TOC

UNIT-2

Vision and Mission of Institute

Vision:

To become a renowned centre of outcome based learning, and work towards academic, professional, cultural and social enrichment of the lives of individuals and communities.

Mission:

- Focus on evaluation of learning outcomes and motivate students to inculcate research aptitude by project based learning.

- Identify areas of focus and provide platform to gain knowledge and solutions based on informed perception of Indian, regional and global needs.

- Offer opportunities for interaction between academia and industry.

- Develop human potential to its fullest extent so that intellectually capable and imaginatively gifted leaders can emerge in a range of professions.

Vision and Mission of Department of Information Technology

Vision:

To establish outcome based excellence in teaching, learning and commitment to support IT Industry.

Mission:

M1: *To provide outcome based education.*

M2: *To provide fundamental & Intellectual knowledge with essential skills to meet current and future need of IT Industry across the globe.*

M3: *To inculcate the philosophy of continuous learning, ethical values & Social Responsibility.*

Syllabus:

RAJASTHAN TECHNICAL UNIVERSITY, KOTA

SYLLABUS

II Year- IV Semester: B.Tech. (Information Technology)

4IT4-06: Theory of Computation

Credit: 3 **Max. Marks: 150(IA:30, ETE:120)**

3L+0T+0P **End Term Exam: 3 Hours**

SN	Contents	Hours
1	**Introduction:** Objective, scope and outcome of the course.	1
2	Finite Automata & Regular Expression: Basic machine, Finite state machine, Transition graph, Transition matrix, Deterministic and non-deterministic finite automation, Equivalence of DFA and NDFA, Decision properties, minimization of finite automata, Mealy & Moore machines. Alphabet, words, Operations, Regular sets, relationship and conversion between Finite automata and regular expression and vice versa, designing regular expressions, closure properties of regular sets, Pumping lemma and regular sets, Myhill- Nerode theorem , Application of pumping lemma, Power of the languages.	7
3	Context Free Grammars (CFG), Derivations and Languages, Relationship between derivation and derivation trees, leftmost and rightmost derivation, sentential forms, parsing and ambiguity, simplification of CFG, normal forms, Greibach and Chomsky Normal form , Problems related to CNF and GNF including membership problem.	8
4	Nondeterministic PDA, Definitions, PDA and CFL, CFG for PDA, Deterministic PDA, and Deterministic PDA and Deterministic CFL , The pumping lemma for CFL's, Closure Properties and Decision properties for CFL, Deciding properties of CFL.	8
5	Turing Machines: Introduction, Definition of Turing Machine, TM language Acceptors and Transducers, Computable Languages and functi Universal TM & Other modification, multiple tracks Turing Machine. Hierarchy of Formal languages: Recursive & recursively enumerable languages, Properties of RL and REL, Introduction of Context sensitive grammers and languages, The Chomsky Hierarchy.	8
6	Tractable and Untractable Problems: P, NP, NP complete and NP hard problems, Un-decidability, examples of these problems like vertex cover problem, Hamiltonian path problem, traveling sales man problem.	8
	Total	40

Program Educational Objectives (PEO):

1. To enrich students with fundamental knowledge, effective computing, problem solving and communication skills enable them to have successful career in Information Technology.
2. To enable students in acquiring Information Technology's latest tools, technologies and management principles to give them an ability to solve multidisciplinary engineering problems.
3. To impart students with ethical values and commitment towards sustainable development in collaborative mode.
4. To imbibe students with research oriented and innovative approaches which help them to identify, analyze, formulate and solve real life problems and motivates them for lifelong learning.
5. To empower students with leadership quality and team building skills that prepare them for employment, entrepreneurship and to become competent professionals to serve societies and global needs.

Program Outcomes (PO):

1. **Engineering Knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems in IT.
2. **Problem analysis:** Identify, formulate, research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences in IT.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations using IT.
4. **Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions using IT.
5. **Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations in IT.
6. **The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice using IT.
7. **Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development in IT.
8. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice using IT.
9. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings in IT.
10. **Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project Management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage IT projects and in multidisciplinary environments.
12. **Life –long Learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological changes needed in IT.

4IT4-06: Theory of Computation

Course Outcomes (COs)

CO1- Able to design and understand and basic properties of DFA & NDFA and formal languages and formal grammars.

