Chapter 3

Context-Free Grammars

Context-Free Grammars and Languages

- Defn. 3.1.1 A context-free grammar is a quadruple (V, Σ, P, S), where
  - V is a finite set of variables (*non-terminals*)
  - Σ, the *alphabet*, is a finite set of terminal symbols
  - P is a finite set of rules of the form $V \times (V \cup \Sigma)^*$, and
  - $S \in V$, is the start symbol

- A *production rule* of the form $A \to w$, where $w \in (V \cup \Sigma)^*$, applied to the string uAv yields uwv, and u and v define the *context* in which A occurs.
  - Because the context places no limitations on the applicability of a rule, such a grammar is called *context-free grammar* (*CFG*)

Context-Free Grammars and Languages

- Defn. 3.1.2. Let $G = (V, \Sigma, P, S)$ be a CFG and $v \in (V \cup \Sigma)^*$. The set of strings derivable from v is defined *recursively* as follows:

  i) Basis: v is derivable from v

  ii) Recursion: If $u = xAy$ is derivable from v and $A \to w \in$ P, then xwy is derivable from v

  iii) Closure: All strings constructed from v and a finite number of applications of (ii) are derivable from v

- The derivability of $w \in (V \cup \Sigma)^*$ from $v \in (V \cup \Sigma)^+$ is denoted

$$v \overset{*}{\Rightarrow} w \text{, or } v \overset{+}{\Rightarrow} w,\ v \overset{n}{\Rightarrow} w,\ v \underset{G}{\overset{*}{\Rightarrow}} w$$

- The language of the grammar G is the set of *terminal strings* derivable from the start symbol of G

CFG and Languages

- <u>Defn. 3.1.3.</u> Let $G = (V, \Sigma, P, S)$ be a CFG

  (i) A string $w \in (V \cup \Sigma)^*$ is a <u>sentential form</u> of G if $S \overset{*}{\underset{G}{\Rightarrow}} w$

  (ii) A string $w \in \Sigma^*$ is a <u>sentence</u> of G if $S \overset{*}{\underset{G}{\Rightarrow}} w$

  (iii) The <u>language</u> of G, denoted $L(G)$, is the set $\{ w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w \}$

  - A set of strings w over an alphabet is called a CFL if there is a CFG that generates w

- Leftmost (Rightmost) derivation: a derivation that transforms the 1st variable occurring in a string from left-to-right (right-to-left)

  e.g., Fig. 3.1(a) and (b) exhibit a *leftmost* derivation, whereas Fig. 3.1(c) shows a *rightmost* derivation ▶

- The derivation of a string can be graphically depicted by a <u>derivation/parse tree</u> ▶

CFG and Languages

$$
\begin{aligned}
&G = (V, \Sigma, P, S)\\
&V = \{S, A\}\\
&\Sigma = \{a, b\}\\
&P: \quad S \rightarrow AA\\
&\qquad\;\; A \rightarrow AAA \mid bA \mid Ab \mid a
\end{aligned}
$$

$$
\begin{aligned}
S &\Rightarrow \underline{A}A\\
&\Rightarrow a\underline{A}\\
&\Rightarrow a\underline{A}AA\\
&\Rightarrow ab\underline{A}AA\\
&\Rightarrow aba\underline{A}A\\
&\Rightarrow abab\underline{A}A\\
&\Rightarrow ababa\underline{A}\\
&\Rightarrow ababaa
\end{aligned}
$$

(a)

$$
\begin{aligned}
S &\Rightarrow AA\\
&\Rightarrow AAAA\\
&\Rightarrow aAAA\\
&\Rightarrow abAAA\\
&\Rightarrow abaAA\\
&\Rightarrow ababAA\\
&\Rightarrow ababaA\\
&\Rightarrow ababaa
\end{aligned}
$$

(b)

$$
\begin{aligned}
S &\Rightarrow A\underline{A}\\
&\Rightarrow \underline{A}a\\
&\Rightarrow AA\underline{A}a\\
&\Rightarrow AAb\underline{A}a\\
&\Rightarrow A\underline{A}baa\\
&\Rightarrow Ab\underline{A}baa\\
&\Rightarrow \underline{A}babaa\\
&\Rightarrow ababaa
\end{aligned}
$$

(c)

$$
\begin{aligned}
S &\Rightarrow AA\\
&\Rightarrow aA\\
&\Rightarrow aAAA\\
&\Rightarrow aAAa\\
&\Rightarrow abAAa\\
&\Rightarrow abAbAa\\
&\Rightarrow ababAa\\
&\Rightarrow ababaa
\end{aligned}
$$

(d)

FIGURE 3.1 Sample derivations of *ababaa* in G.

CFG and Languages

- Design CFG for the following languages:

  (i) The set $\{ 0^n 1^n \mid n \geq 0 \}$.

  (ii) The set $\{ a^i b^j c^k \mid i \neq j \text{ or } j \neq k \}$, i.e., the set of strings of *a*'s followed by *b*'s followed by *c*'s such that there are either a *different* number of *a*'s and *b*'s or a *different* number of *b*'s and *c*'s, or both.

