Chapter 3

Context-Free Grammars, Context-Free Languages, Parse Trees and Ogden's Lemma

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. As usual, first we need to define what the object is (a context-free grammar), and then we need to explain how it is used. Unlike automata, grammars are used to *generate* strings, rather than recognize strings.

Definition 3.1.1 A *context-free grammar (for short, 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 $A \to \alpha$. A production of the form $A \to \epsilon$ 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. First, let us review the concepts of transitive closure and reflexive and transitive closure of a binary relation.

Given a set A, a *binary relation R on A* is any set of ordered pairs, i.e. $R \subseteq A \times A$. For short, instead of binary relation, we often simply say relation. Given any two relations R, S on A, their *composition* $R \circ S$ is defined as

$$R \circ S = \{(x,\, y) \in A \times A \mid \exists z \in A,\ (x,\ z) \in R \text{ and } (z,\, y) \in S\}.$$

The *identity relation I_A on A* is the relation I_A defined such that

$$I_A = \{(x,\, x) \mid x \in A\}.$$

For short, we often denote I_A as I. Note that

$$R \circ I = I \circ R = R$$

for every relation R on A. Given a relation R on A, for any $n \geq 0$ we define R^n as follows:

$$R^0 = I,$$
$$R^{n+1} = R^n \circ R.$$

It is obvious that $R^1 = R$. It is also easily verified by induction that $R^n \circ R = R \circ R^n$. The *transitive closure* R^+ *of the relation* R is defined as

$$R^+ = \bigcup_{n \geq 1} R^n.$$

It is easily verified that R^+ is the smallest transitive relation containing R, and that $(x, y) \in R^+$ iff there is some $n \geq 1$ and some $x_0, x_1, \ldots, x_n \in A$ such that $x_0 = x$, $x_n = y$, and $(x_i, x_{i+1}) \in R$ for all i, $0 \leq i \leq n-1$. The *transitive and reflexive closure* R^* *of the relation* R is defined as

$$R^* = \bigcup_{n \geq 0} R^n.$$

Clearly, $R^* = R^+ \cup I$. It is easily verified that R^* is the smallest transitive and reflexive relation containing R.

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

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

or

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

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

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

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

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 context-free grammar $G = (V, \Sigma, P, S)$, *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.

For $n = 1$, there exist some $\lambda, \rho \in V^*$ and some production $A \to \gamma$, such that $\alpha = \lambda A \rho$ and $w = \lambda\gamma\rho$. Since w is a terminal string, λ, ρ, and γ, are terminal strings. Thus, A is the only nonterminal in α, and the derivation step $\alpha \overset{1}{\Longrightarrow} w$ is a leftmost step (and a rightmost step!).

If $n > 1$, then the derivation $\alpha \overset{n}{\Longrightarrow} w$ is of the form

$$\alpha \Longrightarrow \alpha_1 \overset{n-1}{\Longrightarrow} w.$$

There are two subcases.

Case 1. If the derivation step $\alpha \Longrightarrow \alpha_1$ is a leftmost step $\alpha \underset{lm}{\Longrightarrow} \alpha_1$, by the induction hypothesis, there is a leftmost derivation $\alpha_1 \underset{lm}{\overset{n-1}{\Longrightarrow}} w$, and we get the leftmost derivation

$$\alpha \underset{lm}{\Longrightarrow} \alpha_1 \underset{lm}{\overset{n-1}{\Longrightarrow}} w.$$

Case 2. The derivation step $\alpha \Longrightarrow \alpha_1$ is a not a leftmost step. In this case, there must be some $u \in \Sigma^*$, $\mu, \rho \in V^*$, some nonterminals A and B, and some production $B \to \delta$, such that

$$\alpha = uA\mu B\rho \quad \text{and} \quad \alpha_1 = uA\mu\delta\rho,$$

where A is the leftmost nonterminal in α. Since we have a derivation $\alpha_1 \overset{n-1}{\Longrightarrow} w$ of length $n - 1$, by the induction hypothesis, there is a leftmost derivation

$$\alpha_1 \underset{lm}{\overset{n-1}{\Longrightarrow}} w.$$

Since $\alpha_1 = uA\mu\delta\rho$ where A is the leftmost terminal in α_1, the first step in the leftmost derivation $\alpha_1 \underset{lm}{\overset{n-1}{\Longrightarrow}} w$ is of the form

$$uA\mu\delta\rho \underset{lm}{\Longrightarrow} u\gamma\mu\delta\rho,$$

for some production $A \to \gamma$. Thus, we have a derivation of the form

$$\alpha = uA\mu B\rho \Longrightarrow uA\mu\delta\rho \underset{lm}{\Longrightarrow} u\gamma\mu\delta\rho \overset{n-2}{\underset{lm}{\Longrightarrow}} w.$$

We can commute the first two steps involving the productions $B \to \delta$ and $A \to \gamma$, and we get the derivation

$$\alpha = uA\mu B\rho \underset{lm}{\Longrightarrow} u\gamma\mu B\rho \Longrightarrow u\gamma\mu\delta\rho \overset{n-2}{\underset{lm}{\Longrightarrow}} w.$$

This may no longer be a leftmost derivation, but the first step is leftmost, and we are back in case 1. Thus, we conclude by applying the induction hypothesis to the derivation $u\gamma\mu B\rho \overset{n-1}{\Longrightarrow} w$, as in case 1. □

Lemma 3.2.4 implies that

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

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, the language

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

is context-free, since it is generated by the following context-free grammar:

$$\begin{aligned} S &\to S_1, \\ S &\to S_2, \\ S_1 &\to XC, \\ S_2 &\to AY, \\ X &\to aXb, \\ X &\to ab, \\ Y &\to bYc, \\ Y &\to bc, \\ A &\to aA, \\ A &\to a, \\ C &\to cC, \\ C &\to c. \end{aligned}$$

However, it can be shown that L_3 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.

We now consider various normal forms for context-free 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.

Note that a grammar in Chomsky Normal Form does not have ϵ-rules, i.e., rules of the form $A \to \epsilon$, except when $\epsilon \in L(G)$, in which case $S \to \epsilon$ is the only ϵ-rule. It also does not have *chain rules*, i.e., rules of the form $A \to B$, where $A, B \in N$. Thus, in order to convert a grammar to Chomsky Normal Form, we need to show how to eliminate ϵ-rules and chain rules. This is not the end of the story, since we may still have rules of the form $A \to \alpha$ where either $|\alpha| \geq 3$ or $|\alpha| \geq 2$ and α contains terminals. However, dealing with such rules is a simple recoding matter, and we first focus on the elimination of ϵ-rules and chain rules. It turns out that ϵ-rules must be eliminated first.

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

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

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

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

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 a 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 $E_{i_0} \subseteq E(G)$.

To prove that $E_{i_0} \subseteq E(G)$, we proceed by induction on i. Since $E_0 = \{A \in N \mid (A \to \epsilon) \in P\}$, we have $A \overset{1}{\Longrightarrow} \epsilon$, and thus $A \in E(G)$. By the induction hypothesis, $E_i \subseteq E(G)$. If $A \in E_{i+1}$, either $A \in E_i$ and then $A \in E(G)$, or there is some production $(A \to B_1 \dots B_j \dots B_k) \in P$, such that $B_j \in E_i$ for all j, $1 \leq j \leq k$. By the induction hypothesis, $B_j \overset{+}{\Longrightarrow} \epsilon$ for each j, $1 \leq j \leq k$, and thus

$$A \Longrightarrow B_1 \dots B_j \dots B_k \overset{+}{\Longrightarrow} B_2 \dots B_j \dots B_k \overset{+}{\Longrightarrow} B_j \dots B_k \overset{+}{\Longrightarrow} \epsilon,$$

which shows that $A \in E(G)$.

To prove that $E(G) \subseteq E_{i_0}$, we also proceed by induction, but on the length of a derivation $A \overset{+}{\Longrightarrow} \epsilon$. If $A \overset{1}{\Longrightarrow} \epsilon$, then $A \to \epsilon \in P$, and thus $A \in E_0$ since $E_0 = \{A \in N \mid (A \to \epsilon) \in P\}$. If $A \overset{n+1}{\Longrightarrow} \epsilon$, then

$$A \Longrightarrow \alpha \overset{n}{\Longrightarrow} \epsilon,$$

for some production $A \to \alpha \in P$. If α contains terminals of nonterminals not in $E(G)$, it is impossible to derive ϵ from α, and thus, we must have $\alpha = B_1 \dots B_j \dots B_k$, with $B_j \in E(G)$, for all j, $1 \leq j \leq k$. However, $B_j \overset{n_j}{\Longrightarrow} \epsilon$ where $n_j \leq n$, and by the induction hypothesis, $B_j \in E_{i_0}$. But then, we get $A \in E_{i_0+1} = E_{i_0}$, as desired. □

Having shown that $E(G) = E_{i_0}$, we construct the grammar G'. Its set of production P' is defined as follows. First, we create the production $S' \to S$ where $S' \notin V$, to make sure that S' does not occur on the right-hand side of any rule in P'. 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

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

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

$$S \to ABCDEF,$$
$$A \to \epsilon,$$
$$B \to \epsilon,$$
$$C \to \epsilon,$$
$$D \to \epsilon,$$
$$E \to \epsilon,$$
$$F \to \epsilon,$$
$$\dots \to \dots,$$

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.