CO2- Able to understand the relation between types of languages and types of finite automata and the Context free languages and grammar's, and also Normalizing CFG

CO3- Able to design & understand the minimization of deterministic and nondeterministic finite automata & the concept of Pushdown automata and its application.

CO4- Able to understand basic properties of Turing machines and computing with Turing machines and concepts of tractability and decidability, the concepts of NP-completeness and NP-hard Problem, the challenges for Theoretical Computer Science and its contribution to other sciences.

CO-PO Mapping:

H=3, M=2, L=1

			PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO12	PO12
V Sem.	4IT4-06: Theory of Computation	CO-1	H	M	H	H	M	M	M	-	M	M	L	M
		CO-2	H	H	H	H	M	M	M	-	M	M	L	M
		CO-3	H	M	H	H	M	M	M	-	M	M	L	M
		CO-4	H	M	H	H	M	M	M	-	M	M	M	M

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

6.1 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 6.1. 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.

6.2 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 6.2. 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 6.3. Given a context-free grammar $G = (V, \Sigma, P, S)$, the *language generated by* G is the set

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

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 6.4. 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

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

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 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 proposition 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.

Proposition 6.1. *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. □

Proposition 6.1 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 6.5. 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.

6.3 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 proposition.

Proposition 6.2. *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) 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 $A \stackrel{1}{\Longrightarrow} \epsilon$, 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 proposition are now met. □

From a practical point of view, the construction or Proposition 6.2 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 Proposition 6.2 to the input grammar G, and we explain the construction assuming that $G = (V, \Sigma, P, S)$ satisfies the conditions of Proposition 6.2. 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:

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

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 proposition.

Proposition 6.3. *Given a context-free grammar $G = (V, \Sigma, P, S)$, one can construct a context-free grammar $G' = (V', \Sigma, P', S')$ such that:*

(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 Proposition 6.2 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 proposition. 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 $L(G') = L(G_1) = L(G)$. □

Let us apply the method of Proposition 6.3 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 Proposition 6.3 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.

Proposition 6.4. *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 Proposition 6.3, 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 Proposition 6.4 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

6.4 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 6.6. 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 proposition shows the equivalence between NFA's and right-linear grammars.

Proposition 6.5. *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 nonterminal 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: $Q = N' = N \cup \{E\}$, $q_0 = S$, $F = \{A \in N' \mid A \to \epsilon\}$, 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 proposition 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 $u \in \Sigma^*$.

6.5 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 $E \overset{+}{\Longrightarrow} \alpha$ 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

6.6 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 $A \overset{+}{\Longrightarrow} A\alpha$ 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 6.9.

Proposition 6.6. *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*

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

6.11 Derivations Trees

Definition 6.14. 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 (a tree with set of labels $(V \cup \{\epsilon\})$) 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 either 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 in Figure 6.1. The yield of the derivation tree is $a + a * a$.

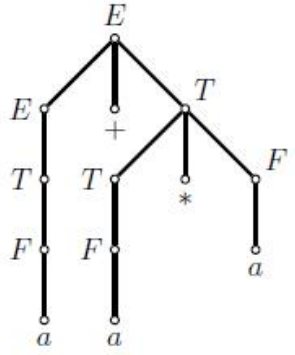

Figure 6.1: A complete derivation tree

Derivations trees are associated to derivations inductively as follows.

Definition 6.15. 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 in Figure 6.2:

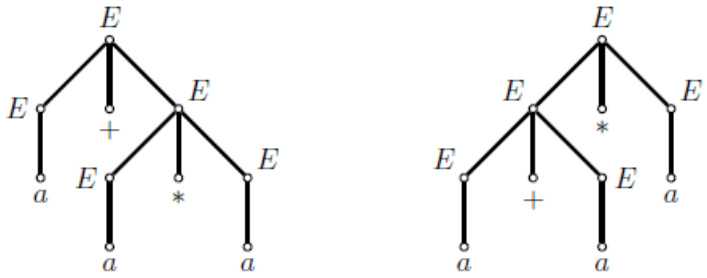

Figure 6.2: Two derivation trees for $a + a * a$

The following proposition is easily shown.

Proposition 6.11. *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).

6.12 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:

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

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.

Proposition 6.12. *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. $\square$

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). $\square$

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 $w \in L(G)$.