rightarrow aA \mid B \]
\[ A \rightarrow aa \ B \]
\[ B \rightarrow b \ B \mid a \]
Construct NFA $M$ such that every state is a grammar variable:

$$S \rightarrow aA \mid B$$

$$A \rightarrow aa \ B$$

$$B \rightarrow b \ B \mid a$$

special final state
Add edges for each production:

$S \rightarrow aA$
$S \rightarrow aA \mid B$
\[ S \rightarrow aA \mid B \]
\[ A \rightarrow a\alpha \ B \]
\[
S \rightarrow aA \mid B \\
A \rightarrow aa \ B \\
B \rightarrow bB
\]
\[ S \rightarrow aA \mid B \]
\[ A \rightarrow aa \ B \]
\[ B \rightarrow bB \mid a \]
\[ S \Rightarrow aA \Rightarrow aaaB \Rightarrow aaabB \Rightarrow aaaba \]
Grammar

\[ \begin{align*}
S & \rightarrow aA \mid B \\
A & \rightarrow aa \ B \\
B & \rightarrow bB \mid a
\end{align*} \]

\[ L(M) = L(G) = aaab^*a + b^*a \]
In General

A right-linear grammar \( G \)

has variables: \( V_0, V_1, V_2, \ldots \)

and productions: \( V_i \rightarrow a_1 a_2 \cdots a_m V_j \)

or

\( V_i \Rightarrow a_1 a_2 \cdots a_m \)
We construct the NFA $M$ such that:

each variable $V_i$ corresponds to a node:

$V_0$ -> $V_1$ -> $V_2$ -> $V_3$ -> $V_4$ -> $V_F$

special final state
For each production: \[ V_i \rightarrow a_1 a_2 \cdots a_m V_j \]
we add transitions and intermediate nodes
For each production: \[ V_i \rightarrow a_1a_2 \cdots a_m \]

we add transitions and intermediate nodes
Resulting NFA $M$ looks like this:

It holds that: $L(G) = L(M)$
The case of Left-Linear Grammars

Let $G$ be a left-linear grammar

We will prove: $L(G)$ is regular

Proof idea:
We will construct a right-linear grammar $G'$ with $L(G) = L(G')^R$
Since $G$ is left-linear grammar the productions look like:

\[ A \rightarrow Ba_1a_2 \cdots a_k \]

\[ A \rightarrow a_1a_2 \cdots a_k \]
Construct right-linear grammar $G'$

In $G$:

$$A \rightarrow Ba_1a_2 \cdots a_k$$

$$A \rightarrow Bv$$

In $G'$:

$$A \rightarrow a_k \cdots a_2a_1B$$

$$A \rightarrow v^RB$$
Construct right-linear grammar $G'$

In $G$:

\[ A \rightarrow a_1a_2 \cdots a_k \]
\[ A \rightarrow \nu \]

In $G'$:

\[ A \rightarrow a_k \cdots a_2a_1 \]
\[ A \rightarrow \nu^R \]
It is easy to see that: \[ L(G) = L(G')^R \]

Since \( G' \) is right-linear, we have:

\[ L(G') \quad \rightarrow \quad L(G')^R \quad \rightarrow \quad L(G) \]

Regular Language \quad \rightarrow \quad Regular Language \quad \rightarrow \quad Regular Language
Proof - Part 2

\[
\begin{aligned}
\{ \text{Languages} \} & \supseteq \{ \text{Regular Languages} \} \\
\{ \text{Generated by Regular Grammars} \} & \supseteq \{ \text{Regular Languages} \}
\end{aligned}
\]

Any regular language \( L \) is generated by some regular grammar \( G \)
Any regular language $L$ is generated by some regular grammar $G$

Proof idea:

Let $M$ be the NFA with $L = L(M)$.

Construct from $M$ a regular grammar $G$ such that $L(M) = L(G)$
Since $L$ is regular
there is an NFA $M$ such that $L = L(M)$

Example:

$L = ab^* ab (b^* ab)^*$

$L = L(M)$
Convert $M$ to a right-linear grammar

$q_0 \rightarrow aq_1$
\[ q_0 \rightarrow aq_1 \] 
\[ q_1 \rightarrow bq_1 \] 
\[ q_1 \rightarrow aq_2 \]
\[ q_0 \rightarrow aq_1 \]
\[ q_1 \rightarrow bq_1 \]
\[ q_1 \rightarrow aq_2 \]
\[ q_2 \rightarrow bq_3 \]
\[ L(G) = L(M) = L \]

\[
\begin{align*}
G \\
q_0 & \rightarrow aq_1 \\
q_1 & \rightarrow bq_1 \\
q_1 & \rightarrow aq_2 \\
q_2 & \rightarrow bq_3 \\
q_3 & \rightarrow q_1 \\
q_3 & \rightarrow \lambda
\end{align*}
\]
In General

For any transition:

\[ q \rightarrow ap \]

Add production:

variable  terminal  variable
For any final state: \( q_f \)

Add production: \( q_f \rightarrow \lambda \)
Since $G$ is right-linear grammar

$G$ is also a regular grammar

with $L(G) = L(M) = L$