Chapter 3

Context-Free Languages and PDA's

3.1 Context-Free Grammars

A context-free grammar basically consists of a finite set of grammar rules. In order to define grammar rules, we assume that we have two kinds of symbols: the terminals, which are the symbols of the alphabet underlying the languages under consideration, and the nonterminals, which behave like variables ranging over strings of terminals. A rule is of the form $A \to \alpha$, where A is a single nonterminal, and the right-hand side α is a string of terminal and/or nonterminal symbols. Unlike automata, grammars are used to *generate* strings, rather than recognize strings.

Definition 3.1.1 A *context-free grammar (CFG)* is a quadruple $G = (V, \Sigma, P, S)$, where

- V is a finite set of symbols called the *vocabulary (or set of grammar symbols)*;
- $\Sigma \subseteq V$ is the set of *terminal symbols (for short, terminals)*;
- $S \in (V - \Sigma)$ is a designated symbol called the *start symbol*;
- $P \subseteq (V - \Sigma) \times V^*$ is a finite set of *productions (or rewrite rules, or rules)*.

The set $N = V - \Sigma$ is called the set of *nonterminal symbols (for short, nonterminals)*. Thus, $P \subseteq N \times V^*$, and every production $\langle A, \alpha \rangle$ is also denoted as $A \longrightarrow \alpha$. A production of the form $A \longrightarrow \epsilon$ is called an *epsilon rule, or null rule*.

Remark: Context-free grammars are sometimes defined as $G = (V_N, V_T, P, S)$. The correspondence with our definition is that $\Sigma = V_T$ and $N = V_N$, so that $V = V_N \cup V_T$. Thus, in this other definition, it is necessary to assume that $V_T \cap V_N = \emptyset$.

Example 1. $G_1 = (\{E, a, b\}, \{a, b\}, P, E)$, where P is the set of rules

$$E \longrightarrow aEb,$$
$$E \longrightarrow ab.$$

As we will see shortly, this grammar generates the language $L_1 = \{a^n b^n \mid n \geq 1\}$, which is not regular.

Example 2. $G_2 = (\{E, +, *, (,), a\}, \{+, *, (,), a\}, P, E)$, where P is the set of rules

$$E \longrightarrow E + E,$$
$$E \longrightarrow E * E,$$
$$E \longrightarrow (E),$$
$$E \longrightarrow a.$$

This grammar generates a set of arithmetic expressions.

3.2 Derivations and Context-Free Languages

The productions of a grammar are used to derive strings. In this process, the productions are used as rewrite rules. Formally, we define the derivation relation associated with a context-free grammar.

Definition 3.2.1 Given a context-free grammar $G = (V, \Sigma, P, S)$, the (one-step) *derivation relation* $\Longrightarrow_G$ *associated with* G is the binary relation $\Longrightarrow_G \subseteq V^* \times V^*$ defined as follows: for all $\alpha, \beta \in V^*$, we have

$$\alpha \Longrightarrow_G \beta$$

iff there exist $\lambda, \rho \in V^*$, and some production $(A \to \gamma) \in P$, such that

$$\alpha = \lambda A \rho \quad \text{and} \quad \beta = \lambda \gamma \rho.$$

The transitive closure of $\Longrightarrow_G$ is denoted as $\overset{+}{\Longrightarrow}_G$ and the reflexive and transitive closure of $\Longrightarrow_G$ is denoted as $\overset{*}{\Longrightarrow}_G$.

When the grammar G is clear from the context, we usually omit the subscript G in $\Longrightarrow_G$, $\overset{+}{\Longrightarrow}_G$, and $\overset{*}{\Longrightarrow}_G$.

A string $\alpha \in V^*$ such that $S \overset{*}{\Longrightarrow} \alpha$ is called a *sentential form*, and a string $w \in \Sigma^*$ such that $S \overset{*}{\Longrightarrow} w$ is called a *sentence*. A derivation $\alpha \overset{*}{\Longrightarrow} \beta$ involving n steps is denoted as $\alpha \overset{n}{\Longrightarrow} \beta$.