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

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

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

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

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 a 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 $E(G) = E_{i_0}$. First, we prove that $I_{A,i} \subseteq I_A$ by induction on i. This is staightforward. Next, we prove that $I_A \subseteq I_{A,i_0}$ by induction on derivations of the form $A \stackrel{+}{\Longrightarrow} B$. In this part of the proof, we use the fact that G_1 has no ϵ-rules except perhaps $S_1 \to \epsilon$, and that S_1 does not occur on the right-hand side of any rule. This implies that a derivation $A \stackrel{n+1}{\Longrightarrow} C$ is necessarily of the form $A \stackrel{n}{\Longrightarrow} B \Longrightarrow C$ for some $B \in N$. Then, in the induction step, we have $B \in I_{A,i_0}$, and thus $C \in I_{A,i_0+1} = I_{A,i_0}$.

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. For example, S_1 does not appear on the right-hand side of any production, since the productions in P_3 have right-hand sides from P_1, and S_1 does not appear on the right-hand side in P_1. It is also easily shown that $L(G') = L(G_1) = L(G)$. $\square$

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 $S' \to \epsilon$ iff $\epsilon \in L(G)$) or chain rules. However, it may contain rules $A \to \alpha$ 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 a 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 $|\alpha| \geq 3$. We can perform some recoding using some new nonterminals. 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 a context-free grammar G, given any string $w \in \Sigma^*$, decides whether $w \in L(G)$. Indeed, we first convert G to a grammar G' in Chomsky Normal Form. If $w = \epsilon$, we can test whether $\epsilon \in L(G)$, since this is the case iff $S' \to \epsilon \in P'$. If $w \neq \epsilon$, letting $n = |w|$, note that since the rules are of the form $A \to BC$ or $A \to a$, where $a \in \Sigma$, any derivation for w has $n - 1 + n = 2n - 1$ steps. Thus, we enumerate all (leftmost) derivations of length $2n - 1$.

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

$$\begin{aligned} A &\to Ba, \\ A &\to a, \\ A &\to \epsilon. \end{aligned}$$

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

$$\begin{aligned} A &\to aB, \\ A &\to a, \\ A &\to \epsilon. \end{aligned}$$

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.*

Proof. Let $L = L(D)$ for some DFA $D = (Q, \Sigma, \delta, q_0, F)$. We construct a right-linear grammar G as follows. Let $V = Q \cup \Sigma$, $S = q_0$, and let P be defined as follows:

$$P = \{p \to aq \mid q = \delta(p, a),\ p, q \in Q,\ a \in \Sigma\} \cup \{p \to \epsilon \mid p \in F\}.$$

It is easily shown by induction on the length of w that

$$p \stackrel{*}{\Longrightarrow} wq \quad \text{iff} \quad q = \delta^*(p, w),$$

and thus, $L(D) = L(G)$.

Conversely, let $G = (V, \Sigma, P, S)$ be a right-linear grammar. First, let $G = (V', \Sigma, P', S)$ be the right-linear grammar obtained from G by adding the new terminal E to N, replacing every rule in P of the form $A \to a$ where $a \in \Sigma$ by the rule $A \to aE$, and adding the rule $E \to \epsilon$. It is immediately verified that $L(G') = L(G)$. Next, we construct the NFA $M = (Q, \Sigma, \delta, q_0, F)$ as follows: $Q = N'$, $q_0 = S$, $F = \{A \in N' \mid A \to \epsilon\}$, and

$$\delta(A, a) = \{B \in N \mid A \to aB \in P'\},$$

for all $A \in N'$ and all $a \in \Sigma$. It is easily shown by induction on the length of w that

$$A \stackrel{*}{\Longrightarrow} wB \quad \text{iff} \quad B \in \delta^*(A, w),$$

and thus, $L(M) = L(G') = L(G)$. □

A similar lemma holds for left-linear grammars. It is also easily shown that the regular languages are exactly the languages generated by context-free grammars whose rules are of the form

$$A \to Bu,$$
$$A \to u,$$

where $A, B \in N$, and $u \in \Sigma^*$.

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

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

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

$$E \longrightarrow aEb,$$
$$E \longrightarrow ab,$$
$$A \longrightarrow cAd,$$
$$A \longrightarrow cd.$$

This time, the nonterminal A generates strings of the form $c^n d^n$, but there is no derivation $E \stackrel{+}{\Longrightarrow} \alpha$ 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 $U(G)$. The details are left as an exercise. 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.

3.6 The Greibach Normal Form

Every CFG G can also be converted to an equivalent grammar in *Greibach Normal Form (for short, GNF)*. A context-free grammar $G = (V, \Sigma, P, S)$ is in Greibach Normal Form iff its productions are of the form

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

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.

Note that a grammar in Greibach Normal Form does not have ϵ-rules other than possibly $S \to \epsilon$. More importantly, except for the special rule $S \to \epsilon$, every rule produces some terminal symbol.

An important consequence of the Greibach Normal Form is that every nonterminal is not left recursive. A nonterminal A is *left recursive* iff $A \stackrel{+}{\Longrightarrow} A\alpha$ for some $\alpha \in V^*$. Left recursive nonterminals cause top-down determinitic parsers to loop. The Greibach Normal Form provides a way of avoiding this problem.

There are no easy proofs that every CFG can be converted to a Greibach Normal Form. A particularly elegant method due to Rosenkrantz using least fixed-points and matrices will be given in section 3.9.

Lemma 3.6.1 *Given a 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 Greibach Normal Form, that is, a grammar whose productions are of the form*

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

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'.

3.7 Least Fixed-Points

Context-free languages can also be characterized as least fixed-points of certain functions induced by grammars. This characterization yields a rather quick proof that every context-free grammar can be converted to Greibach Normal Form. This characterization also reveals very clearly the recursive nature of the context-free languages.

We begin by reviewing what we need from the theory of partially ordered sets.

Definition 3.7.1 Given a partially ordered set $\langle A, \leq \rangle$, an *ω-chain* $(a_n)_{n \geq 0}$ is a sequence such that $a_n \leq a_{n+1}$ for all $n \geq 0$. The *least-upper bound* of an ω-chain (a_n) is an element $a \in A$ such that:

(1) $a_n \leq a$, for all $n \geq 0$;

(2) For any $b \in A$, if $a_n \leq b$, for all $n \geq 0$, then $a \leq b$.

A partially ordered set $\langle A, \leq \rangle$ is an *ω-chain complete poset* iff it has a least element $\bot$, and iff every ω-chain has a least upper bound denoted as $\bigsqcup a_n$.

Remark: The ω in ω-chain means that we are considering countable chains (ω is the ordinal associated with the order-type of the set of natural numbers). This notation may seem arcane, but is standard in denotational semantics.

For example, given any set X, the power set 2^X ordered by inclusion is an ω-chain complete poset with least element $\emptyset$. The Cartesian product $\underbrace{2^X \times \cdots \times 2^X}_{n}$ ordered such that

$$(A_1, \ldots, A_n) \leq (B_1, \ldots, B_n)$$

iff $A_i \subseteq B_i$ (where $A_i, B_i \in 2^X$) is an ω-chain complete poset with least element $(\emptyset, \ldots, \emptyset)$.

We are interested in functions between partially ordered sets.

Definition 3.7.2 Given any two partially ordered sets $\langle A_1, \leq_1 \rangle$ and $\langle A_2, \leq_2 \rangle$, a function $f \colon A_1 \to A_2$ is *monotonic* iff for all $x, y \in A_1$,

$$x \leq_1 y \quad \text{implies that} \quad f(x) \leq_2 f(y).$$

If $\langle A_1, \leq_1 \rangle$ and $\langle A_2, \leq_2 \rangle$ are ω-chain complete posets, a function $f \colon A_1 \to A_2$ is *ω-continuous* iff it is monotonic, and for every ω-chain (a_n),

$$f(\bigsqcup a_n) = \bigsqcup f(a_n).$$

Remark: Note that we are not requiring that an ω-continuous function $f: A_1 \to A_2$ preserve least elements, i.e., it is possible that $f(\bot_1) \neq \bot_2$.

We now define the crucial concept of a least fixed-point.

Definition 3.7.3 Let $\langle A, \leq \rangle$ be a partially ordered set, and let $f: A \to A$ be a function. A *fixed-point of* f is an element $a \in A$ such that $f(a) = a$. The *least fixed-point of* f is an element $a \in A$ such that $f(a) = a$, and for every $b \in A$ such that $f(b) = b$, then $a \leq b$.

The following lemma gives sufficient conditions for the existence of least fixed-points. It is one of the key lemmas in denotational semantics.

Lemma 3.7.4 *Let $\langle A, \leq \rangle$ be an ω-chain complete poset with least element $\bot$. Every ω-continuous function $f: A \to A$ has a unique least fixed-point x_0 given by*

$$x_0 = \bigsqcup f^n(\bot).$$

Furthermore, for any $b \in A$ such that $f(b) \leq b$, then $x_0 \leq b$.

