Chapter 2

Ordinary grammars

2.1 Definitions by rewriting, by deduction, by parse trees and by equations

Definition of strings with well-nested brackets. Let a left bracket be denoted by a, and a right bracket by b. A formal language over the alphabet $\Sigma = \{a, b\}$.

- ε is a well-nested string;
- if u is a well-nested string, then so is aub;
- if u and v are well-nested strings, then so is uv.

Equivalent reformulation: w is a well-nested string if and only if

- $w = \varepsilon$, or
- $w = aub$ for some well-nested string u, or
- $w = uv$ for some well-nested strings u and v.

Let S be an abstract symbol that denotes a well-nested string. In the notation of formal grammars, the above definition is written as follows.

$$S \to \varepsilon \mid aSb \mid SS$$

Here the vertical line separates alternative forms of S, and is therefore a logical disjunction. Such a definition is called a formal grammar.

Definition 2.1. *An* ordinary formal grammar *is a quadruple $G = (\Sigma, N, R, S)$, in which:*

- Σ *is the* alphabet *of the language being defined, that is, a finite set of* symbols*, from which the strings in the language are built;*
- N *is a finite set of* category symbols *representing the syntactic categories defined in the grammar. Each of these symbols denotes a certain property that every string over Σ is deemed to have or not to have. (also called syntactic types, nonterminal symbols, variables or predicate symbols, depending on the outlook on grammars);*
- R *is a finite set of* grammar rules*, each reprenenting a possible structure of strings with the property $A \in N$ as a concatenation $X_1 \dots X_\ell$ of zero or more symbols symbols $X_1, \dots, X_\ell \in \Sigma \cup N$, with $\ell \geqslant 0$.*

$$A \to X_1 \dots X_\ell \tag{2.1}$$

- *$S \in N$ is a distinguished category symbol representing all well-formed sentences defined by the grammar (the letter S stands for "sentence", also occasionally referred as an "initial symbol" or a "start symbol").*

A rule $A \to X_1 \dots X_n$ states that if a string w is representable as a concatenation $w_1 \dots w_n$ of n substrings, where each i-th substring has the property X_i, then w has the property A.

If $A \to \alpha_1, \dots, A \to \alpha_m$ are all rules for a symbol $A \in N$, these rules may be written as

$$A \to \alpha_1 \mid \dots \mid \alpha_m,$$

in which the vertical lines separating the alternatives are, in essense, disjunction operators.

This intuitive meaning of a grammar can be formalized in three ways. One definition employs string rewriting: this definition, popularized by Chomsky [3], is the most well-known. Another definition uses deduction on items of the form "w has the property A", and one more definition is by a solution of language equations.

Consider the above grammar for well-nested strings, written in formal notation. The language of all well-nested strings is known as the *Dyck language*, after the German mathematician Walther von Dyck.

Example 2.1. *The Dyck language is generated by a grammar $G = (\Sigma, N, R, S)$, where $\Sigma = \{a, b\}$, $N = \{S\}$ and $R = \{S \to aSb,\ S \to SS,\ S \to \varepsilon\}$. A grammar such as this shall be written as follows.*

$$S \to aSb \mid SS \mid \varepsilon$$

2.1.1 Definition by string rewriting

One approach to defining the meaning of a grammar is by rewriting the so-called *sentential forms*, which are strings over a combined alphabet $\Sigma \cup N$ containing both symbols of the target language and category symbols. A sentential form serves as a scheme of a sentence, in which every occurrence of a category symbol $A \in N$ stands for *some string with the property A.* At each step of rewriting, some category symbol A is replaced by the right-hand side of some rule for A, thus obtaining a more precise sentential form. The string rewriting begins by taking the initial symbol S (that is, the least precise sentential form) and proceeds until only symbols of the alphabet remain (that is, an actual sentence of the language is obtained).