Note that a derivation step

$$\alpha \Longrightarrow_G \beta$$

is rather nondeterministic. Indeed, one can choose among various occurrences of nonterminals A in α, and also among various productions $A \to \gamma$ with left-hand side A.

For example, using the grammar

$$G_1 = (\{E, a, b\}, \{a, b\}, P, E),$$

where P is the set of rules

$$E \longrightarrow aEb,$$
$$E \longrightarrow ab,$$

every derivation from E is of the form

$$E \stackrel{*}{\Longrightarrow} a^n E b^n \Longrightarrow a^n abb^n = a^{n+1} b^{n+1},$$

or

$$E \stackrel{*}{\Longrightarrow} a^n E b^n \Longrightarrow a^n aEbb^n = a^{n+1} E b^{n+1},$$

where $n \geq 0$.

Grammar G_1 is very simple: every string $a^n b^n$ has a unique derivation. This is usually not the case.

For example, using the grammar

$$G_2 = (\{E, +, *, (,), a\}, \{+, *, (,), a\}, P, E),$$

where P is the set of rules

$$\begin{aligned} E &\longrightarrow E + E, \\ E &\longrightarrow E * E, \\ E &\longrightarrow (E), \\ E &\longrightarrow a, \end{aligned}$$

the string $a + a * a$ has the following distinct derivations, where the boldface indicates which occurrence of E is rewritten:

$$\begin{aligned} \mathbf{E} &\Longrightarrow \mathbf{E} * E \Longrightarrow \mathbf{E} + E * E \\ &\Longrightarrow a + \mathbf{E} * E \Longrightarrow a + a * \mathbf{E} \Longrightarrow a + a * a, \end{aligned}$$

and

$$\begin{aligned} \mathbf{E} &\Longrightarrow \mathbf{E} + E \Longrightarrow a + \mathbf{E} \\ &\Longrightarrow a + \mathbf{E} * E \Longrightarrow a + a * \mathbf{E} \Longrightarrow a + a * a. \end{aligned}$$

In the above derivations, the leftmost occurrence of a nonterminal is chosen at each step. Such derivations are called *leftmost derivations*.

We could systematically rewrite the rightmost occurrence of a nonterminal, getting *rightmost derivations*. The string $a+a*a$ also has the following two rightmost derivations, where the boldface indicates which occurrence of E is rewritten:

$$\begin{aligned}\mathbf{E} &\Longrightarrow E + \mathbf{E} \Longrightarrow E + E * \mathbf{E} \\ &\Longrightarrow E + \mathbf{E} * a \Longrightarrow \mathbf{E} + a * a \Longrightarrow a + a * a,\end{aligned}$$

and

$$\begin{aligned}\mathbf{E} &\Longrightarrow E * \mathbf{E} \Longrightarrow \mathbf{E} * a \\ &\Longrightarrow E + \mathbf{E} * a \Longrightarrow \mathbf{E} + a * a \Longrightarrow a + a * a.\end{aligned}$$

The language generated by a context-free grammar is defined as follows.

Definition 3.2.2 Given a context-free grammar $G = (V, \Sigma, P, S)$, the *language generated by G* is the set

$$L(G) = \{w \in \Sigma^* \mid S \stackrel{+}{\Longrightarrow} w\}.$$

A language $L \subseteq \Sigma^*$ is a *context-free language (for short, CFL)* iff $L = L(G)$ for some context-free grammar G.

It is technically very useful to consider derivations in which the leftmost nonterminal is always selected for rewriting, and dually, derivations in which the rightmost nonterminal is always selected for rewriting.