Proof. First, we prove that the sequence

$$\bot\,, f(\bot)\,, f^2(\bot),\ \ldots,\ f^n(\bot),\ \ldots$$

is an ω-chain. This is shown by induction on n. Since $\bot$ is the least element of A, we have $\bot \leq f(\bot)$. Assuming by induction that $f^n(\bot) \leq f^{n+1}(\bot)$, since f is ω-continuous, it is monotonic, and thus we get $f^{n+1}(\bot) \leq f^{n+2}(\bot)$, as desired.

Since A is an ω-chain complete poset, the ω-chain $(f^n(\bot))$ has a least upper bound

$$x_0 = \bigsqcup f^n(\bot).$$

Since f is ω-continuous, we have

$$f(x_0) = f(\bigsqcup f^n(\bot)) = \bigsqcup f(f^n(\bot)) = \bigsqcup f^{n+1}(\bot) = x_0,$$

and x_0 is indeed a fixed-point of f.

Clearly, if $f(b) \leq b$ implies that $x_0 \leq b$, then $f(b) = b$ implies that $x_0 \leq b$. Thus, assume that $f(b) \leq b$ for some $b \in A$. We prove by induction of n that $f^n(\bot) \leq b$. Indeed, $\bot \leq b$, since $\bot$ is the least element of A. Assuming by induction that $f^n(\bot) \leq b$, by monotonicity of f, we get

$$f(f^n(\bot)) \leq f(b),$$

and since $f(b) \leq b$, this yields

$$f^{n+1}(\bot) \leq b.$$

Since $f^n(\bot) \leq b$ for all $n \geq 0$, we have

$$x_0 = \bigsqcup f^n(\bot) \leq b.$$

□

The second part of lemma 3.7.4 is very useful to prove that functions have the same least fixed-point. For example, under the conditions of lemma 3.7.4, if $g: A \to A$ is another ω-chain continuous function, letting x_0 be the least fixed-point of f and y_0 be the least fixed-point of g, if $f(y_0) \leq y_0$ and $g(x_0) \leq x_0$, we can deduce that $x_0 = y_0$. Indeed, since $f(y_0) \leq y_0$ and x_0 is the least fixed-point of f, we get $x_0 \leq y_0$, and since $g(x_0) \leq x_0$ and y_0 is the least fixed-point of g, we get $y_0 \leq x_0$, and therefore $x_0 = y_0$.

Lemma 3.7.4 also shows that the least fixed-point x_0 of f can be approximated as much as desired, using the sequence $(f^n(\bot))$. We will now apply this fact to context-free grammars. For this, we need to show how a context-free grammar $G = (V, \Sigma, P, S)$ with m nonterminals induces an ω-continuous map

$$\Phi_G: \underbrace{2^{\Sigma^*} \times \cdots \times 2^{\Sigma^*}}_{m} \to \underbrace{2^{\Sigma^*} \times \cdots \times 2^{\Sigma^*}}_{m}.$$

3.8 Context-Free Languages as Least Fixed-Points

Given a context-free grammar $G = (V, \Sigma, P, S)$ with m nonterminals $A_1, \ldots A_m$, grouping all the productions having the same left-hand side, the grammar G can be concisely written as

$$\begin{aligned}
A_1 &\to \alpha_{1,1} + \cdots + \alpha_{1,n_1},\\
\cdots &\to \cdots\\
A_i &\to \alpha_{i,1} + \cdots + \alpha_{i,n_i},\\
\cdots &\to \cdots\\
A_m &\to \alpha_{m,1} + \cdots + \alpha_{m,n_n}.
\end{aligned}$$

Given any set A, let $\mathcal{P}_{fin}(A)$ be the set of finite subsets of A.

Definition 3.8.1 Let $G = (V, \Sigma, P, S)$ be a context-free grammar with m nonterminals A_1, ..., A_m. For any m-tuple $\Lambda = (L_1, \ldots, L_m)$ of languages $L_i \subseteq \Sigma^*$, we define the function

$$\Phi[\Lambda]: \mathcal{P}_{fin}(V^*) \to 2^{\Sigma^*}$$

inductively as follows:

$$\begin{aligned}
\Phi[\Lambda](\emptyset) &= \emptyset,\\
\Phi[\Lambda](\{\epsilon\}) &= \{\epsilon\},\\
\Phi[\Lambda](\{a\}) &= \{a\}, \quad \text{if } a \in \Sigma,\\
\Phi[\Lambda](\{A_i\}) &= L_i, \quad \text{if } A_i \in N,\\
\Phi[\Lambda](\{\alpha X\}) &= \Phi[\Lambda](\{\alpha\})\Phi[\Lambda](\{X\}), \quad \text{if } \alpha \in V^+,\ X \in V,\\
\Phi[\Lambda](Q \cup \{\alpha\}) &= \Phi[\Lambda](Q) \cup \Phi[\Lambda](\{\alpha\}), \quad \text{if } Q \in \mathcal{P}_{fin}(V^*),\ Q \neq \emptyset,\ \alpha \in V^*,\ \alpha \notin Q.
\end{aligned}$$

Then, writing the grammar G as

$$\begin{aligned} A_1 &\to \alpha_{1,1} + \cdots + \alpha_{1,n_1}, \\ \cdots &\to \cdots \\ A_i &\to \alpha_{i,1} + \cdots + \alpha_{i,n_i}, \\ \cdots &\to \cdots \\ A_m &\to \alpha_{m,1} + \cdots + \alpha_{m,n_n}, \end{aligned}$$

we define the map

$$\Phi_G: \underbrace{2^{\Sigma^*} \times \cdots \times 2^{\Sigma^*}}_{m} \to \underbrace{2^{\Sigma^*} \times \cdots \times 2^{\Sigma^*}}_{m}$$

such that

$$\Phi_G(L_1, \ldots L_m) = (\Phi[\Lambda](\{\alpha_{1,1}, \ldots, \alpha_{1,n_1}\}), \ldots, \Phi[\Lambda](\{\alpha_{m,1}, \ldots, \alpha_{m,n_m}\}))$$

for all $\Lambda = (L_1, \ldots, L_m) \in \underbrace{2^{\Sigma^*} \times \cdots \times 2^{\Sigma^*}}_{m}$.

One should verify that the map $\Phi[\Lambda]$ is well defined, but this is easy. The following lemma is easily shown:

Lemma 3.8.2 *Given a context-free grammar $G = (V, \Sigma, P, S)$ with m nonterminals A_1, ..., A_m, the map*

$$\Phi_G: \underbrace{2^{\Sigma^*} \times \cdots \times 2^{\Sigma^*}}_{m} \to \underbrace{2^{\Sigma^*} \times \cdots \times 2^{\Sigma^*}}_{m}$$

is ω-continuous.

Now, $\underbrace{2^{\Sigma^*} \times \cdots \times 2^{\Sigma^*}}_{m}$ is an ω-chain complete poset, and the map Φ_G is ω-continous. Thus, by lemma 3.7.4, the map Φ_G has a least-fixed point. It turns out that the components of this least fixed-point are precisely the languages generated by the grammars (V, Σ, P, A_i). Before proving this fact, let us give an example illustrating it.

Example. Consider the grammar $G = (\{A, B, a, b\}, \{a, b\}, P, A)$ defined by the rules

$$\begin{aligned} A &\to BB + ab, \\ B &\to aBb + ab. \end{aligned}$$

The least fixed-point of Φ_G is the least upper bound of the chain

$$(\Phi_G^n(\emptyset, \emptyset)) = ((\Phi_{G,A}^n(\emptyset, \emptyset), \Phi_{G,B}^n(\emptyset, \emptyset)),$$

where

$$\Phi_{G,A}^0(\emptyset, \emptyset) = \Phi_{G,B}^0(\emptyset, \emptyset) = \emptyset,$$

and

$$\Phi^{n+1}_{G,A}(\emptyset,\emptyset) = \Phi^{n}_{G,B}(\emptyset,\emptyset)\Phi^{n}_{G,B}(\emptyset,\emptyset) \cup \{ab\},$$
$$\Phi^{n+1}_{G,B}(\emptyset,\emptyset) = a\Phi^{n}_{G,B}(\emptyset,\emptyset)b \cup \{ab\}.$$

It is easy to verify that

$$\Phi^{1}_{G,A}(\emptyset,\emptyset) = \{ab\},$$
$$\Phi^{1}_{G,B}(\emptyset,\emptyset) = \{ab\},$$
$$\Phi^{2}_{G,A}(\emptyset,\emptyset) = \{ab, abab\},$$
$$\Phi^{2}_{G,B}(\emptyset,\emptyset) = \{ab, aabb\},$$
$$\Phi^{3}_{G,A}(\emptyset,\emptyset) = \{ab, abab, abaabb, aabbab, aabbaabb\},$$
$$\Phi^{3}_{G,B}(\emptyset,\emptyset) = \{ab, aabb, aaabbb\}.$$

By induction, we can easily prove that the two components of the least fixed-point are the languages

$$L_A = \{a^m b^m a^n b^n \mid m, n \geq 1\} \cup \{ab\} \quad \text{and} \quad L_B = \{a^n b^n \mid n \geq 1\}.$$

Letting $G_A = (\{A, B, a, b\}, \{a, b\}, P, A)$ and $G_B = (\{A, B, a, b\}, \{a, b\}, P, B)$, it is indeed true that $L_A = L(G_A)$ and $L_B = L(G_B)$.

