**Introduction**

- We have seen that some languages can be represented by regular expressions, which can be also recognized by finite automata.
- There are languages, such as \( \{0^n1^n \mid n \geq 0\} \) that cannot be described (specified) by finite automata or regular expressions.
- Grammars provide a more powerful mechanism for language specification beyond finite automata.

**Grammars and Languages**

Hantao Zhang  
hzhang@cs.uiowa.edu  
The University of Iowa, Department of Computer Science  
Based on Professor Rus’ Lecture Notes

**More Terminology**

- The specification rules of a grammar are also called *productions* or *substitution rules*.
- Some symbols in the rules serve as a wildcard to denote a set of strings and are called *nonterminals* or *variables*, e.g., \( S \) and \( A \) in \( G_1 \).
- Some symbols in the rules will appear in the language specified by the rules and are called *terminals*, e.g., 0 and 1 in \( G_1 \). They are analogous to the input alphabet of an automaton.

**Grammar: Informal**

- A grammar consists of a collection of specification rules where one variable is designated as *start symbol*.
- **Example:** The grammar \( G_1 \) has the following specification rules:

\[
S \rightarrow A \\
A \rightarrow 01 \\
A \rightarrow \epsilon
\]
Example String Generation

Using CFG $G_1$ we can generate the string 000111 as follows:
$S \Rightarrow A \Rightarrow 0A1 \Rightarrow 00A11 \Rightarrow 000A111 \Rightarrow 000111$

- The sequence of substitutions used to obtain a string using a grammar is called a *derivation*.
- All strings of terminals generated in this way constitute the language specified by the grammar.
- We write $L(G)$ for the language generated by the grammar $G$. Thus, $L(G_1) = \{0^n1^n \mid n \geq 0\}$.

Language Specification

A grammar is used for a language specification by generating each string of the language in following manner:

1. Let $x$ be the start variable; it is the *lhs* of one of the specification rules, the top rule, unless specified otherwise.
2. Find a substring of $x$ which is identical to the *lhs* of a rule. Replace the substring with the *rhs* of that rule in $x$.
3. Repeat step 2 until no variables remain in $x$ thus generated.

Formal Definition of a Grammar

A grammar is a 4-tuple $G = (V, \Sigma, R, S)$ where:

1. $V$ is a finite set of symbols called the *variables* or *nonterminals*.
2. $\Sigma$ is an alphabet, disjoint from $V$, called *constants* or *terminals*.
3. $R$ is a finite set of *rules* (or specification rules) of the form $\text{lhs} \rightarrow \text{rhs}$, where $\text{lhs} \in (V \cup \Sigma)^+$, $\text{lhs}$ contains at least one symbol in $V$, and $\text{rhs} \in (V \cup \Sigma)^*$.
4. $S \in V$ is the start variable.

More on Specification Rules

- The *lhs* of a specification rule is a nonempty string of variables and terminals, containing at least one variable.
- The *rhs* of a specification rule consists of a string of variables and terminals.
### Derivation

- If $u, v, w \in (V \cup \Sigma)^*$ (i.e., are strings of variables and terminals) and $\alpha \rightarrow \beta \in R$ (i.e., is a rule of the grammar) then we say that $uvw$ yields $u\beta v$, written $u\alpha v \Rightarrow u\beta v$

- We may also say that $u\beta v$ is **directly derived** from $uvw$ using the rule $\alpha \rightarrow \beta$.

- We write $u \Rightarrow^n v$ if there exists a sequence $u_0, u_1, u_2, \ldots, u_k \in (V \cup \Sigma)^*$, for $k > 0$, and $u = u_0 \Rightarrow u_1 \Rightarrow u_2 \Rightarrow \ldots \Rightarrow u_k = v$.

- We also say that $u_0, u_1, u_2, \ldots, u_k$ is a derivation of $v$ from $u$.

- We write $u \Rightarrow v$ if $u = v$ or $u \Rightarrow v$ for some $k > 0$.

### Example of a Grammar

$G = (\{S, A\}, \{0, 1\}, P, S)$ where $P$:

- $S \rightarrow A$
- $A \rightarrow 0A1$
- $A \rightarrow \epsilon$

### Types of $\alpha \rightarrow \beta$

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha \in V$</td>
<td>$\beta = \epsilon$</td>
<td>$\epsilon$-rule</td>
</tr>
<tr>
<td>$\alpha \in V$</td>
<td>$\beta \in V$</td>
<td>unit</td>
</tr>
<tr>
<td>$\alpha \in V$</td>
<td>$\beta \in \Sigma^*$</td>
<td>terminating</td>
</tr>
<tr>
<td>$\alpha \in V$</td>
<td>$\beta \in (V + \epsilon)\Sigma^*$</td>
<td>left-linear</td>
</tr>
<tr>
<td>$\alpha \in V$</td>
<td>$\beta \in \Sigma^*(V + \epsilon)$</td>
<td>right-linear</td>
</tr>
<tr>
<td>$\alpha \in V$</td>
<td>$\beta \in \Sigma^*(V + \epsilon)$</td>
<td>linear</td>
</tr>
<tr>
<td>$\alpha \in V$</td>
<td>$\beta = uAv$</td>
<td>context-free</td>
</tr>
<tr>
<td>$\alpha = uAv$, $\beta = uvw$, $u, v \in (V + \Sigma)^*$, $A \in V$, $w \in (V + \Sigma)^+$</td>
<td>context-sensitive</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$</td>
<td>\alpha</td>
</tr>
<tr>
<td></td>
<td></td>
<td>phrase structure</td>
</tr>
</tbody>
</table>