Definition 2.1(R) (Chomsky [3])**.** *Let $G = (\Sigma, N, R, S)$ be a grammar. Define a relation $\Longrightarrow$ of one-step derivability on $(\Sigma \cup N)^*$ as follows.*

$$sAs' \Longrightarrow s\alpha s' \quad (\textit{for all } A \to \alpha \in R \textit{ and } s, s' \in (\Sigma \cup N)^*)$$

The relations of reachability in zero or more steps, in one or more steps, in exactly ℓ steps and in at most ℓ steps are denoted by $\Longrightarrow^*$, $\Longrightarrow^+$, $\Longrightarrow^{\ell}$ and $\Longrightarrow^{\leqslant \ell}$, respectively. The language generated by a string $\alpha \in (\Sigma \cup N)^*$ is the set of all strings over Σ obtained from it in finitely many rewriting steps.

$$L_G(\alpha) = \{\, w \mid w \in \Sigma^*,\ \alpha \Longrightarrow^* w \}$$

The language generated by the grammar is the language generated by its initial symbol S.

$$L(G) = L_G(S) = \{\, w \mid w \in \Sigma^*,\ S \Longrightarrow *w \}$$

If multiple grammars are being considered, then the relation of one-step rewriting shall be denoted by $\overset{G}{\Longrightarrow}$.

Example 2.1(R). *In the grammar for the Dyck language given in Example 2.1, the string abaabb can be obtained by rewriting S as follows (the category symbol rewritten at each step is underlined).*

$$\underline{S} \Longrightarrow \underline{S}S \Longrightarrow a\underline{S}bS \Longrightarrow ab\underline{S} \Longrightarrow aba\underline{S}b \Longrightarrow abaa\underline{S}bb \Longrightarrow abaabb$$

Hence, $abaabb \in L(G)$.

The same string can be obtained by applying the same rules in a different order.

$$\underline{S} \Longrightarrow S\underline{S} \Longrightarrow Sa\underline{S}b \Longrightarrow \underline{S}aaSbb \Longrightarrow aSbaa\underline{S}bb \Longrightarrow a\underline{S}baabb \Longrightarrow abaabb$$

Both rewriting sequences represent the same parse of this string. The order of applying the rules is irrelevant.

The definition by rewriting incurred some corresponding terminology. Category symbols are called "nonterminal symbols", because these symbols require further rewriting for the rewriting sequence to terminate; accordingly, the symbols of the alphabet Σ are called "terminal symbols". Rules of a grammar are called "productions". This terminology, which reflects the technical aspects of the definition, but not the nature of the grammars, is generally avoided in this book, but knowing it is essential for reading the original papers.

2.1.2 Definition by deduction

According to the second definition, the language generated by a grammar is defined by a formal deduction system. This definition is important for making the logical nature of the grammars explicit. It also very well corresponds to the deductions performed by parsing algorithms.

Definition 2.1(D) (implicit in Kowalski [14, Ch. 3])**.** *For a grammar $G = (\Sigma, N, R, S)$, consider elementary propositions (items) of the form "a string w has a property X", with $w \in \Sigma^*$ and $X \in \Sigma \cup N$, denoted by $X(w)$. The deduction system uses the following axioms, which say that a one-symbol string a has the property a; that is, "a is a".*

$$\vdash a(a) \qquad\qquad \text{for all } a \in \Sigma$$

Each rule $A \to X_1 \dots X_\ell$, with $\ell \geqslant 0$ and $X_i \in \Sigma \cup N$, is regarded as the following schema for deduction rules.

$$X_1(u_1), \dots, X_\ell(u_\ell) \vdash A(u_1 \dots u_\ell) \qquad\qquad \text{for all } u_1, \dots, u_\ell \in \Sigma^*$$

A derivation (or a proof) of a proposition $A(u)$ is a sequence of such axioms and deductions, where the set of premises at every step consists of earlier derived propositions.

$$\begin{aligned} I_1 &\vdash X_1(u_1) \\ &\vdots \\ I_{z-1} &\vdash X_{z-1}(u_{z-1}) \\ I_z &\vdash A(u) \\ &\quad (\text{with } I_j \subseteq \{\, X_i(u_i) \mid i \in \{1, \dots, j-1\}\}, \text{ for all } j) \end{aligned}$$

The existence of such a derivation is denoted by $\vdash_G A(u)$.

Whenever an item $X(w)$ can be deduced from the above axioms by the given deduction rules, this is denoted by $\vdash X(w)$. Define $L_G(X) = \{\, w \mid \vdash X(w)\}$ and $L(G) = L_G(S) = \{\, w \mid \vdash S(w)\}$.