Definition 3.2.3 Given a context-free grammar $G = (V, \Sigma, P, S)$, the (one-step) *leftmost derivation relation* $\underset{lm}{\Longrightarrow}$ *associated with* G is the binary relation $\underset{lm}{\Longrightarrow} \subseteq V^* \times V^*$ defined as follows: for all $\alpha, \beta \in V^*$, we have

$$\alpha \underset{lm}{\Longrightarrow} \beta$$

iff there exist $u \in \Sigma^*$, $\rho \in V^*$, and some production $(A \to \gamma) \in P$, such that

$$\alpha = uA\rho \quad \text{and} \quad \beta = u\gamma\rho.$$

The transitive closure of $\underset{lm}{\Longrightarrow}$ is denoted as $\overset{+}{\underset{lm}{\Longrightarrow}}$ and the reflexive and transitive closure of $\underset{lm}{\Longrightarrow}$ is denoted as $\overset{*}{\underset{lm}{\Longrightarrow}}$.

The (one-step) *rightmost derivation relation* $\underset{rm}{\Longrightarrow}$ *associated with* G is the binary relation $\underset{rm}{\Longrightarrow} \subseteq V^* \times V^*$ defined as follows: for all $\alpha, \beta \in V^*$, we have

$$\alpha \underset{rm}{\Longrightarrow} \beta$$

iff there exist $\lambda \in V^*$, $v \in \Sigma^*$, and some production $(A \to \gamma) \in P$, such that

$$\alpha = \lambda A v \quad \text{and} \quad \beta = \lambda \gamma v.$$

The transitive closure of $\underset{rm}{\Longrightarrow}$ is denoted as $\overset{+}{\underset{rm}{\Longrightarrow}}$ and the reflexive and transitive closure of $\underset{rm}{\Longrightarrow}$ is denoted as $\overset{*}{\underset{rm}{\Longrightarrow}}$.

Remarks: It is customary to use the symbols a, b, c, d, e for terminal symbols, and the symbols A, B, C, D, E for nonterminal symbols. The symbols u, v, w, x, y, z denote terminal strings, and the symbols $\alpha, \beta, \gamma, \lambda, \rho, \mu$ denote strings in V^*. The symbols X, Y, Z usually denote symbols in V.

Given a CFG $G = (V, \Sigma, P, S)$, *parsing a string w* consists in finding out whether $w \in L(G)$, and if so, in producing a derivation for w.

The following lemma is technically very important. It shows that leftmost and rightmost derivations are "universal". This has some important practical implications for the complexity of parsing algorithms.

Lemma 3.2.4 *Let $G = (V, \Sigma, P, S)$ be a context-free grammar. For every $w \in \Sigma^*$, for every derivation $S \overset{+}{\Longrightarrow} w$, there is a leftmost derivation $S \underset{lm}{\overset{+}{\Longrightarrow}} w$, and there is a rightmost derivation $S \underset{rm}{\overset{+}{\Longrightarrow}} w$.*

Proof. Of course, we have to somehow use induction on derivations, but this is a little tricky, and it is necessary to prove a stronger fact. We treat leftmost derivations, rightmost derivations being handled in a similar way.

Claim: For every $w \in \Sigma^*$, for every $\alpha \in V^+$, for every $n \geq 1$, if $\alpha \overset{n}{\Longrightarrow} w$, then there is a leftmost derivation $\alpha \underset{lm}{\overset{n}{\Longrightarrow}} w$.

The claim is proved by induction on n. □

Lemma 3.2.4 implies that

$$L(G) = \{w \in \Sigma^* \mid S \underset{lm}{\overset{+}{\Longrightarrow}} w\} = \{w \in \Sigma^* \mid S \underset{rm}{\overset{+}{\Longrightarrow}} w\}.$$

We observed that if we consider the grammar

$$G_2 = (\{E, +, *, (,), a\}, \{+, *, (,), a\}, P, E),$$

where P is the set of rules

$$\begin{aligned} E &\longrightarrow E + E, \\ E &\longrightarrow E * E, \\ E &\longrightarrow (E), \\ E &\longrightarrow a, \end{aligned}$$

the string $a + a * a$ has the following two distinct leftmost derivations, where the boldface indicates which occurrence of E is rewritten:

$$\begin{aligned} \mathbf{E} &\Longrightarrow \mathbf{E} * E \Longrightarrow \mathbf{E} + E * E \\ &\Longrightarrow a + \mathbf{E} * E \Longrightarrow a + a * \mathbf{E} \Longrightarrow a + a * a, \end{aligned}$$

and

$$\begin{aligned} \mathbf{E} &\Longrightarrow \mathbf{E} + E \Longrightarrow a + \mathbf{E} \\ &\Longrightarrow a + \mathbf{E} * E \Longrightarrow a + a * \mathbf{E} \Longrightarrow a + a * a. \end{aligned}$$

When this happens, we say that we have an ambiguous grammars. In some cases, it is possible to modify a grammar to make it unambiguous. For example, the grammar G_2 can be modified as follows.

