Rewrite Systems and Grammars

A *rewrite system* (or *production system* or *rule-based system*) is:

- a list of rules, and
- an algorithm for applying them.

Each rule has a left-hand side and a right-hand side.

Example rules:

\[
S \rightarrow aSb \\
\]  
\[
aS \rightarrow \varepsilon \\
\]  
\[
aSb \rightarrow bS \alpha bSa \\
\]


A Rewrite System Formalism

A rewrite system formalism specifies:

- The form of the rules
- How *simple-rewrite* works:
  - How to choose rules?
  - When to quit?
A Rewrite System Formalism

\( \text{simple-rewrite}(R: \text{rewrite system}, \ w: \text{initial string}) = \)

1. Set \( \text{working-string} \) to \( w \).

2. Until told by \( R \) to halt do:
   Match the lhs of some rule against some part of \( \text{working-string} \).
   Replace the matched part of \( \text{working-string} \) with the rhs of the rule that was matched.

3. Return \( \text{working-string} \).
A Rewrite System Formalism

**EXAMPLE:**

\[ w = S_a S \]

Rules:

1. \[ S \rightarrow a S_b \]
2. \[ a S \rightarrow \epsilon \]

- What order to apply the rules?
- When to quit?
Rule Based Systems

- Expert systems
- Cognitive modeling
- Business practice modeling
- General models of computation
- Grammars
Grammars

Define Languages

A grammar is a set of rules that are stated in terms of two alphabets:

- a **terminal alphabet**, $\Sigma$, that contains the symbols that make up the strings in $L(G)$, and

- a **nonterminal alphabet**, the elements of which will function as working symbols that will be used while the grammar is operating. These symbols will disappear by the time the grammar finishes its job and generates a string.

- A grammar has a unique start symbol, often called $S$. 
Grammars

A grammar $G$ is a quadruple, 
$$(V, \Sigma, R, S),$$  
where:

- $V$ is the rule alphabet, which contains nonterminals and terminals.
- $\Sigma$ (the set of terminals) is a subset of $V$,
- $R$ (the set of rules) is a finite subset of $V^*(V - \Sigma)V^* \times V^*$, 
- $S$ (the start symbol) is an element of $V - \Sigma$. 
Grammars

An element \((\alpha, \beta)\) in \(R\) is called a production and is written as \(\alpha \rightarrow \beta\).

\(\alpha\) is referred to as the left hand side of the production and \(\beta\) is referred to as the right hand side of the production.

Notice that the left side must have at least one nonterminal.
Grammars

Example: \( G_1 = (V, \Sigma, R, S) \) where

- \( V = \{A, S, 0, 1\} \)
- \( \Sigma = \{0, 1\} \)
- \( R = \{S \rightarrow 0A1, 0A \rightarrow 00A1, A \rightarrow \varepsilon\} \)

Note: Grammars are generators, while automata are recognizers. A grammar defines a language in a recursive manner.
Grammars

• A sentential form of a grammar $G = (V, \Sigma, R, S)$ is defined recursively as follows:
  1. $S$ is a sentential form
  2. If $\alpha\beta\gamma$ is a sentential form and $\beta \rightarrow \delta$ is a production in $R$, then $\alpha\delta\gamma$ is also a sentential form

• A sentential form of $G$ containing no nonterminal symbol is called a sentence generated by $G$. 
Grammars

• A **sentential form** of a grammar $G = (V, \Sigma, R, S)$ is defined recursively as follows:
  1. $S$ is a sentential form
  2. If $\alpha\beta\gamma$ is a sentential form and $\beta \rightarrow \delta$ is a production in $R$, then $\alpha\delta\gamma$ is also a sentential form

• A sentential form of $G$ containing no nonterminal symbol is called a *sentence* generated by $G$. 
Grammars

The *language* generated by a grammar \( G \), denoted by \( L(G) \), is the set of all sentences generated by \( G \).