We have the following theorem due to Ginsburg and Rice:

Theorem 3.8.3 *Given a context-free grammar $G = (V, \Sigma, P, S)$ with m nonterminals A_1, ..., A_m, the least fixed-point of the map Φ_G is the m-tuple of languages*

$$(L(G_{A_1}), \ldots, L(G_{A_m})),$$

where $G_{A_i} = (V, \Sigma, P, A_i)$.

Proof. Writing G as

$$A_1 \to \alpha_{1,1} + \cdots + \alpha_{1,n_1},$$
$$\cdots \to \cdots$$
$$A_i \to \alpha_{i,1} + \cdots + \alpha_{i,n_i},$$
$$\cdots \to \cdots$$
$$A_m \to \alpha_{m,1} + \cdots + \alpha_{m,n_n},$$

let $M = \max\{|\alpha_{i,j}|\}$ be the maximum length of right-hand sides of rules in P. Let

$$\Phi^{n}_{G}(\emptyset, \ldots, \emptyset) = (\Phi^{n}_{G,1}(\emptyset, \ldots, \emptyset), \ldots, \Phi^{n}_{G,m}(\emptyset, \ldots, \emptyset)).$$

Then, for any $w \in \Sigma^*$, observe that

$$w \in \Phi^1_{G,i}(\emptyset, \ldots, \emptyset)$$

iff there is some rule $A_i \to \alpha_{i,j}$ with $w = \alpha_{i,j}$, and that

$$w \in \Phi^n_{G,i}(\emptyset, \ldots, \emptyset)$$

for some $n \geq 2$ iff there is some rule $A_i \to \alpha_{i,j}$ with $\alpha_{i,j}$ of the form

$$\alpha_{i,j} = u_1 A_{j_1} u_2 \cdots u_k A_{j_k} u_{k+1},$$

where $u_1, \ldots, u_{k+1} \in \Sigma^*$, $k \geq 1$, and some $w_1, \ldots, w_k \in \Sigma^*$ such that

$$w_h \in \Phi^{n-1}_{G,j_h}(\emptyset, \ldots, \emptyset),$$

and

$$w = u_1 w_1 u_2 \cdots u_k w_k u_{k+1}.$$

We prove the following two claims.

Claim 1: For every $w \in \Sigma^*$, if $A_i \overset{n}{\Longrightarrow} w$, then $w \in \Phi^p_{G,i}(\emptyset, \ldots, \emptyset)$, for some $p \geq 1$.

Claim 2: For every $w \in \Sigma^*$, if $w \in \Phi^n_{G,i}(\emptyset, \ldots, \emptyset)$, with $n \geq 1$, then $A_i \overset{p}{\Longrightarrow} w$ for some $p \leq (M+1)^{n-1}$.

Proof of Claim 1. We proceed by induction on n. If $A_i \overset{1}{\Longrightarrow} w$, then $w = \alpha_{i,j}$ for some rule $A \to \alpha_{i,j}$, and by the remark just before the claim, $w \in \Phi^1_{G,i}(\emptyset, \ldots, \emptyset)$.

If $A_i \overset{n+1}{\Longrightarrow} w$ with $n \geq 1$, then

$$A_i \overset{n}{\Longrightarrow} \alpha_{i,j} \Longrightarrow w$$

for some rule $A_i \to \alpha_{i,j}$. If

$$\alpha_{i,j} = u_1 A_{j_1} u_2 \cdots u_k A_{j_k} u_{k+1},$$

where $u_1, \ldots, u_{k+1} \in \Sigma^*$, $k \geq 1$, then $A_{j_h} \overset{n_h}{\Longrightarrow} w_h$, where $n_h \leq n$, and

$$w = u_1 w_1 u_2 \cdots u_k w_k u_{k+1}$$

for some $w_1, \ldots, w_k \in \Sigma^*$. By the induction hypothesis,

$$w_h \in \Phi^{p_h}_{G,j_h}(\emptyset, \ldots, \emptyset),$$

for some $p_h \geq 1$, for every h, $1 \leq h \leq k$. Letting $p = \max\{p_1, \ldots, p_k\}$, since each sequence $(\Phi^q_{G,i}(\emptyset, \ldots, \emptyset))$ is an ω-chain, we have $w_h \in \Phi^p_{G,j_h}(\emptyset, \ldots, \emptyset)$ for every h, $1 \leq h \leq k$, and by the remark just before the claim, $w \in \Phi^{p+1}_{G,i}(\emptyset, \ldots, \emptyset)$. $\square$

Proof of Claim 2. We proceed by induction on n. If $w \in \Phi^1_{G,i}(\emptyset, \ldots, \emptyset)$, by the remark just before the claim, then $w = \alpha_{i,j}$ for some rule $A \to \alpha_{i,j}$, and $A_i \overset{1}{\Longrightarrow} w$.

If $w \in \Phi^n_{G,i}(\emptyset, \ldots, \emptyset)$ for some $n \geq 2$, then there is some rule $A_i \to \alpha_{i,j}$ with $\alpha_{i,j}$ of the form

$$\alpha_{i,j} = u_1 A_{j_1} u_2 \cdots u_k A_{j_k} u_{k+1},$$

where $u_1, \ldots, u_{k+1} \in \Sigma^*$, $k \geq 1$, and some $w_1, \ldots, w_k \in \Sigma^*$ such that

$$w_h \in \Phi^{n-1}_{G,j_h}(\emptyset, \ldots, \emptyset),$$

and

$$w = u_1 w_1 u_2 \cdots u_k w_k u_{k+1}.$$

By the induction hypothesis, $A_{j_h} \overset{p_h}{\Longrightarrow} w_h$ with $p_h \leq (M+1)^{n-2}$, and thus

$$A_i \Longrightarrow u_1 A_{j_1} u_2 \cdots u_k A_{j_k} u_{k+1} \overset{p_1}{\Longrightarrow} \cdots \overset{p_k}{\Longrightarrow} w,$$

so that $A_i \overset{p}{\Longrightarrow} w$ with

$$p \leq p_1 + \cdots + p_k + 1 \leq M(M+1)^{n-2} + 1 \leq (M+1)^{n-1},$$

since $k \leq M$. □

Combining Claim 1 and Claim 2, we have

$$L(G_{A_i}) = \bigcup_n \Phi^n_{G,i}(\emptyset, \ldots, \emptyset),$$

which proves that the least fixed-point of the map Φ_G is the m-tuple of languages

$$(L(G_{A_1}), \ldots, L(G_{A_m})).$$

□

We now show how theorem 3.8.3 can be used to give a short proof that every context-free grammar can be converted to Greibach Normal Form.

3.9 Least Fixed-Points and the Greibach Normal Form

The hard part in converting a grammar $G = (V, \Sigma, P, S)$ to Greibach Normal Form is to convert it to a grammar in so-called *weak Greibach Normal Form*, where the productions are of the form

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

where $a \in \Sigma$, $\alpha \in V^*$, and if $S \to \epsilon$ is a rule, then S does not occur on the right-hand side of any rule. Indeed, if we first convert G to Chomsky Normal Form, it turns out that we will get rules of the form $A \to aBC$, $A \to aB$ or $A \to a$.

Using the algorithm for eliminating ϵ-rules and chain rules, we can first convert the original grammar to a grammar with no chain rules and no ϵ-rules except possibly $S \to \epsilon$, in which case, S does not appear on the right-hand side of rules. Thus, for the purpose of converting to weak Greibach Normal Form, we can assume that we are dealing with grammars without chain rules and without ϵ-rules. Let us also assume that we computed the set $T(G)$ of nonterminals that actually derive some terminal string, and that useless productions involving symbols not in $T(G)$ have been deleted.

Let us explain the idea of the conversion using the following grammar:

$$A \to AaB + BB + b.$$
$$B \to Bd + BAa + aA + c.$$

The first step is to group the right-hand sides α into two categories: those whose leftmost symbol is a terminal ($\alpha \in \Sigma V^*$) and those whose leftmost symbol is a nonterminal ($\alpha \in NV^*$). It is also convenient to adopt a matrix notation, and we can write the above grammar as

$$(A, B) = (A, B) \begin{pmatrix} aB & \emptyset \\ B & \{d, Aa\} \end{pmatrix} + (b, \{aA, c\})$$

Thus, we are dealing with matrices (and row vectors) whose entries are finite subsets of V^*. For notational simplicity, braces around singleton sets are omitted. The finite subsets of V^* form a semiring, where addition is union, and multiplication is concatenation. Addition and multiplication of matrices are as usual, except that the semiring operations are used. We will also consider matrices whose entries are languages over Σ. Again, the languages over Σ form a semiring, where addition is union, and multiplication is concatenation. The identity element for addition is $\emptyset$, and the identity element for multiplication is $\{\epsilon\}$. As above, addition and multiplication of matrices are as usual, except that the semiring operations are used. For example, given any languages $A_{i,j}$ and $B_{i,j}$ over Σ, where $i, j \in \{1, 2\}$, we have

$$\begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix} \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} = \begin{pmatrix} A_{1,1}B_{1,1} \cup A_{1,2}B_{2,1} & A_{1,1}B_{1,2} \cup A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} \cup A_{2,2}B_{2,1} & A_{2,1}B_{1,2} \cup A_{2,2}B_{2,2} \end{pmatrix}$$

Letting $X = (A, B)$, $K = (b, \{aA, c\})$, and

$$H = \begin{pmatrix} aB & \emptyset \\ B & \{d, Aa\} \end{pmatrix}$$

the above grammar can be concisely written as

$$X = XH + K.$$

More generally, given any context-free grammar $G = (V, \Sigma, P, S)$ with m nonterminals $A_1, \ldots, A_m$, assuming that there are no chain rules, no ϵ-rules, and that every nonterminal belongs to $T(G)$, letting

$$X = (A_1, \ldots, A_m),$$

we can write G as

$$X = XH + K,$$

for some appropriate $m \times m$ matrix H in which every entry contains a set (possibly empty) of strings in V^+, and some row vector K in which every entry contains a set (possibly empty) of strings α each beginning with a terminal ($\alpha \in \Sigma V^*$).