Let

$$G_3 = (\{E, T, F, +, *, (,), a\}, \{+, *, (,), a\}, P, E),$$

where P is the set of rules

$$\begin{aligned} E &\longrightarrow E + T, \\ E &\longrightarrow T, \\ T &\longrightarrow T * F, \\ T &\longrightarrow F, \\ F &\longrightarrow (E), \\ F &\longrightarrow a. \end{aligned}$$

We leave as an exercise to show that $L(G_3) = L(G_2)$, and that every string in $L(G_3)$ has a unique leftmost derivation. Unfortunately, it is not always possible to modify a context-free grammar to make it unambiguous.

There exist context-free languages that have no unambiguous context-free grammars. For example, it can be shown that

$$L_3 = \{a^m b^m c^n \mid m, n \geq 1\} \cup \{a^m b^n c^n \mid m, n \geq 1\}$$

is context-free, but has no unambiguous grammars. All this motivates the following definition.

Definition 3.2.5 A context-free grammar $G = (V, \Sigma, P, S)$ is *ambiguous* if there is some string $w \in L(G)$ that has two distinct leftmost derivations (or two distinct rightmost derivations). Thus, a grammar G is *unambiguous* if every string $w \in L(G)$ has a unique leftmost derivation (or a unique rightmost derivation). A context-free language L is *inherently ambiguous* if every CFG G for L is ambiguous.

Whether or not a grammar is ambiguous affects the complexity of parsing. Parsing algorithms for unambiguous grammars are more efficient than parsing algorithms for ambiguous grammars.

3.3 Normal Forms for Context-Free Grammars, Chomsky Normal Form

One of the main goals of this section is to show that every CFG G can be converted to an equivalent grammar in *Chomsky Normal Form (for short, CNF)*. A context-free grammar $G = (V, \Sigma, P, S)$ is in Chomsky Normal Form iff its productions are of the form

$$\begin{aligned} A &\to BC, \\ A &\to a, \quad \text{or} \\ S &\to \epsilon, \end{aligned}$$

where $A, B, C \in N$, $a \in \Sigma$, $S \to \epsilon$ is in P iff $\epsilon \in L(G)$, and S does not occur on the right-hand side of any production.

The first step to eliminate ϵ-rules is to compute the set $E(G)$ of *erasable (or nullable) nonterminals*

$$E(G) = \{A \in N \mid A \overset{+}{\Longrightarrow} \epsilon\}.$$

The set $E(G)$ is computed using a sequence of approximations E_i defined as follows:

$$\begin{aligned} E_0 &= \{A \in N \mid (A \to \epsilon) \in P\}, \\ E_{i+1} &= E_i \cup \{A \mid \exists (A \to B_1 \dots B_j \dots B_k) \in P, \\ &\qquad\qquad B_j \in E_i,\ 1 \leq j \leq k\}. \end{aligned}$$

Clearly, the E_i form an ascending chain

$$E_0 \subseteq E_1 \subseteq \cdots \subseteq E_i \subseteq E_{i+1} \subseteq \cdots \subseteq N,$$

and since N is finite, there is a least i, say i_0, such that $E_{i_0} = E_{i_0+1}$. We claim that $E(G) = E_{i_0}$. Actually, we prove the following lemma.