### Language Specified by $G$

If $G = (V, \Sigma, R, S)$ is a grammar then the language specified by $G$ (or the language of $G$) is

$L(G) = \{w \in \Sigma^* | S \Rightarrow w\}$
More notations

To distinguish nonterminal from terminal strings we often enclose nonterminals in angular parentheses, ⟨⟩, and terminals in quotes “,”.

If two or more rules have the same lhs, as in the example $A \rightarrow 0A1$ and $A \rightarrow \epsilon$, we may compact them using the form

$lhs \rightarrow rhs_1 | rhs_2 | \ldots | rhs_n$

where | is used with the meaning of an “or”.

E.g., the rules $A \rightarrow 0A1$ and $A \rightarrow B$ may be written $A \rightarrow 0A1 | \epsilon$.

Types of Grammars

Let $G = (V, \Sigma, P, S)$

<table>
<thead>
<tr>
<th>$\forall p \in P$</th>
<th>$G \ &amp; \ L(G)$</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p$ is left-linear</td>
<td>left-linear</td>
<td>3</td>
</tr>
<tr>
<td>$p$ is right-linear</td>
<td>right-linear</td>
<td>3</td>
</tr>
<tr>
<td>$p$ is linear</td>
<td>linear</td>
<td>2.5</td>
</tr>
<tr>
<td>$p$ is context-free</td>
<td>context-free</td>
<td>2</td>
</tr>
<tr>
<td>$p$ is context-sensitive</td>
<td>context-sensitive</td>
<td>1</td>
</tr>
<tr>
<td>$p$ is nondecreasing</td>
<td>nondecreasing</td>
<td>1</td>
</tr>
<tr>
<td>$p$ is phrase structure</td>
<td>phrase structure</td>
<td>0</td>
</tr>
</tbody>
</table>

Note

The CFG $G_2$ has ten variables (capitalized and in angular brackets) and 9 terminals (written in the standard English alphabet) plus a space character

Also, the CFG $G_2$ has 18 rules

Examples strings that belongs to $L(G_2)$ are:

- a boy sees
- the boy sees a flower
- a girl with a flower likes the boy

CFG $G_2$

The CFG $G_2$ specifies a fragment of English

- ⟨SENTENCE⟩ → ⟨NOUN – PHRASE⟩⟨VERB – PHRASE⟩
- ⟨NOUN – PHRASE⟩ → ⟨CP – NOUN⟩ | ⟨CP – NOUN⟩⟨PREP – PHRASE⟩
- ⟨VERB – PHRASE⟩ → ⟨CP – VERB⟩ | ⟨CP – VERB⟩⟨PREP – PHRASE⟩
- ⟨PREP – PHRASE⟩ → ⟨PREP⟩⟨CP – NOUN⟩
- ⟨CP – NOUN⟩ → ⟨ARTICLE⟩⟨NOUN⟩
- ⟨CP – VERB⟩ → ⟨VERB⟩ | ⟨VERB⟩⟨NOUN – PHRASE⟩
- ⟨ARTICLE⟩ → a | the
- ⟨NPUN⟩ → boy | girl | flower
- ⟨VERB⟩ → touches | likes | sees
- ⟨PREP⟩ → with
Linear grammar rules

Let $G = (V, \Sigma, P, S)$ be a CFG and $r \in P$.

- $r$ is linear if $\text{rhs}(r) \in \Sigma^* \circ V \circ \Sigma^*$
- $r$ is right-linear if $\text{rhs}(r) \in \Sigma^* \circ V$
- $r$ is left-linear if $\text{lhs}(r) \in V \circ \Sigma^*$
- $r$ is terminating if $\text{lhs}(r) \in \Sigma^*$

Example Derivation with $G_2$

\[
\begin{align*}
\text{(SENTENCE)} & \Rightarrow \text{\{NOUN \- PHRASE\}\{VERB \- PHRASE\}} \\
& \Rightarrow \text{\{CP \- NOUN\}\{VERB \- PHRASE\}} \\
& \Rightarrow \text{\{ARTICLE\}\{NOUN\}\{VERB \- PHRASE\}} \\
& \Rightarrow a \text{\{NOUN\}\{VERB \- PHRASE\}} \\
& \Rightarrow a \text{\{CP \- VERB\}} \\
& \Rightarrow a \text{\{VERB\}} \\
& \Rightarrow a \text{\{VERB\}} \text{sees}
\end{align*}
\]