Given an $m \times m$ square matrix $A = (A_{i,j})$ of languages over Σ, we can define the matrix A^* whose entry $A^*_{i,j}$ is given by

$$A^*_{i,j} = \bigcup_{n \geq 0} A^n_{i,j},$$

where $A^0 = Id_m$, the identity matrix, and A^n is the n-th power of A. Similarly, we define A^+ where

$$A^+_{i,j} = \bigcup_{n \geq 1} A^n_{i,j}.$$

Given a matrix A where the entries are finite subset of V^*, where $N = \{A_1, \ldots, A_m\}$, for any m-tuple $\Lambda = (L_1, \ldots, L_m)$ of languages over Σ, we let

$$\Phi[\Lambda](A) = (\Phi[\Lambda](A_{i,j})).$$

Given a system $X = XH + K$ where H is an $m \times m$ matrix and X, K are row matrices, if H and K do not contain any nonterminals, we claim that the least fixed-point of the grammar G associated with $X = XH + K$ is KH^*. This is easily seen by computing the approximations $X^n = \Phi^n_G(\emptyset, \ldots, \emptyset)$. Indeed, $X^0 = K$, and

$$X^n = KH^n + KH^{n-1} + \cdots + KH + K = K(H^n + H^{n-1} + \cdots + H + I_m).$$

Similarly, if Y is an $m \times m$ matrix of nonterminals, the least fixed-point of the grammar associated with $Y = HY + H$ is H^+ (provided that H does not contain any nonterminals).

Given any context-free grammar $G = (V, \Sigma, P, S)$ with m nonterminals $A_1, \ldots, A_m$, writing G as $X = XH + K$ as explained earlier, we can form another grammar GH by creating m^2 new nonterminals $Y_{i,j}$, where the rules of this new grammar are defined by the system of two matrix equations

$$X = KY + K,$$
$$Y = HY + H,$$

where $Y = (Y_{i,j})$.

The following lemma is the key to the Greibach Normal Form.

Lemma 3.9.1 *Given any context-free grammar $G = (V, \Sigma, P, S)$ with m nonterminals A_1, ..., A_m, writing G as*

$$X = XH + K$$

as explained earlier, if GH is the grammar defined by the system of two matrix equations

$$X = KY + K,$$
$$Y = HY + H,$$

as explained above, then the components in X of the least-fixed points of the maps Φ_G and Φ_{GH} are equal.

Proof. Let U be the least-fixed point of Φ_G, and let (V, W) be the least fixed-point of Φ_{GH}. We shall prove that $U = V$. For notational simplicity, let us denote $\Phi[U](H)$ as $H[U]$ and $\Phi[U](K)$ as $K[U]$.

Since U is the least fixed-point of $X = XH + K$, we have

$$U = UH[U] + K[U].$$

Since $H[U]$ and $K[U]$ do not contain any nonterminals, by a previous remark, $K[U]H^*[U]$ is the least-fixed point of $X = XH[U] + K[U]$, and thus,

$$K[U]H^*[U] \leq U.$$

On the other hand, by monotonicity,

$$K[U]H^*[U]H\Big[K[U]H^*[U]\Big] + K\Big[K[U]H^*[U]\Big] \leq K[U]H^*[U]H[U] + K[U] = K[U]H^*[U],$$

and since U is the least fixed-point of $X = XH + K$,

$$U \leq K[U]H^*[U].$$

Therefore, $U = K[U]H^*[U]$. We can prove in a similar manner that $W = H[V]^+$.

Let $Z = H[U]^+$. We have

$$K[U]Z + K[U] = K[U]H[U]^+ + K[U] = K[U]H[U]^* = U,$$

and

$$H[U]Z + H[U] = H[U]H[U]^+ + H[U] = H[U]^+ = Z,$$

and since (V, W) is the least fixed-point of $X = KY + K$ and $Y = HY + H$, we get $V \leq U$ and $W \leq H[U]^+$.

We also have

$$V = K[V]W + K[V] = K[V]H[V]^+ + K[V] = K[V]H[V]^*,$$

and

$$VH[V] + K[V] = K[V]H[V]^*H[V] + K[V] = K[V]H[V]^* = V,$$

and since U is the least fixed-point of $X = XH + K$, we get $U \leq V$. Therefore, $U = V$, as claimed. $\square$

Note that the above lemma actually applies to any grammar. Applying lemma 3.9.1 to our example grammar, we get the following new grammar:

$$(A, B) = (b, \{aA, c\}) \begin{pmatrix} Y_1 & Y_2 \\ Y_3 & Y_4 \end{pmatrix} + (b, \{aA, c\}),$$
$$\begin{pmatrix} Y_1 & Y_2 \\ Y_3 & Y_4 \end{pmatrix} = \begin{pmatrix} aB & \emptyset \\ B & \{d, Aa\} \end{pmatrix} \begin{pmatrix} Y_1 & Y_2 \\ Y_3 & Y_4 \end{pmatrix} + \begin{pmatrix} aB & \emptyset \\ B & \{d, Aa\} \end{pmatrix}$$

There are still some nonterminals appearing as leftmost symbols, but using the equations defining A and B, we can replace A with

$$\{bY_1, aAY_3, cY_3, b\}$$

and B with

$$\{bY_2, aAY_4, cY_4, aA, c\},$$

obtaining a system in weak Greibach Normal Form. This amounts to converting the matrix

$$H = \begin{pmatrix} aB & \emptyset \\ B & \{d, Aa\} \end{pmatrix}$$

to the matrix

$$L = \begin{pmatrix} aB & \emptyset \\ \{bY_2, aAY_4, cY_4, aA, c\} & \{d, bY_1a, aAY_3a, cY_3a, ba\} \end{pmatrix}$$

The weak Greibach Normal Form corresponds to the new system

$$X = KY + K,$$
$$Y = LY + L.$$

This method works in general for any input grammar with no ϵ-rules, no chain rules, and such that every nonterminal belongs to $T(G)$. Under these conditions, the row vector K contains some nonempty entry, all strings in K are in ΣV^*, and all strings in H are in V^+. After obtaining the grammar GH defined by the system

$$X = KY + K,$$
$$Y = HY + H,$$

we use the system $X = KY + K$ to express every nonterminal A_i in terms of expressions containing strings $\alpha_{i,j}$ involving a terminal as the leftmost symbol ($\alpha_{i,j} \in \Sigma V^*$), and we replace all leftmost occurrences of nonterminals in H (occurrences A_i in strings of the form $A_i\beta$, where $\beta \in V^*$) using the above expressions. In this fashion, we obtain a matrix L, and it is immediately shown that the system

$$X = KY + K,$$
$$Y = LY + L,$$

generates the same tuple of languages. Furthermore, this last system corresponds to a weak Greibach Normal Form.

It we start with a grammar in Chomsky Normal Form (with no production $S \to \epsilon$) such that every nonterminal belongs to $T(G)$, we actually get a Greibach Normal Form (the entries in K are terminals, and the entries in H are nonterminals). Thus, we have justified lemma 3.6.1. The method is also quite economical, since it introduces only m^2 new nonterminals. However, the resulting grammar may contain some useless nonterminals.

3.10 Tree Domains and Gorn Trees

Derivation trees play a very important role in parsing theory and in the proof of a strong version of the pumping lemma for the context-free languages known as Ogden's lemma. Thus, it is important to define derivation trees rigorously. We do so using Gorn trees.

Let $\mathbf{N}_+ = \{1, 2, 3, \ldots\}$.

Definition 3.10.1 A *tree domain* D is a nonempty subset of strings in $\mathbf{N}_+^*$ satisfying the conditions:

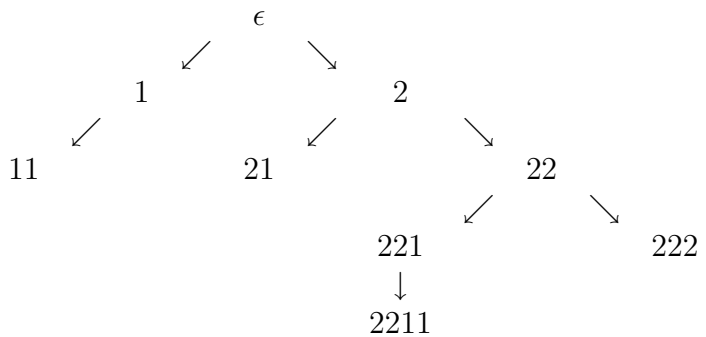
(1) For all $u, v \in \mathbf{N}_+^*$, if $uv \in D$, then $u \in D$.