Lemma 3.3.1 *Given any context-free grammar* $G = (V, \Sigma, P, S)$*, one can construct a context-free grammar* $G' = (V', \Sigma, P', S')$ *such that:*

(1) $L(G') = L(G)$*;*

(2) P' *contains no* ϵ*-rules other than* $S' \to \epsilon$*, and* $S' \to \epsilon \in P'$ *iff* $\epsilon \in L(G)$*;*

(3) S' *does not occur on the right-hand side of any production in* P'*.*

Proof. We begin by proving that $E(G) = E_{i_0}$. For this, we prove that $E(G) \subseteq E_{i_0}$ and $E_{i_0} \subseteq E(G)$.

Having shown that $E(G) = E_{i_0}$, we construct the grammar G'. Its set of production P' is defined as follows. Let

$$P_1 = \{A \to \alpha \in P \mid \alpha \in V^+\} \cup \{S' \to S\},$$

and let P_2 be the set of productions

$$P_2 = \{A \to \alpha_1\alpha_2 \ldots \alpha_k\alpha_{k+1} \mid \exists \alpha_1 \in V^*, \ldots, \exists \alpha_{k+1} \in V^*, \\ \exists B_1 \in E(G), \ldots, \exists B_k \in E(G) \\ A \to \alpha_1 B_1 \alpha_2 \ldots \alpha_k B_k \alpha_{k+1} \in P, k \geq 1, \alpha_1 \ldots \alpha_{k+1} \neq \epsilon\}.$$

Note that $\epsilon \in L(G)$ iff $S \in E(G)$. If $S \notin E(G)$, then let $P' = P_1 \cup P_2$, and if $S \in E(G)$, then let $P' = P_1 \cup P_2 \cup \{S' \to \epsilon\}$.

We claim that $L(G') = L(G)$, which is proved by showing that every derivation using G can be simulated by a derivation using G', and vice-versa. All the conditions of the lemma are now met. □

From a practical point of view, the construction or lemma 3.3.1 is very costly. For example, given a grammar containing the productions

$$\begin{aligned} S &\to ABCDEF, \\ A &\to \epsilon, \\ B &\to \epsilon, \\ C &\to \epsilon, \\ D &\to \epsilon, \\ E &\to \epsilon, \\ F &\to \epsilon, \\ \ldots &\to \ldots, \end{aligned}$$

eliminating ϵ-rules will create $2^6 - 1 = 63$ new rules corresponding to the 63 nonempty subsets of the set

$$\{A, B, C, D, E, F\}.$$

We now turn to the elimination of chain rules, i.e., rules of the form

$$A \to B$$

where $A, B \in N$.

It turns out that matters are greatly simplified if we first apply lemma 3.3.1 to the input grammar G, and we explain the construction assuming that $G = (V, \Sigma, P, S)$ satisfies the conditions of lemma 3.3.1. For every nonterminal $A \in N$, we define the set

$$I_A = \{B \in N \mid A \overset{+}{\Longrightarrow} B\}.$$

The sets I_A are computed using approximations $I_{A,i}$ defined as follows:

$$I_{A,0} = \{B \in N \mid (A \to B) \in P\},$$
$$I_{A,i+1} = I_{A,i} \cup \{C \in N \mid \exists (B \to C) \in P, \text{ and } B \in I_{A,i}\}.$$

Clearly, for every $A \in N$, the $I_{A,i}$ form an ascending chain

$$I_{A,0} \subseteq I_{A,1} \subseteq \cdots \subseteq I_{A,i} \subseteq I_{A,i+1} \subseteq \cdots \subseteq N,$$

and since N is finite, there is a least i, say i_0, such that $I_{A,i_0} = I_{A,i_0+1}$. We claim that $I_A = I_{A,i_0}$. Actually, we prove the following lemma.

Lemma 3.3.2 *Given any context-free grammar $G = (V, \Sigma, P, S)$, one can construct a context-free grammar $G' = (V', \Sigma, P', S')$ such that:*

(1) $L(G') = L(G)$;

(2) Every rule in P' is of the form $A \to \alpha$ where $|\alpha| \geq 2$, or $A \to a$ where $a \in \Sigma$, or $S' \to \epsilon$ iff $\epsilon \in L(G)$;

(3) S' does not occur on the right-hand side of any production in P'.