Theorem 2.19

If $L \subseteq \Sigma^*$ is a regular language then there is a right linear grammar $G$ with $L(G) = L$

Proof: by construction. Since $L$ is regular let DFA $D = (Q, \Sigma, \delta, q_0, F)$ with $L(D) = L$ and $\Sigma \cap Q = \emptyset$. Construct $G = (Q, \Sigma, P, q_0)$ where:

$P = \{ q \rightarrow tq' | t \in \Sigma \land \delta(q, t) = q' \} \cup \{ q \rightarrow \epsilon | q \in F \}$

1. $G$ is right-linear and directly simulates $D$
2. if $x = x_1x_2\ldots x_n \in L(D)$ with $q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow \ldots \rightarrow q_n \in \delta(q_0, x)$
   \[
   \delta(q_{n-1}, x_n) = q_n
   \]
3. Since $q_n \in F$ and $q_n \rightarrow \epsilon \in P$, $x_1x_2\ldots x_n \in L(D)$
4. Hence, $L(D) = L(G)$

Example of Linear Grammar

$G = (\{A, B\}, \{0, 1\}, \{A \rightarrow 0A | B, B \rightarrow 1B | \epsilon\}, A)$ is a right-linear grammar

$A \Rightarrow 0A \Rightarrow 00A \Rightarrow 00B \Rightarrow 001B \Rightarrow 0011B \Rightarrow 00111B \Rightarrow 001111$ is a derivation in $G$

$L(G) = 0^*1^*$

Computation Theory – p.16/99

Computation Theory – p.17/99

Computation Theory – p.18/99

Computation Theory – p.19/99
Lemma 2.20

For each right-linear grammar \( G = (N, \Sigma, P, S) \) there is a right-linear grammar \( G' = (V', \Sigma, P', S) \) where each non-terminating rule \( A \rightarrow \alpha \in P' \) has \( |\alpha| \leq 2 \) and each terminating rule \( A \rightarrow x \in P' \) has \( |x| \leq 1 \).

Proof idea: Note that \( G \) does not forbid erasing \( \varepsilon \) or unit rules (\( \varepsilon \) may belong to a regular language). For \( A \rightarrow \alpha \in P \) with \( |\alpha| > 2 \), to transform it equivalently we can use construction from Fleck’s book...

Example

Consider \( G = (\{A, B\}, \{a, b\}, \{A \rightarrow abA|abB, B \rightarrow aabB|b\}, A) \). \( G' \) stated in Lemma 2.20 is \( G' = (\{A, X_1, Y_1, B, Z_1, Z_2\}, \{a, b\}, P', A) \) where \( P' \) is:

\[
\begin{align*}
A &\rightarrow aX_1|aY_1, X_1 \rightarrow bA, Y_1 \rightarrow bB, \\
B &\rightarrow aZ_1|b, Z_1 \rightarrow aZ_2, Z_2 \rightarrow aB
\end{align*}
\]

The NFA \( M \) accepting the language specified by \( G' \) constructed in the above example

![Diagram of NFA](image)

Theorem 2.21

For each right-linear grammar \( G = (V, \Sigma, P, S) \), \( L(G) \) is regular

Proof: by construction

1. For \( G = (V, \Sigma, P, S) \), using Lemma 2.20 we may assume that \( P \) has only productions of the form \( X \rightarrow xY \) and \( X \rightarrow z \), \( X, Y \in V \), \( z \in \Sigma \cup \{\varepsilon\} \).

2. An NFA \( M \) that accepts \( L(G) \) is \( M = (V \cup \{\theta\}, \Sigma, \delta, S, \{\theta\}) \) where for each \( X \in V \) and \( z \in \Sigma \cup \{\varepsilon\} \):

   - if \( X \rightarrow z \in P \) then \( \delta(X, z) = \{\theta\} \cup \{Y|X \rightarrow zY \in P\} \)
   - if \( X \rightarrow z \notin P \) then \( \delta(X, z) = \{Y|X \rightarrow zY \in P\} \)

3. It can be directly checked that \( L(M) = L(G) \)
**Relationship**

- The results produced here for regular grammars show that any regular language, $RL$, is a context-free language, $CFL$.
- However, as we know, the language $L = \{0^n1^n | n \geq 0\}$ is not regular.
- A CFG that generates $L = \{0^n1^n | n \geq 0\}$ is:
  $$G = (\{S\}, \{0, 1\}, S \rightarrow 0S1|\epsilon)$$ which shows that not every context free language is regular.
- Hence, $RL \subset CFL$

**Note**

- There is a perfect duality between recognition and generation of regular languages: when an acceptor makes an atomic step it consumes one input symbol; when a grammar rule is used in an atomic step it produces one symbol.
- The results established for right-linear grammars hold also for left-linear grammars:
  **Theorem:** A language is right-linear iff it’s left-linear.
- The term *regular grammar* is further used to refer to a grammar which is right-linear or left-linear.