(2) For all $u \in \mathbf{N}_+^*$, for every $i \in \mathbf{N}_+$, if $ui \in D$ then $uj \in D$ for every j, $1 \leq j \leq i$.

The tree domain

$$D = \{\epsilon, 1, 2, 11, 21, 22, 221, 222, 2211\}$$

is represented as follows:



A tree labeled with symbols from a set Δ is defined as follows.

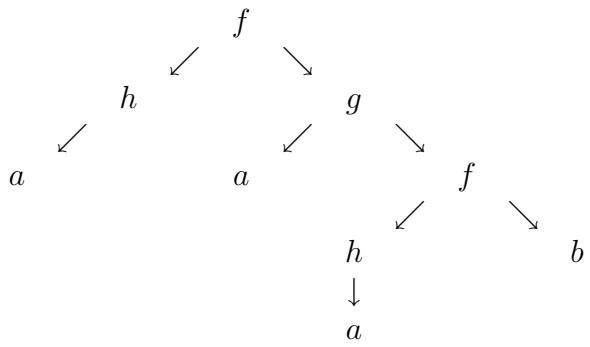
Definition 3.10.2 Given a set Δ of labels, a Δ-tree (for short, a *tree*) is a total function $t : D \to \Delta$, where D is a tree domain.

The domain of a tree t is denoted as $dom(t)$. Every string $u \in dom(t)$ is called a *tree address* or a *node*.

Let $\Delta = \{f, g, h, a, b\}$. The tree $t : D \to \Delta$, where D is the tree domain of the previous example and t is the function whose graph is

$$\{(\epsilon, f), (1, h), (2, g), (11, a), (21, a), (22, f), (221, h), (222, b), (2211, a)\}$$

is represented as follows:



The *outdegree* (sometimes called *ramification*) $r(u)$ of a node u is the cardinality of the set

$$\{i \mid ui \in dom(t)\}.$$

Note that the outdegree of a node can be infinite. Most of the trees that we shall consider will be *finite-branching*, that is, for every node u, $r(u)$ will be an integer, and hence finite. If the outdegree of all nodes in a tree is bounded by n, then we can view the domain of the tree as being defined over $\{1, 2, \ldots, n\}^*$.

A node of outdegree 0 is called a *leaf*. The node whose address is ϵ is called the *root* of the tree. A tree is *finite* if its domain $dom(t)$ is finite. Given a node u in $dom(t)$, every node of the form ui in $dom(t)$ with $i \in \mathbf{N}_+$ is called a *son* (or *immediate successor*) of u.

Tree addresses are totally ordered *lexicographically*: $u \leq v$ if either u is a prefix of v or, there exist strings $x, y, z \in \mathbf{N}_+^*$ and $i, j \in \mathbf{N}_+$, with $i < j$, such that $u = xiy$ and $v = xjz$.

In the first case, we say that u is an *ancestor* (or *predecessor*) of v (or u *dominates* v) and in the second case, that u is *to the left* of v.

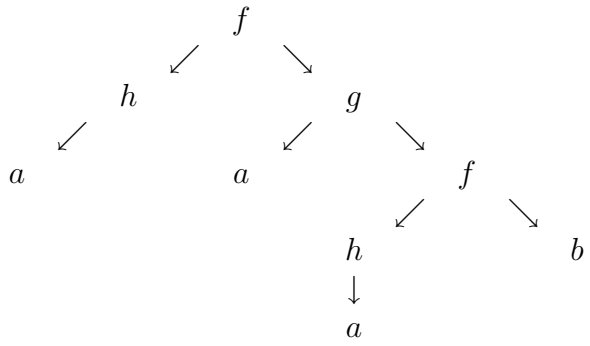
If $y = \epsilon$ and $z = \epsilon$, we say that xi is a *left brother* (or *left sibling*) of xj, $(i < j)$. Two tree addresses u and v are *independent* if u is not a prefix of v and v is not a prefix of u.

Given a finite tree t, the *yield* of t is the string

$$t(u_1)t(u_2)\cdots t(u_k),$$

where $u_1, u_2, \ldots, u_k$ is the sequence of leaves of t in lexicographic order.

For example, the yield of the tree below is *aaab*:



Given a finite tree t, the *depth* of t is the integer

$$d(t) = \max\{|u| \mid u \in dom(t)\}.$$

Given a tree t and a node u in $dom(t)$, the *subtree rooted at* u is the tree t/u, whose domain is the set

$$\{v \mid uv \in dom(t)\}$$

and such that $t/u(v) = t(uv)$ for all v in $dom(t/u)$.

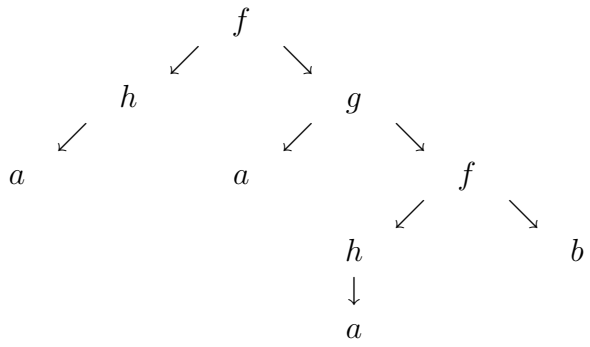
Another important operation is the operation of tree replacement (or tree substitution).

Definition 3.10.3 Given two trees t_1 and t_2 and a tree address u in t_1, the *result of replacing* t_2 at u in t_1, denoted by $t_1[u \leftarrow t_2]$, is the function whose graph is the set of pairs

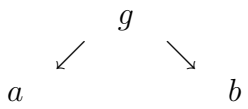
$$\{(v, t_1(v)) \mid v \in dom(t_1),\ u \text{ is not a prefix of } v\} \cup \{(uv, t_2(v)) \mid v \in dom(t_2)\}.$$

Let t_1 and t_2 be the trees defined by the following diagrams:

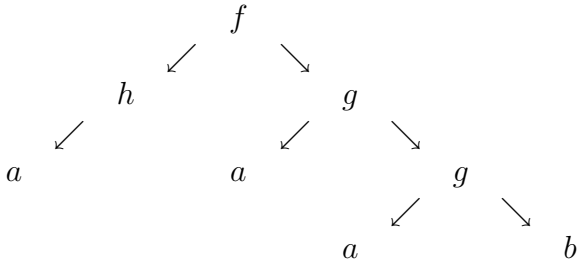
Tree t_1



Tree t_2



The tree $t_1[22 \leftarrow t_2]$ is defined by the following diagram:



We can now define derivation trees and relate derivations to derivation trees.

3.11 Derivations Trees

Definition 3.11.1 Given a context-free grammar $G = (V, \Sigma, P, S)$, for any $A \in N$, an *A-derivation tree for G* is a $(V \cup \{\epsilon\})$-tree t such that:

(1) $t(\epsilon) = A$;

(2) For every nonleaf node $u \in dom(t)$, if $u1, \ldots, uk$ are the successors of u, then eiher there is a production $B \to X_1 \cdots X_k$ in P such that $t(u) = B$ and $t(ui) = X_i$ for all i, $1 \leq i \leq k$, or $B \to \epsilon \in P$, $t(u) = B$ and $t(u1) = \epsilon$. A *complete derivation (or parse tree)* is an S-tree whose yield belongs to Σ^*.

A derivation tree for the grammar

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

where P is the set of rules

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

is shown below. The yield of the derivation tree is $a + a * a$.

(* Picture of tree goes here *)

Derivations trees are associated to derivations inductively as follows.

Definition 3.11.2 Given a context-free grammar $G = (V, \Sigma, P, S)$, for any $A \in N$, if $\pi : A \overset{n}{\Longrightarrow} \alpha$ is a derivation in G, we construct an *A-derivation tree t_π with yield α* as follows.

(1) If $n = 0$, then t_π is the one-node tree such that $dom(t_\pi) = \{\epsilon\}$ and $t_\pi(\epsilon) = A$.

(2) If $A \overset{n-1}{\Longrightarrow} \lambda B \rho \Longrightarrow \lambda\gamma\rho = \alpha$, then if t_1 is the A-derivation tree with yield $\lambda B\rho$ associated with the derivation $A \overset{n-1}{\Longrightarrow} \lambda B\rho$, and if t_2 is the tree associated with the production $B \to \gamma$ (that is, if
$$\gamma = X_1 \cdots X_k,$$
then $dom(t_2) = \{\epsilon, 1, \ldots, k\}$, $t_2(\epsilon) = B$, and $t_2(i) = X_i$ for all i, $1 \leq i \leq k$, or if $\gamma = \epsilon$, then $dom(t_2) = \{\epsilon, 1\}$, $t_2(\epsilon) = B$, and $t_2(1) = \epsilon$), then
$$t_\pi = t_1[u \leftarrow t_2],$$
where u is the address of the leaf labeled B in t_1.