Proof. First, we apply lemma 3.3.1 to the grammar G, obtaining a grammar $G_1 = (V_1, \Sigma, S_1, P_1)$. The proof that $I_A = I_{A,i_0}$ is similar to the proof that $E(G) = E_{i_0}$.

We now define the following sets of rules. Let

$$P_2 = P_1 - \{A \to B \mid A \to B \in P_1\},$$

and let

$$P_3 = \{A \to \alpha \mid B \to \alpha \in P_1,\ \alpha \notin N_1,\ B \in I_A\}.$$

We claim that $G' = (V_1, \Sigma, P_2 \cup P_3, S_1)$ satisfies the conditions of the lemma.

Let us apply the method of lemma 3.3.2 to the grammar

$$G_3 = (\{E, T, F, +, *, (,), a\}, \{+, *, (,), a\}, P, E),$$

where P is the set of rules

$$\begin{aligned} E &\longrightarrow E + T, \\ E &\longrightarrow T, \\ T &\longrightarrow T * F, \\ T &\longrightarrow F, \\ F &\longrightarrow (E), \\ F &\longrightarrow a. \end{aligned}$$

We get $I_E = \{T, F\}$, $I_T = \{F\}$, and $I_F = \emptyset$.

The new grammar G_3' has the set of rules

$$
\begin{aligned}
E &\longrightarrow E + T,\\
E &\longrightarrow T * F,\\
E &\longrightarrow (E),\\
E &\longrightarrow a,\\
T &\longrightarrow T * F,\\
T &\longrightarrow (E),\\
T &\longrightarrow a,\\
F &\longrightarrow (E),\\
F &\longrightarrow a.
\end{aligned}
$$

At this stage, the grammar obtained in lemma 3.3.2 no longer has ϵ-rules (except perhaps $S' \rightarrow \epsilon$ iff $\epsilon \in L(G)$) or chain rules. However, it may contain rules $A \rightarrow \alpha$ with $|\alpha| \geq 3$, or with $|\alpha| \geq 2$ and where α contains terminals(s).

To obtain the Chomsky Normal Form. we need to eliminate such rules. This is not difficult, but notationally a bit messy.

Lemma 3.3.3 *Given any context-free grammar* $G = (V, \Sigma, P, S)$*, one can construct a context-free grammar* $G' = (V', \Sigma, P', S')$ *such that* $L(G') = L(G)$ *and* G' *is in Chomsky Normal Form, that is, a grammar whose productions are of the form*

$$\begin{aligned} A &\to BC, \\ A &\to a, \quad \textit{or} \\ S' &\to \epsilon, \end{aligned}$$

where $A, B, C \in N'$*,* $a \in \Sigma$*,* $S' \to \epsilon$ *is in* P' *iff* $\epsilon \in L(G)$*, and* S' *does not occur on the right-hand side of any production in* P'*.*

Proof. First, we apply lemma 3.3.2, obtaining G_1.

Let Σ_r be the set of terminals occurring on the right-hand side of rules $A \to \alpha \in P_1$, with $|\alpha| \geq 2$. For every $a \in \Sigma_r$, let X_a be a new nonterminal not in V_1. Let

$$P_2 = \{X_a \to a \mid a \in \Sigma_r\}.$$

Let $P_{1,r}$ be the set of productions

$$A \to \alpha_1 a_1 \alpha_2 \cdots \alpha_k a_k \alpha_{k+1},$$

where $a_1, \ldots, a_k \in \Sigma_r$ and $\alpha_i \in N_1^*$.

For every production

$$A \to \alpha_1 a_1 \alpha_2 \cdots \alpha_k a_k \alpha_{k+1}$$

in $P_{1,r}$, let

$$A \to \alpha_1 X_{a_1} \alpha_2 \cdots \alpha_k X_{a_k} \alpha_{k+1}$$

be a new production, and let P_3 be the set of all such productions.

Let $P_4 = (P_1 - P_{1,r}) \cup P_2 \cup P_3$.