**Terminology:**

\( \Rightarrow_G \) (directly derives) is a relation on \( V^* \). If \( \beta \rightarrow \delta \) is a production in \( R \), then \( \alpha\beta\gamma \Rightarrow_G \alpha\delta\gamma \)

\( \Rightarrow^k \) k-fold product of \( \Rightarrow \)

\( \Rightarrow^+ \) transitive closure of \( \Rightarrow \)

\( \Rightarrow^* \) reflexive transitive closure of \( \Rightarrow \)
Grammars

If \( \alpha \Rightarrow^k \beta \) then there is a sequence of strings \( \alpha_0, \alpha_i, \alpha_{i+1}, \alpha_k \) such that

\[
\alpha_0 = \alpha, \quad \alpha_{i-1} \Rightarrow \alpha_i \quad (1 \leq i \leq k), \quad \alpha_k = \beta
\]

This sequence of strings is called a derivation sequence of length \( k \).

\[
L(G) = \{ w | w \text{ in } \sum^* \text{ and } S \Rightarrow^* w \}
\]

Example: \( L(G_1) = \{0^n1^n | n > 0\} \)
Classification of Grammars

**Definition:** A grammar $G$ is said to be

1) **Right-linear** if each production in $R$ is of the form $A \rightarrow xB$ or $A \rightarrow x$ where $A$ and $B$ are in $V-\Sigma$ and $x$ in $\Sigma^*$.

2) **Context-free** if each production is of the form $A \rightarrow \alpha$, where $A$ is in $V-\Sigma$ and $\alpha$ is in $V^*$.

3) **Context-sensitive** if each production is of the form $\alpha \rightarrow \beta$, where $|\alpha| \leq |eta|$.

4) A grammar with no restrictions as above is called **unrestricted**.
Classification of Grammars

If a language L is generated by a ‘type x’ grammar, then the language L is said to be ‘type x’ language.

Right-linear grammars generate right-linear languages, context-free grammars (CFGs) generate context-free languages (CFLs), context-sensitive grammars (CSGs) generate context-sensitive languages (CSLs).

The four types are referred to as Chomsky hierarchy.
# Classification of Grammars

**Examples:**

<table>
<thead>
<tr>
<th>A → B</th>
<th>C</th>
<th>S → 0A1</th>
</tr>
</thead>
<tbody>
<tr>
<td>B → 0B</td>
<td>1B</td>
<td>011</td>
</tr>
<tr>
<td>C → 0D</td>
<td>1C</td>
<td>ε</td>
</tr>
<tr>
<td>D → 0C</td>
<td>1D</td>
<td>A → 01</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
S & \rightarrow B \\
S & \rightarrow aS_a \\
B & \rightarrow \varepsilon \\
B & \rightarrow bB
\end{align*}
\]

\[
\begin{align*}
S & \rightarrow 0A1 \\
0A & \rightarrow 00A1 \\
A & \rightarrow \varepsilon
\end{align*}
\]
Lemma 1
(1) $\emptyset$, (2) $\{ \varepsilon \}$, (3) $\{ a \}$ for all $a$ in $\Sigma$ are right-linear languages.

Proof:
In each case we construct a right-linear grammar generating the language.

(1) $G = (\{S\}, \Sigma, \emptyset, S)$ is a right linear grammar such that $L(G) = \emptyset$.

(2) $G = (\{S\}, \Sigma, \{ S \rightarrow \varepsilon \}, S)$ is a right-linear grammar such that $L(G) = \{ \varepsilon \}$.

(3) $G_a = (\{S, a\}, \{ a \}, \{ S \rightarrow a \}, S)$ is a right-linear grammar such that $L(G_a) = \{ a \}$. 
Equivalence of regular languages and right-linear languages

Lemma 2
If $L_1$ and $L_2$ are right-linear languages, then
(1) $L_1 \cup L_2$, (2) $L_1 L_2$, (3) $L_1^*$ are right-linear languages.

Proof: Since $L_1$ and $L_2$ are right-linear languages, we can assume that there are right-linear grammars $G_1 = (V_1, \Sigma, R_1, S_1)$ and $G_2 = (V_2, \Sigma, R_2, S_2)$ such that $L(G_1) = L_1$ and $L(G_2) = L_2$.