The tree t_π is the *A-derivation tree associated with the derivation $A \overset{n}{\Longrightarrow} \alpha$.*

Given 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 parse trees associated with two derivations of the string $a + a * a$ are shown below.

(* insert picture of trees here *)

The following lemma is easily shown.

Lemma 3.11.3 *Let $G = (V, \Sigma, P, S)$ be a context-free grammar. For any derivation $A \overset{n}{\Longrightarrow} \alpha$, there is a unique A-derivation tree associated with this derivation, with yield α. Conversely, for any A-derivation tree t with yield α, there is a unique leftmost derivation $A \overset{*}{\underset{lm}{\Longrightarrow}} \alpha$ in G having t as its associated derivation tree.*

We will now prove a strong version of the pumping lemma for context-free languages due to Bill Ogden (1968).

3.12 Ogden's Lemma

Ogden's lemma states some combinatorial properties of parse trees that are deep enough. The yield w of such a parse tree can be split into 5 substrings u, v, x, y, z such that

$$w = uvxyz,$$

where u, v, x, y, z satisfy certain conditions. It turns out that we get a more powerful version of the lemma if we allow ourselves to *mark* certain occurrences of symbols in w before invoking the lemma. We can imagine that *marked occurrences* in a nonempty string w are occurrences of symbols in w in boldface, or red, or any given color (but one color only). For example, given $w = aaababbbaa$, we can mark the symbols of even index as follows:

$$a\mathbf{a}a\mathbf{b}a\mathbf{b}b\mathbf{b}a\mathbf{a}.$$

More rigorously, we can define a *marking* of a nonnull string $w\colon \{1, \ldots, n\} \to \Sigma$ as any function $m\colon \{1, \ldots, n\} \to \{0, 1\}$. Then, a letter w_i in w is a *marked occurrence* iff $m(i) = 1$, and an *unmarked occurrence* if $m(i) = 0$. The number of marked occurrences in w is equal to

$$\sum_{i=1}^{n} m(i).$$

Ogden's lemma only yields useful information for grammars G generating an infinite language. We could make this hypothesis, but it seems more elegant to use the precondition that the lemma only applies to strings $w \in L(D)$ such that w contains at least K marked occurrences, for a constant K large enough. If K is large enough, $L(G)$ will indeed be infinite.

Lemma 3.12.1 *For every context-free grammar G, there is some integer $K > 1$ such that, for every string $w \in \Sigma^+$, for every marking of w, if $w \in L(G)$ and w contains at least K marked occurrences, then there exists some decomposition of w as $w = uvxyz$, and some $A \in N$, such that the following properties hold:*

(1) There are derivations $S \overset{+}{\Longrightarrow} uAz$, $A \overset{+}{\Longrightarrow} vAy$, and $A \overset{+}{\Longrightarrow} x$, so that

$$uv^n xy^n z \in L(G)$$

for all $n \geq 0$ (the pumping property);

(2) x contains some marked occurrence;

(3) Either (both u and v contain some marked occurrence), or (both y and z contain some marked occurrence);

(4) vxy contains less than K marked occurrences.

Proof. Let t be any parse tree for w. We call a leaf of t a *marked leaf* if its label is a marked occurrence in the marked string w. The general idea is to make sure that K is large enough so that parse trees with yield w contain enough repeated nonterminals along some path from the root to some marked leaf. Let $r = |N|$, and let

$$p = \max\{2,\ \max\{|\alpha| \mid (A \to \alpha) \in P\}\}.$$

We claim that $K = p^{2r+3}$ does the job.

The key concept in the proof is the notion of a B-node. Given a parse tree t, a *B-node* is a node with at least two immediate successors u_1, u_2, such that for $i = 1, 2$, either u_i is a marked leaf, or u_i has some marked leaf as a descendant. We construct a path from the root to some marked leaf, so that for every B-node, we pick the leftmost successor with the maximum number of marked leaves as descendants. Formally, define a path $(s_0, \ldots, s_n)$ from the root to some marked leaf, so that:

(i) Every node s_i has some marked leaf as a descendant, and s_0 is the root of t;

(ii) If s_j is in the path, s_j is not a leaf, and s_j has a single immediate descendant which is either a marked leaf or has marked leaves as its descendants, let s_{j+1} be that unique immediate descendant of s_i.

(iii) If s_j is a B-node in the path, then let s_{j+1} be the leftmost immediate successors of s_j with the maximum number of marked leaves as descendants (assuming that if s_{j+1} is a marked leaf, then it is its own descendant).

(iv) If s_j is a leaf, then it is a marked leaf and $n = j$.

We will show that the path $(s_0, \ldots, s_n)$ contains at least $2r + 3$ B-nodes.

Claim: For every i, $0 \leq i \leq n$, if the path $(s_i, \ldots, s_n)$ contains b B-nodes, then s_i has at most p^b marked leaves as descendants.

Proof. We proceed by "backward induction", i.e., by induction on $n - i$. For $i = n$, there are no B-nodes, so that $b = 0$, and there is indeed $p^0 = 1$ marked leaf s_n. Assume that the claim holds for the path $(s_{i+1}, \ldots, s_n)$.

If s_i is not a B-node, then the number b of B-nodes in the path $(s_{i+1}, \ldots, s_n)$ is the same as the number of B-nodes in the path $(s_i, \ldots, s_n)$, and s_{i+1} is the only immediate successor of s_i having a marked leaf as descendant. By the induction hypothesis, s_{i+1} has at most p^b marked leaves as descendants, and this is also an upper bound on the number of marked leaves which are descendants of s_i.

If s_i is a B-node, then if there are b B-nodes in the path $(s_{i+1}, \ldots, s_n)$, there are $b + 1$ B-nodes in the path $(s_i, \ldots, s_n)$. By the induction hypothesis, s_{i+1} has at most p^b marked leaves as descendants. Since s_i is a B-node, s_{i+1} was chosen to be the leftmost immediate successor of s_i having the maximum number of marked leaves as descendants. Thus, since

the outdegree of s_i is at most p, and each of its immediate successors has at most p^b marked leaves as descendants, the node s_i has at most $pp^d = p^{d+1}$ marked leaves as descendants, as desired. □

Applying the claim to s_0, since w has at least $K = p^{2r+3}$ marked occurrences, we have $p^b \geq p^{2r+3}$, and since $p \geq 2$, we have $b \geq 2r+3$, and the path $(s_0, \ldots, s_n)$ contains at least $2r+3$ B-nodes (Note that this would not follow if we had $p = 1$).

Let us now select the lowest $2r+3$ B-nodes in the path, $(s_0, \ldots, s_n)$, and denote them $(b_1, \ldots, b_{2r+3})$. Every B-node b_i has at least two immediate successors $u_i < v_i$ such that u_i or v_i is on the path $(s_0, \ldots, s_n)$. If the path goes through u_i, we say that b_i is a *right B-node* and if the path goes through v_i, we say that b_i is a *left B-node*. Since $2r+3 = r+2+r+1$, either there are $r+2$ left B-nodes or there are $r+2$ right B-nodes in the path $(b_1, \ldots, b_{2r+3})$. Let us assume that there are $r+2$ left B-nodes, the other case being similar.

Let $(d_1, \ldots, d_{r+2})$ be the lowest $r+2$ left B-nodes in the path. Since there are $r+1$ B-nodes in the sequence $(d_2, \ldots, d_{r+2})$, and there are only r distinct nonterminals, there are two nodes d_i and d_j, with $2 \leq i < j \leq r+2$, such that $t(d_i) = t(d_j) = A$, for some $A \in N$. We can assume that d_i is an ancestor of d_j, and thus, $d_j = d_i\alpha$, for some $\alpha \neq \epsilon$.

If we prune out the subtree t/d_i rooted at d_i from t, we get an S-derivation tree having a yield of the form uAz, and we have a derivation of the form $S \stackrel{+}{\Longrightarrow} uAz$, since there are at least $r+2$ left B-nodes on the path, and we are looking at the lowest $r+1$ left B-nodes. Considering the subtree t/d_i, pruning out the subtree t/d_j rooted at α in t/d_i, we get an A-derivation tree having a yield of the form vAy, and we have a derivation of the form $A \stackrel{+}{\Longrightarrow} vAy$. Finally, the subtree t/d_j is an A-derivation tree with yield x, and we have a derivation $A \stackrel{+}{\Longrightarrow} x$. This proves (1) of the lemma.

Since s_n is a marked leaf and a descendant of d_j, x contains some marked occurrence, proving (2).

Since d_1 is a left B-node, some left sibbling of the immediate successor of d_1 on the path has some distinguished leaf in u as a descendant. Similarly, since d_i is a left B-node, some left sibbling of the immediate successor of d_i on the path has some distinguished leaf in v as a descendant. This proves (3).

$(d_j, \ldots, b_{2r+3})$ has at most $2r+1$ B-nodes, and by the claim shown earlier, d_j has at most p^{2r+1} marked leaves as descendants. Since $p^{2r+1} < p^{2r+3} = K$, this proves (4). □

Observe that condition (2) implies that $x \neq \epsilon$, and condition (3) implies that either $u \neq \epsilon$ and $v \neq \epsilon$, or $y \neq \epsilon$ and $z \neq \epsilon$. Thus, the pumping condition (1) implies that the set $\{uv^nxy^nz \mid n \geq 0\}$ is an infinite subset of $L(G)$, and $L(G)$ is indeed infinite, as we mentioned earlier. Note that $K \geq 3$, and in fact, $K \geq 32$. The "standard pumping lemma" due to Bar-Hillel, Perles, and Shamir, is obtained by letting all occurrences be marked in $w \in L(G)$.