- Given the following grammar:

$$S \rightarrow A\,1\,B$$
$$A \rightarrow 0A \mid \lambda$$
$$B \rightarrow 0B \mid 1B \mid \lambda$$

Give the *leftmost* and *rightmost derivation* of the string 00101

CFG and Languages

- Defn. 3.1.4. Let $G = (V, \Sigma, P, S)$ be a CFG and $S \overset{*}{\underset{G}{\Rightarrow}} w$ a derivation. The derivation tree, *DT*, of $S \overset{*}{\underset{G}{\Rightarrow}} w$ is an ordered tree that can be built iteratively as follows:

  (i) Initialize *DT* *T* with root S

  (ii) If $A \rightarrow x_1 \ldots x_n$, where $x_i \in (V \cup \Sigma)$, is a rule in the derivation applied to rAv, then add $x_1 \ldots x_n$ as the children of A in T

  (iii) If $A \rightarrow \lambda$ is a rule in the derivation applied to uAv, then add λ as the only child of A in T

  e.g., Fig. 3.2 for Fig. 3.1(a) $S \Rightarrow AA \Rightarrow aA \Rightarrow aAAA$
  $\Rightarrow abAAA \Rightarrow abaAA \Rightarrow ababAA$
  $\Rightarrow ababaA \Rightarrow ababaa$

  Fig. 3.3 for Fig. 3.1(a)...(d)

- Example. Let G be the CFG .э. $P = S \rightarrow zMNz,\ M \rightarrow aMa \mid z,$
  $N \rightarrow bNb \mid z$

  which generates strings of the form $za^nza^nb^mzb^mz$, where $n, m \geq 0$

3.2 Examples of Context-Free Grammar (CFG)

- Many CFGs are the *union* of simpler CFGs, i.e., combining individual grammars by putting their rules S_1, S_2, ..., S_n together using S, the start symbol:

$$S \rightarrow S_1 \mid S_2 \mid \dots \mid S_n$$

- Example. Consider the language $\{ 0^n1^n \mid n \geq 0 \} \cup \{ 1^n0^n \mid n \geq 0 \}$

  Step 1. Construct the CFG for the language $\{ 0^n1^n \mid n \geq 0 \}$

$$S_1 \rightarrow 0\ S_1\ 1 \mid \lambda$$

  Step 2. Construct the CFG for the language $\{ 1^n0^n \mid n \geq 0 \}$

$$S_2 \rightarrow 1\ S_2\ 0 \mid \lambda$$

  Step 3. Construct the CFG for the language $\{ 0^n1^n \mid n \geq 0 \} \cup \{ 1^n0^n \mid n \geq 0 \}$

$$S \rightarrow S_1 \mid S_2$$

$$S_1 \rightarrow 0\ S_1\ 1 \mid \lambda$$

$$S_2 \rightarrow 1\ S_2\ 0 \mid \lambda$$

3.2. Examples of CFG

- Example. Consider the following grammar:

$$S \to aSa \mid bSb \mid a \mid b \mid \lambda$$

where $S \to aSa \mid bSb$ capture the recursive generation process and the grammar generates the set of *palindromes* over $\{a, b\}$

- Example. Consider a CFG which generates the language consisting of even number of a's and even number of b's:

$S \to aB \mid bA \mid \lambda$	{S: even a's and even b's}
$A \to aC \mid bS$	{A: even a's and odd b's}
$B \to aS \mid bC$	{B: odd a's and even b's}
$C \to aA \mid bB$	{C: odd a's and odd b's}

- Example. Same as above except odd a's and odd b's

$$S \to aB \mid bA$$
$$A \to aC \mid bS$$
$$B \to aS \mid bC$$
$$C \to aA \mid bB \mid \lambda$$

4.5 Chomsky Normal Form

- A *simplified* normal form which restricts the length and composition of the R.H.S. of a rule in CFG

- Defn 4.5.1. A CFG $G = (V, \Sigma, P, S)$ is in chomsky normal form if each rule in G has one of the following forms:

  i) $A \rightarrow BC$

  ii) $A \rightarrow a$

  iii) $S \rightarrow \lambda$

  where $A, B, C, S \in V$, and $B, C \in V - \{ S \}$, and $a \in \Sigma$

- The *derivation tree* for a string generated by a CFG in chomsky normal form is a *binary tree*

Chomsky Normal Form

- Theorem 4.5.2. Let $G = (V, \sum, P, S)$ be a CFG. There is an algorithm to construct a grammar $G' = (V', \sum', P', S')$ in *chomsky normal form* that is equivalent to G

Proof (sketch):

(i) For each rule $A \rightarrow w$, where $|w| > 1$, replace each terminal symbol $a \in w$ by a distinct variable Y and create new rule $Y \rightarrow a$.

(ii) For each modified rule $X \rightarrow w$, w is either a terminal or a string in V^+. Rules in the latter form must be broken into a sequence of rules, each of whose R.H.S. consists of two variables.