Example 2.1(D). *According to the grammar in Example 2.1, the membership of the string abaabb in the Dyck language is logically deduced as follows.*

$$\begin{aligned}
&\vdash S(\varepsilon) && \textit{(rule } S \to \varepsilon\textit{)}\\
&\vdash a(a) && \textit{(axiom)}\\
&\vdash b(b) && \textit{(axiom)}\\
a(a), S(\varepsilon), b(b) &\vdash S(ab) && \textit{(rule } S \to aSb\textit{)}\\
a(a), S(ab), b(b) &\vdash S(aabb) && \textit{(rule } S \to aSb\textit{)}\\
S(ab), S(aabb) &\vdash S(abaabb) && \textit{(rule } S \to SS\textit{)}
\end{aligned}$$

2.1.3 Definition by parse trees

A parse tree conveys the parse of a string according to a grammar. It can be obtained from a tree corresponding to a deduction.

Definition 2.1(T). *Let $G = (\Sigma, N, R, S)$ be a grammar and consider trees of the following form. Each node of the tree is labelled with a symbol from $\Sigma \cup N$, and its sons are linearly ordered. A node labelled with $a \in \Sigma$ must have no sons. For each node labelled with $A \in N$, let $X_1, \ldots, X_\ell$ be the labels in its sons; then the grammar must contain a rule $A \to X_1 \ldots X_\ell$. The* yield *of a tree is a string $w \in \Sigma^*$ formed by all nodes labelled with symbols in Σ, written according to the linear order. If X is the label of the root, such a tree is called a* parse tree of w from X.

Define $L_G(X) = \{ w \mid$ there is a parse tree of w from $X \}$ and $L(G) = L_G(S)$.

In each node labelled with $A \in N$, the rule used in this node can be determined from the node's sons. Nevertheless, it is often convenient to write the rule explicitly, as in the sample parse tree given in Figure 2.1, in which every node labelled by any rule should have label S by Definition 2.1(T).

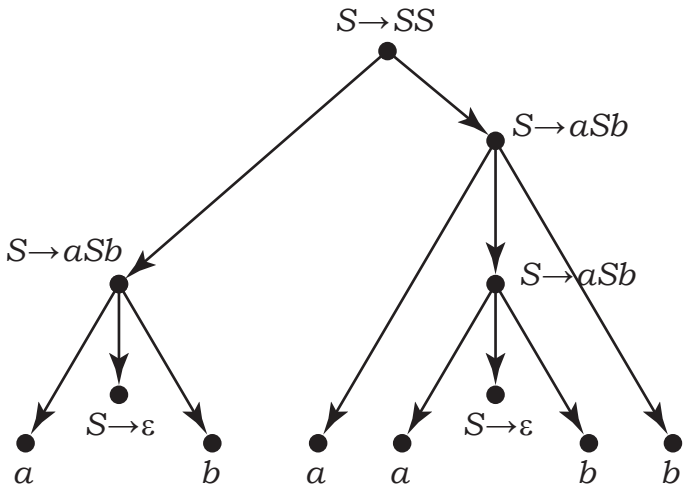

Figure 2.1: A parse tree of the string *abaabb* according to the grammar in Example 2.1.

2.1.4 Towards a definition by equations

Another equivalent definition of the language generated by a grammar is by a solution of a system of equations with languages as unknowns.

Let the category symbols in a grammar $G = (\Sigma, N, R, S)$ be numbered from 1 and n, with $N = \{A_1, A_2, \ldots, A_n\}$. Each category symbol A_i is interpreted as a *variable* that assumes the

value of a formal language over Σ. The correct value of this variable should be the set of strings with the property A_i. These values are defined by a system of *language equations* of the following form, where the right-hand sides $\varphi_i \colon (2^{\Sigma^*})^n \to 2^{\Sigma^*}$ are any functions on languages.

$$\begin{cases} A_1 &= \varphi_1(A_1, \ldots, A_n) \\ &\vdots \\ A_n &= \varphi_n(A_1, \ldots, A_n) \end{cases}$$

The right-hand side φ_i of each equation is constructed according to the rules of the grammar for the symbol A_i: each rule for A_i is transcribed as a concatenation of variables and singleton constant languages $\{a\}$, and the whole function φ_i is the union of all these concatenation.