We can assume that nonterminals of $G_1$ and $G_2$ are disjoint (otherwise rename the symbols).

Define the following grammars:

(1) $G_3 = (V_1 \cup V_2 \cup \{S_3\}, \Sigma, R_1 \cup R_2 \cup \{S_3 \rightarrow S_1 | S_2\}, S_3)$
Equivalence of regular languages and right-linear languages

(2) $G_4 = (V_1 \cup V_2, \Sigma, R_4, S_1)$ where $R_4$ is defined as follows:
   (i) if $A \rightarrow xB$ is in $R_1$, then $A \rightarrow xB$ is in $R_4$
   (ii) if $A \rightarrow x$ is in $R_1$, then $A \rightarrow xS_2$ is in $R_4$
   (iii) All productions in $R_2$ are in $R_4$

(3) $G_5 = (V_1 \cup \{S_5\}, \Sigma, R_5, S_5)$ where $R_5$ is defined as follows:
   (i) if $A \rightarrow xB$ is in $R_1$, then $A \rightarrow xB$ is in $R_5$
   (ii) if $A \rightarrow x$ is in $R_1$, then $A \rightarrow xS_5$ and $A \rightarrow x$ are in $R_5$
   (iii) $S_5 \rightarrow S_1 \mid \varepsilon$ are in $R_5$

Claim: (1) $L(G_3) = L(G_1) \cup L(G_2)$,
      (2) $L(G_4) = L(G_1)L(G_2)$, and
      (3) $L(G_5) = (L(G_1))^*$.
Equivalence of regular languages and right-linear languages

(Proof of claim (2):
If $S_1 \Rightarrow^* w_1$ in $G_1$ then $S_1 \Rightarrow^* w_1S_2$ in $G_4$.
If $S_2 \Rightarrow^* x$ in $G_2$ then $S_2 \Rightarrow x^*$ in $G_4$.
So, $L(G_1)L(G_2) \subseteq L(G_4)$.
Now suppose $S_1 \Rightarrow^* w$ in $G_4$. Since there are no productions of the form $A \rightarrow x$ in $G_4$ that came out of $G_1$, we can write the derivation in the form $S_1 \Rightarrow^* xS_2 \Rightarrow^* xy$ where $w = xy$. By construction, there must be derivations of the form $S_1 \Rightarrow^* x$ and $S_2 \Rightarrow^* y$. Hence, $L(G_4) \subseteq L(G_1)L(G_2)$.

Proofs of claims 1 and 3 are left as exercise.
Equivalence of regular languages and right-linear languages

**Theorem**: A regular language is a right-linear language

**Proof**: Follows from Lemma 1 and inductive applications of Lemma 2 on the construction of the regular language.
Left-linear grammars

Definitions:
A grammar \( G = (V, \Sigma, R, S) \) is said to be left-linear if each production is of the form \( A \rightarrow Bx \) or \( A \rightarrow x \), where \( A \) and \( B \) are in \( V-\Sigma \) and \( x \) is in \( \Sigma^* \).

\[
\begin{align*}
A & \rightarrow B \mid C \\
B & \rightarrow B0 \mid B1 \mid 110 \\
C & \rightarrow D0 \mid C1 \mid \varepsilon \\
D & \rightarrow C0 \mid D1
\end{align*}
\]
Equivalence of left-linear languages and regular languages

Let \( G = (V, \Sigma, R, S) \) be a left-linear grammar.
Let \( G' = (V, \Sigma, R', S) \) be the grammar obtained from \( G \) as follows:
If \( A \rightarrow \alpha \) is a production in \( R \) add \( A \rightarrow \alpha^R \) to \( R' \)
Then \( L(G') = L(G)' \). \( G' \) is a right-linear grammar.
Since \( L(G') \) is a regular language. \( L(G)' \) is also regular.
If \( L \) is a regular language then there is a left linear grammar \( G \) such that \( L = L(G) \)