Now, productions $A \to \alpha$ in P_4 with $|\alpha| \geq 2$ do not contain terminals.

However, we may still have productions $A \to \alpha \in P_4$ with $|\alpha| \geq 3$.

For every production of the form

$$A \to B_1 \cdots B_k,$$

where $k \geq 3$, create the new nonterminals

$$[B_1 \cdots B_{k-1}], [B_1 \cdots B_{k-2}], \cdots, [B_1 B_2 B_3], [B_1 B_2],$$

and the new productions

$$\begin{aligned} A &\to [B_1 \cdots B_{k-1}] B_k, \\ [B_1 \cdots B_{k-1}] &\to [B_1 \cdots B_{k-2}] B_{k-1}, \\ \cdots &\to \cdots, \\ [B_1 B_2 B_3] &\to [B_1 B_2] B_3, \\ [B_1 B_2] &\to B_1 B_2. \end{aligned}$$

All the productions are now in Chomsky Normal Form, and it is clear that the same language is generated. □

Applying the first phase of the method of lemma 3.3.3 to the grammar G'_3, we get the rules

$$\begin{aligned}
E &\longrightarrow EX_+T,\\
E &\longrightarrow TX_*F,\\
E &\longrightarrow X_(EX_),\\
E &\longrightarrow a,\\
T &\longrightarrow TX_*F,\\
T &\longrightarrow X_(EX_),\\
T &\longrightarrow a,\\
F &\longrightarrow X_(EX_),\\
F &\longrightarrow a,\\
X_+ &\longrightarrow +,\\
X_* &\longrightarrow *,\\
X_(&\longrightarrow (,\\
X_) &\longrightarrow).
\end{aligned}$$

After applying the second phase of the method, we get the following grammar in Chomsky Normal Form:

$$
\begin{aligned}
E &\longrightarrow [EX_+]T,\\
[EX_+] &\longrightarrow EX_+,\\
E &\longrightarrow [TX_*]F,\\
[TX_*] &\longrightarrow TX_*,\\
E &\longrightarrow [X_(E]X_),\\
[X_(E] &\longrightarrow X_(E,\\
E &\longrightarrow a,\\
T &\longrightarrow [TX_*]F,\\
T &\longrightarrow [X_(E]X_),\\
T &\longrightarrow a,\\
F &\longrightarrow [X_(E]X_),\\
F &\longrightarrow a,\\
X_+ &\longrightarrow +,\\
X_* &\longrightarrow *,\\
X_(&\longrightarrow (,\\
X_) &\longrightarrow).
\end{aligned}
$$

For large grammars, it is often convenient to use the abbreviation which consists in grouping productions having a common left-hand side, and listing the right-hand sides separated by the symbol |. Thus, a group of productions

$$\begin{aligned} A &\to \alpha_1, \\ A &\to \alpha_2, \\ \cdots &\to \cdots, \\ A &\to \alpha_k, \end{aligned}$$

may be abbreviated as

$$A \to \alpha_1 \mid \alpha_2 \mid \cdots \mid \alpha_k.$$

An interesting corollary of the CNF is the following decidability result.

There is an algorithm which, given any context-free grammar G, given any string $w \in \Sigma^*$, decides whether $w \in L(G)$.

There are much better parsing algorithms than this naive algorithm. We now show that every regular language is context-free.

3.4 Regular Languages are Context-Free

The regular languages can be characterized in terms of very special kinds of context-free grammars, right-linear (and left-linear) context-free grammars.

Definition 3.4.1 A context-free grammar $G = (V, \Sigma, P, S)$ is *left-linear* iff its productions are of the form

$$A \to Ba,$$
$$A \to a,$$
$$A \to \epsilon.$$

where $A, B \in N$, and $a \in \Sigma$.

A context-free grammar $G = (V, \Sigma, P, S)$ is *right-linear* iff its productions are of the form

$$A \to aB,$$
$$A \to a,$$
$$A \to \epsilon.$$

where $A, B \in N$, and $a \in \Sigma$.

The following lemma shows the equivalence between NFA's and right-linear grammars.