Lemma 3.12.2 *For every context-free grammar G (without ϵ-rules), there is some integer $K > 1$ such that, for every string $w \in \Sigma^+$, if $w \in L(G)$ and $|w| \geq K$, then there exists some decomposition of w as $w = uvxyz$, and some $A \in N$, such that the following properties hold:*

(1) There are derivations $S \overset{+}{\Longrightarrow} uAz$, $A \overset{+}{\Longrightarrow} vAy$, *and* $A \overset{+}{\Longrightarrow} x$, *so that*

$$uv^n xy^n z \in L(G)$$

for all $n \geq 0$ *(the pumping property);*

(2) $x \neq \epsilon$;

(3) Either $v \neq \epsilon$ *or* $y \neq \epsilon$;

(4) $|vxy| \leq K$.

A stronger version could be stated, and we are just following tradition in stating this standard version of the pumping lemma.

Ogden's lemma or the pumping lemma can be used to show that certain languages are not context-free. The method is to proceed by contradiction, i.e., to assume (contrary to what we wish to prove) that a language L is indeed context-free, and derive a contradiction of Ogden's lemma (or of the pumping lemma). Thus, as in the case of the regular languages, it would be helpful to see what the negation of Ogden's lemma is, and for this, we first state Ogden's lemma as a logical formula.

For any nonnull string $w\colon \{1, \ldots, n\} \to \Sigma$, for any marking $m\colon \{1, \ldots, n\} \to \{0, 1\}$ of w, for any substring y of w, where $w = xyz$, with $|x| = h$ and $k = |y|$, the number of marked occurrences in y, denoted as $|m(y)|$, is defined as

$$|m(y)| = \sum_{i=h+1}^{i=h+k} m(i).$$

We will also use the following abbreviations:

$$\begin{aligned}
nat &= \{0, 1, 2, \ldots\},\\
nat32 &= \{32, 33, \ldots\},\\
A &\equiv w = uvxyz,\\
B &\equiv |m(x)| \geq 1,\\
C &\equiv (|m(u)| \geq 1 \wedge |m(v)| \geq 1) \vee (|m(y)| \geq 1 \wedge |m(z)| \geq 1),\\
D &\equiv |m(vxy)| < K,\\
P &\equiv \forall n\colon nat\ (uv^n xy^n z \in L(D)).
\end{aligned}$$

Ogden's lemma can then be stated as

$$\forall G\colon \mathrm{CFG}\ \exists K\colon nat32\ \forall w\colon \Sigma^*\ \forall m\colon \mathrm{marking}$$
$$\Big((w \in L(D) \wedge |m(w)| \geq K) \supset (\exists u, v, x, y, z\colon \Sigma^*\ A \wedge B \wedge C \wedge D \wedge P)\Big).$$

Recalling that

$$\neg(A \wedge B \wedge C \wedge D \wedge P) \equiv \neg(A \wedge B \wedge C \wedge D) \vee \neg P \equiv (A \wedge B \wedge C \wedge D) \supset \neg P$$

and

$$\neg(P \supset Q) \equiv P \wedge \neg Q,$$

the negation of Ogden's lemma can be stated as

$$\exists G\colon \mathrm{CFG}\ \forall K\colon nat32\ \exists w\colon \Sigma^*\ \exists m\colon \mathrm{marking}$$
$$\Big((w \in L(D) \wedge |m(w)| \geq K) \wedge (\forall u, v, x, y, z\colon \Sigma^*\ (A \wedge B \wedge C \wedge D) \supset \neg P)\Big).$$

Since

$$\neg P \equiv \exists n\colon nat\ (uv^n xy^n z \notin L(D)),$$

in order to show that Ogden's lemma is contradicted, one needs to show that for some context-free grammar G, for every $K \geq 2$, there is some string $w \in L(D)$ and some marking m of w with at least K marked occurrences in w, such that for every possible decomposition $w = uvxyz$ satisfying the constraints $A \wedge B \wedge C \wedge D$, there is some $n \geq 0$ such that $uv^n xy^n z \notin L(D)$. When proceeding by contradiction, we have a language L that we are (wrongly) assuming to be context-free and we can use any CFG grammar G generating L. The creative part of the argument is to pick the right $w \in L$ and the right marking of w (not making any assumption on K).

As an illustration, we show that the language

$$L = \{a^n b^n c^n \mid n \geq 1\}$$

is not context-free. Since L is infinite, we will be able to use the pumping lemma.

The proof proceeds by contradiction. If L was context-free, there would be some context-free grammar G such that $L = L(G)$, and some constant $K > 1$ as in Ogden's lemma. Let $w = a^K b^K c^K$, and choose the $b's$ as marked occurrences. Then by Ogden's lemma, x contains some marked occurrence, and either both u, v or both y, z contain some marked occurrence. Assume that both u and v contain some b. We have the following situation:

$$\underbrace{a \cdots a b \cdots b}_{u}\underbrace{b \cdots b}_{v}\underbrace{b \cdots b c \cdots c}_{xyz}.$$

If we consider the string $uvvxyyz$, the number of a's is still K, but the number of b's is strictly greater than K since v contains at least one b, and thus $uvvxyyz \notin L$, a contradiction.

If both y and z contain some b we will also reach a contradiction because in the string $uvvxyyz$, the number of c's is still K, but the number of b's is strictly greater than K. Having reached a contradiction in all cases, we conclude that L is not context-free.

Let us now show that the language

$$L = \{a^m b^n c^m d^n \mid m, n \geq 1\}$$

is not context-free.

Again, we proceed by contradiction. This time, let

$$w = a^K b^K c^K d^K,$$

where the b's and c's are marked occurrences.

By Ogden's lemma, either both u, v contain some marked occurrence, or both y, z contain some marked occurrence, and x contains some marked occurrence. Let us first consider the case where both u, v contain some marked occurrence.

If v contains some b, since $uvvxyyz \in L$, v must contain only b's, since otherwise we would have a bad string in L, and we have the following situation:

$$\underbrace{a \cdots ab \cdots b}_{u}\underbrace{b \cdots b}_{v}\underbrace{b \cdots bc \cdots cd \cdots d}_{xyz}.$$

Since $uvvxyyz \in L$, the only way to preserve an equal number of b's and d's is to have $y \in d^+$. But then, vxy contains c^K, which contradicts (4) of Ogden's lemma.

If v contains some c, since x also contains some marked occurrence, it must be some c, and v contains only c's and we have the following situation:

$$\underbrace{a \cdots ab \cdots bc \cdots c}_{u}\underbrace{c \cdots c}_{v}\underbrace{c \cdots cd \cdots d}_{xyz}.$$

Since $uvvxyyz \in L$ and the number of a's is still K whereas the number of c's is strictly more than K, this case is impossible.

Let us now consider the case where both y, z contain some marked occurrence. Reasoning as before, the only possibility is that $v \in a^+$ and $y \in c^+$:

$$\underbrace{a \cdots a}_{u}\underbrace{a \cdots a}_{v}\underbrace{a \cdots ab \cdots bc \cdots c}_{x}\underbrace{c \cdots c}_{y}\underbrace{c \cdots cd \cdots d}_{z}.$$

But then, vxy contains b^K, which contradicts (4) of Ogden's Lemma. Since a contradiction was obtained in all cases, L is not context-free.

Ogden's lemma can also be used to show that the context-free language

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

is inherently ambiguous. The proof is quite involved.

Another corollary of the pumping lemma is that it is decidable whether a context-free grammar generates an infinite language.

Lemma 3.12.3 *Given any context-free grammar G (without ϵ-rules), there is some $K > 1$ with the following property:*

$L(G)$ is infinite iff there is some $w \in L(G)$ such that $K \leq |w| < 2K$.

Proof. Let $K = p^{2r+3}$ be the constant from the proof of lemma 3.12.1. If there is some $w \in L(G)$ such that $|w| \geq K$, we already observed that the pumping lemma implies that $L(G)$ contains an infinite subset of the form $\{uv^n xy^n z \mid n \geq 0\}$. Conversely, assume that $L(G)$ is infinite. If $|w| < K$ for all $w \in L(G)$, then $L(G)$ is finite. Thus, there is some $w \in L(G)$ such that $|w| \geq K$. Let $w \in L(G)$ be a minimal string such that $|w| \geq K$. By the pumping lemma, we can write w as $w = uvxyxz$, where $x \neq \epsilon$, $vy \neq \epsilon$, and $|vxy| \leq K$. By the pumping property, $uxz \in L(G)$. If $|w| \geq 2K$, then

$$|uxz| = |uvxyz| - |vy| > |uvxyz| - |vxy| \geq 2K - K = K,$$

and $|uxz| < |uvxyz|$, contradicting the minimality of w. Thus, we must have $|w| < 2K$. □

In particular, if G is in Chomsky Normal Form, it can be shown that we just have to consider derivations of length at most $4K - 3$.