➢ Example 4.5.1

- One of the applications of using CFGs that are in Chomsky Normal Form

- Constructing binary search trees to accomplish "optimal" time and space search complexity for parsing an input string

3.5 Leftmost Derivations and Ambiguity

- Theorem 3.5.1 Let $G = (V, \Sigma, P, S)$ be a CFG. A string $w \in L(G)$ iff there is a *leftmost derivation* of w from S.

  Proof. It is clear that if there is a leftmost derivation of w from S, $w \in L(G)$.

  We can show that every string in $w \in L(G)$ is derivable in a leftmost manner, i.e., $S \overset{*}{\Rightarrow} w$, is a leftmost derivation.
  If there is any rule application that is not leftmost, the rule applications can be reordered so that they are leftmost.

- Is there a unique leftmost derivation for every string in a CFL?
  - Answer: No. (Consider the two leftmost derivations in Fig. 3.1.) ◀
  - The possibility of a string having several leftmost derivations introduces the notion of **ambiguity**.
  - The ambiguity increases the burden on *debugging* a program, which should be avoided.

3.5 Leftmost Derivations and Ambiguity

- Defn. 3.5.2 A CFG G is ambiguous if there is a string $w \in L(G)$ that can be derived by two distinct leftmost derivations. A grammar that is not ambiguous is called unambiguous.

- Example 3.5.1 The grammar G, which is defined as

$$S \to aS \mid Sa \mid a$$

is *ambiguous*, since there are two leftmost derivations on *aa:*

$$S \Rightarrow aS \Rightarrow aa \qquad \text{and} \qquad S \Rightarrow Sa \Rightarrow aa$$

however, G', which is defined as $S \to aS \mid a$, is *unambiguous*.

- Unfortunately, there are some CFLs that cannot be generated by any *unambiguous* grammars. Such languages are called **inherently ambiguous**.

- A grammar is unambiguous if, at each leftmost-derivation step, there is only one rule that can lead to a derivation of the desired string.

3.5 Leftmost Derivations and Ambiguity

- Example 3.5.2 The *ambiguous* grammar G,

$$S \to bS \mid Sb \mid a$$

can be converted into *unambiguous* grammar G_1 or G_2, where

$$G_1\text{: } S \to bS \mid aA \qquad A \to bA \mid \lambda$$

$$G_2\text{: } S \to bS \mid A \qquad A \to Ab \mid a$$

- Example 3.5.3 The following grammar G is *ambiguous*:

$S \to aSb \mid aSbb \mid \lambda$ (in Example 3.2.4), since

$S \Rightarrow aSb \Rightarrow aaSbbb \Rightarrow aabbb$, and

$S \Rightarrow aSbb \Rightarrow aaSbbb \Rightarrow aabbb$

which can be converted into an *unambiguous* grammar

$$S \to aSb \mid A \mid \lambda \qquad A \to aAbb \mid abb$$

3.5 Leftmost Derivations and Ambiguity

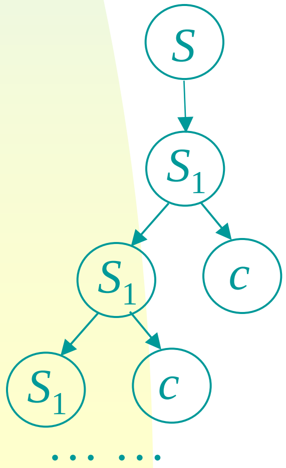
- Example. An *inherently ambiguous* language

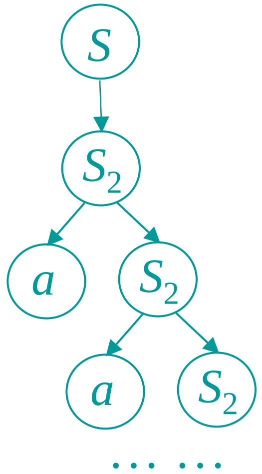
$$L = \{ a^n b^n c^m \mid n, m \geq 0 \} \cup \{ a^n b^m c^m \mid m, n \geq 0 \}$$

  - Every grammar that generates L is *ambiguous*
  - Consider the following grammar of L:

$$S \rightarrow S_1 \mid S_2,$$
$$S_1 \rightarrow S_1 c \mid A, \quad A \rightarrow aAb \mid \lambda$$
$$S_2 \rightarrow aS_2 \mid B, \quad B \rightarrow bBc \mid \lambda$$

  - the strings $\{ a^n b^n c^n \mid n \geq 0 \}$ always have two different DTs, e.g.,





3.5 Leftmost Derivations and Ambiguity

- Another example of *inherently ambiguous* language:

  $$L = \{ a^n b^n c^m d^m \mid n, m > 0 \} \cup \{ a^n b^m c^m d^n \mid n, m > 0 \}$$

- The problem of determining whether an arbitrary language is *inherently ambiguous* is recursively unsolvable.
  - i.e., there is no algorithm that determines whether an arbitrary language is *inherently ambiguous*.

- Reference:

  "Ambiguity in context free languages," S. Ginsburg and J. Ullian, *Journal of the ACM*, (13)1: 62- 89, January 1966.