Lemma 3.4.2 *A language L is regular if and only if it is generated by some right-linear grammar.*

3.5 Useless Productions in Context-Free Grammars

Given a context-free grammar $G = (V, \Sigma, P, S)$, it may contain rules that are useless for a number of reasons. For example, consider the grammar

$$G_3 = (\{E, A, a, b\}, \{a, b\}, P, E),$$

where P is the set of rules

$$\begin{aligned} E &\longrightarrow aEb, \\ E &\longrightarrow ab, \\ E &\longrightarrow A, \\ A &\longrightarrow bAa. \end{aligned}$$

The problem is that the nonterminal A does not derive any terminal strings, and thus, it is useless, as well as the last two productions.

Let us now consider the grammar

$$G_4 = (\{E, A, a, b, c, d\}, \{a, b, c, d\}, P, E),$$

where P is the set of rules

$$\begin{aligned} E &\longrightarrow aEb, \\ E &\longrightarrow ab, \\ A &\longrightarrow cAd, \\ A &\longrightarrow cd. \end{aligned}$$

This time, the nonterminal A generates strings of the form $c^n d^n$, but there is no derivation $E \overset{+}{\Longrightarrow} \alpha$ from E where A occurs in α. The nonterminal A is not connected to E, and the last two rules are useless. Fortunately, it is possible to find such useless rules, and to eliminate them.

Let $T(G)$ be the set of nonterminals that actually derive some terminal string, i.e.

$$T(G) = \{A \in (V - \Sigma) \mid \exists w \in \Sigma^*,\ A \Longrightarrow^+ w\}.$$

The set $T(G)$ can be defined by stages.

We define the sets T_n $(n \geq 1)$ as follows:

$$T_1 = \{A \in (V - \Sigma) \mid \exists (A \longrightarrow w) \in P, \text{ with } w \in \Sigma^*\},$$

and

$$T_{n+1} = T_n \cup \{A \in (V - \Sigma) \mid \exists (A \longrightarrow \beta) \in P, \text{ with } \beta \in (T_n \cup \Sigma)^*\}$$

It is easy to prove that there is some least n such that $T_{n+1} = T_n$, and that for this n, $T(G) = T_n$.

If $S \notin T(G)$, then $L(G) = \emptyset$, and G is equivalent to the trivial grammar $G' = (\{S\}, \Sigma, \emptyset, S)$.

If $S \in T(G)$, then let $U(G)$ be the set of nonterminals that are actually useful, i.e.,

$$U(G) = \{A \in T(G) \mid \exists \alpha, \beta \in (T(G) \cup \Sigma)^*,\ S \Longrightarrow^* \alpha A \beta\}.$$

The set $U(G)$ can also be computed by stages.

We define the sets U_n $(n \geq 1)$ as follows:

$$U_1 = \{A \in T(G) \mid \exists(S \longrightarrow \alpha A \beta) \in P, \text{ with } \alpha, \beta \in (T(G) \cup \Sigma)^*\},$$

and

$$U_{n+1} = U_n \cup \{B \in T(G) \mid \exists(A \longrightarrow \alpha B \beta) \in P, \text{ with } A \in U_n,\ \alpha, \beta \in (T(G) \cup \Sigma)^*\}.$$

It is easy to prove that there is some least n such that $U_{n+1} = U_n$, and that for this n, $U(G) = U_n \cup \{S\}$.

Then, we can use $U(G)$ to transform G into an equivalent CFG in which every nonterminal is useful (i.e., for which $V - \Sigma = U(G)$). Indeed, simply delete all rules containing symbols not in $U(G)$.

We say that a context-free grammar G is *reduced* if all its nonterminals are useful, i.e., $N = U(G)$.

It should be noted than although dull, the above considerations are important in practice. Certain algorithms for constructing parsers, for example, LR-parsers, may loop if useless rules are not eliminated!

We now consider another normal form for context-free grammars, the Greibach Normal Form.

But first, we need to explain how context-free languages arise as least fixed points of certain language-valued functions induced by context-free grammars.