Let Σ be an alphabet, let $n \geqslant 1$. Consider vectors on n languages of the form $(L_1, \ldots, L_n)$; the set of all such vectors is $(2^{\Sigma^*})^n$. Define a partial order of componentwise inclusion ($\sqsubseteq$) on the set of these vectors as $(K_1, \ldots, K_n) \sqsubseteq (L_1, \ldots, L_n)$ if and only if $K_i \subseteq L_i$. The least element is $\bot = (\varnothing, \ldots, \varnothing)$, the greatest one is $\top = (\Sigma^*, \ldots, \Sigma^*)$. For any two such vectors, their componentwise union is denoted by $(K_1, \ldots, K_n) \sqcup (L_1, \ldots, L_n) = (K_1 \cup L_1, \ldots, K_n \cup L_n)$.

Let $\varphi = (\varphi_1, \ldots, \varphi_n)$ be a vector function representing the right-hand side of the system. Assume that φ has the following two properties:

- φ is *monotone*, in the sense that for any two vectors K and L, the inequality $K \sqsubseteq L$ implies $\varphi(K) \sqsubseteq \varphi(L)$.
- φ is $\sqcup$-*continuous*, in the sense that for every increasing sequence of vectors of languages $\{L^{(i)}\}_{i=1}^{\infty}$ it holds that
$$\bigsqcup_{i=1}^{\infty} \varphi(L^{(i)}) = \varphi\Big(\bigsqcup_{i=1}^{\infty} L^{(i)}\Big).$$

Lemma 2.1. *If φ is monotone and $\sqcup$-continuous, then the least solution of a system $X = \varphi(X)$ is the vector*

$$L = \bigsqcup_{k=0}^{\infty} \varphi^k(\bot)$$

Proof. The sequence $\{\varphi^k(\bot)\}_{k \geqslant 0}$ is monotone, because $\bot \sqsubseteq \varphi(\bot)$ by the definition of the least element, and then $\varphi^{k-1}(\bot) \sqsubseteq \varphi^k(\bot)$ implies $\varphi^k(\bot) \sqsubseteq \varphi^{k+1}(\bot)$ by the monotonicity of φ. To see that L is the solution of the system, consider that

$$\varphi(L) = \varphi\big(\bigsqcup_{k=0}^{\infty} \varphi^k(\bot)\big) = \bigsqcup_{k=0}^{\infty} \varphi(\varphi^k(\bot)) = \bigsqcup_{k=0}^{\infty} \varphi^{k+1}(\bot)$$

because φ is $\sqcup$-continuous, and that $\bigsqcup_{k=0}^{\infty} \varphi^{k+1}(\bot) = \bigsqcup_{k=0}^{\infty} \varphi^k(\bot) = L$, because these two sequences are in fact the same. This shows that $\varphi(L) = L$.

Let K be any vector with $\varphi(K) = K$ and consider the sequences $\{\varphi^k(\bot)\}_{k=0}^{\infty}$ and $\{\varphi^k(K)\}_{k=0}^{\infty}$ Then each element of the former sequence is a subset of the corresponding element of the latter sequence: $\varphi^k(\bot) \sqsubseteq \varphi^k(K)$, which can be proved inductively on k, using the monotonicity of φ. This inequality is extended to the least upper bounds of the sequences as $L = \bigsqcup_{k=0}^{\infty} \varphi^k(\bot) \sqsubseteq \bigsqcup_{k=0}^{\infty} \varphi^k(K) = K$, which proves that L is the least among all solutions. □

2.1.5 Definition by language equations

Definition 2.1(E) (Ginsburg and Rice [9]). *Let $G = (\Sigma, N, R, S)$ be an ordinary grammar. The associated system of language equations is a system of equations in variables N, with each*

variable representing an unknown language over Σ, which contains one equation of the form $A = \varphi$ for each variable $A \in N$. This equation is of the following form.

$$A = \bigcup_{A \to X_1 \ldots X_\ell \in R} X_1 \cdot \ldots \cdot X_\ell \quad (\textit{for all } A \in N) \tag{2.2}$$

Each $X_i \in \Sigma$ in the equation represents a constant language $\{a\}$, and a rule $A \to \varepsilon$ is represented by a constant $\{\varepsilon\}$. Let $(\ldots, L_A, \ldots)_{A \in N}$ be the least solution of this system. Then $L_G(A)$ is defined as L_A for each $A \in N$.

Example 2.1(E). *The language equation corresponding to the grammar in Example 2.1 is*

$$S = (\{a\} \cdot S \cdot \{b\}) \cup (S \cdot S) \cup \{\varepsilon\}$$

and the Dyck language is its least solution (whereas the greatest solution is $S = \Sigma^*$).

As demonstrated by this example, the system of equations (2.2) corresponding to a grammar need not have a unique solution. However, it always has some solutions, and among them there is the *least solution* with respect to componentwise inclusion. This least solution can be obtained as a limit of an ascending sequence of vectors of languages, with the first element $\bot = (\varnothing, \ldots, \varnothing)$, and with every next element obtained by applying the right-hand sides of (2.2) as a vector function $\varphi\colon \left(2^{\Sigma^*}\right)^n \to \left(2^{\Sigma^*}\right)^n$ to the previous element. Since this function is monotone with respect to the partial ordering $\sqsubseteq$ of componentwise inclusion, the resulting sequence $\{\varphi^k(\bot)\}_{k \to \infty}$ is ascending, and the continuity of φ implies that its limit (least upper bound) $\bigsqcup_{k \geqslant 0} \varphi^k(\bot)$ is the least solution.

Example 2.2. *For the system of equations in Example 2.1(E), the sequence $\{\varphi^k(\bot)\}_{k \geqslant 0}$ takes the form*

$$\begin{aligned}
&\varphi^0(\bot) = \varnothing,\\
&\varphi^1(\bot) = \{\varepsilon\},\\
&\varphi^2(\bot) = \{\varepsilon, ab\},\\
&\varphi^3(\bot) = \{\varepsilon, ab, aabb, abab\},\\
&\varphi^4(\bot) = \{\varepsilon, ab, aabb, abab, aaabbb, aababb, abaabb, ababab, aabbab, aabbaabb, aabbabab, ababaabb, abababab\},\\
&\quad\vdots
\end{aligned}$$

In particular, $abaabb \in \varphi^4(\bot)$, because $ab, aabb \in \varphi^3(\bot)$ and $abaabb \in \varphi^3(\bot) \cdot \varphi^3(\bot) \subseteq \varphi^4(\bot)$. (by the concatenation $S \cdot S$ in the right-hand side of the equation).

Introduces its own terminology: category symbols are called *variables*.

2.1.6 Equivalence of the four definitions

To see that a grammar generates the same language under each of Definitions 2.1(R), 2.1(D), 2.1(T), 2.1(E).

The proof is quite boring, and a reader who reads it until the end may exclaim that *all these definitions are the same and there is nothing to prove.* If this happens, then the main goal of this section—that of building an understanding of the definition of ordinary grammars—will be accomplished.

Notation: for a $|N|$-tuple $L = (\ldots, L_B, \ldots)_{B \in N}$, let $[L]_A := L_A$ for each $A \in N$, and $[L]_a := \{a\}$ for each $a \in \Sigma$.

Theorem 2.1. *Let $G = (\Sigma, N, R, S)$ be a grammar, as in Definition 2.1. For every $X \in \Sigma \cup N$ and $w \in \Sigma^*$, the following four statements are equivalent:*

(R). $X \Longrightarrow^* w$,

(D). $\vdash X(w)$,

(T). there is a parse tree of w from X,

(E). $w \in \left[\bigsqcup_{k \geqslant 0} \varphi^k(\bot)\right]_X$.

Proof. (R) ⇒ (D) Induction on the number of steps in the rewriting of X to w.

Basis: $X \Longrightarrow^* w$ in zero steps. Then $X = w = a \in \Sigma^*$ and $\vdash a(a)$ by an axiom.

Induction step. Let $X \Longrightarrow^k w$ with $k \geqslant 1$. Then $X = A \in N$ and the rewriting begins by applying a rule $A \to X_1 \dots X_\ell$. Then, each X_i is rewritten to a string $w_i \in \Sigma^*$ in less than k steps, and $w = w_1 \dots w_\ell$. By the induction hypothesis, $\vdash X_i(w_i)$ for each i. Then the desired item $A(w)$ is deduced as

$$X_1(w_1), \dots, X_\ell(w_\ell) \vdash A(w) \qquad \text{by the rule } A \to X_1 \dots X_\ell.$$

(D) ⇒ (T) Induction on the number of applications of grammar rules in the deduction $\vdash X(w)$.

Basis: no rules applied. Then $X(w)$ must be an axiom, that is, $X = w = a$, and the requested tree consists of a single node labelled with a.

Induction step. Let $X(w)$ be deduced using one or more rules. Then $X = A \in N$. Consider the last step of its deduction, which must be of the form

$$X_1(w_1), \dots, X_\ell(w_\ell) \vdash A(w), \qquad \text{by some rule } A \to X_1 \dots X_\ell,$$

where $w_1 \dots w_\ell = w$. Each of its premises, $X_i(w_i)$, can then be deduced using fewer rules, and hence, by the induction hypothesis, there exists a parse tree with the root X_i and with the yield w_i. Construct a new tree by adding a new root labelled with A and by connecting it to $X_1, \dots, X_\ell$. This is a valid parse tree with the yield w.

(T) ⇒ (E) Consider a parse tree with a root $X \in \Sigma \cup N$ and yield $w \in \Sigma^*$, and let m be the number of nodes labelled with symbols in N. The proof is by induction on m.

Basis: no such nodes. Then the tree consists of a unique node labelled $X = a$, and its yield is $w = a$. Thus the claim holds as $a \in [\bot]_a$.

Induction step. If a tree contains at least one node labelled with a category symbol, then its root is among such nodes, that is, $X = A \in N$. Let $X_1, \dots, X_\ell$ be the labels of the sons of this node. For each X_i, consider the subtree with X_i as a root, and let w_i be the yield of that subtree. Then $w = w_1 \dots w_\ell$.

By the induction hypothesis for each i-th subtree, $w_i \in \left[\bigsqcup_{k \geqslant 0} \varphi^k(\bot)\right]_{X_i}$. Concatenating these statements gives $w_1 \dots w_\ell \in \left[\bigsqcup_{k \geqslant 0} \varphi^k(\bot)\right]_{X_1} \dots \left[\bigsqcup_{k \geqslant 0} \varphi^k(\bot)\right]_{X_\ell}$, and since $\bigsqcup_{k \geqslant 0} \varphi^k(\bot)$ is a solution of the system of language equations corresponding to the grammar, this implies that $w \in \left[\bigsqcup_{k \geqslant 0} \varphi^k(\bot)\right]_A$, as claimed.

(E) ⇒ (R) If $w \in \left[\bigsqcup_{k \geqslant 0} \varphi^k(\bot)\right]_X$, then there exists such a number $k \geqslant 0$, that $w \in [\varphi^k(\bot)]_X$. The proof is by induction on the least such number k.

Basis: $w \in [\varphi^0(\bot)]_X = [\bot]_X$. Then $w = X = a$ and $a \Longrightarrow a$ in zero steps.

Induction step. Let $w \in [\varphi^k(\bot)]_X$ with $k \geqslant 1$ and $w \notin [\varphi^{k-1}(\bot)]_X$. Then $X = A \in N$ and accordingly

$$w \in \varphi_A(\varphi^{k-1}(\bot)) = \bigcup_{A \to X_1 \dots X_\ell \in R} [\varphi^{k-1}(\bot)]_{X_1} \dots [\varphi^{k-1}(\bot)]_{X_\ell}.$$

This means that there exists such a rule $A \to X_1 \dots X_\ell$ and such a partition $w = w_1 \dots w_\ell$, that $w_i \in [\varphi^{k-1}(\bot)]_{X_i}$ for each i. Then, by the induction hypothesis, X_i can be rewritten to w_i. Using these rewritings, A can be rewritten to w as follows:

$$A \Longrightarrow X_1 \dots X_\ell \Longrightarrow^* w_1 \dots w_\ell = w.$$

□

2.2 Examples

What can ordinary grammars do? They can form parse tree matching remote substrings to each other, thus counting them.

Example 2.3. *The language* $\{ a^n b^n \mid n \geqslant 0 \}$ *is described by the following grammar.*

$$S \to aSb \mid \varepsilon$$

Example 2.4. *The language* $\{ a^n b^{2n} \mid n \geqslant 0 \}$ *is described by the following grammar.*

$$S \to aSbb \mid \varepsilon$$

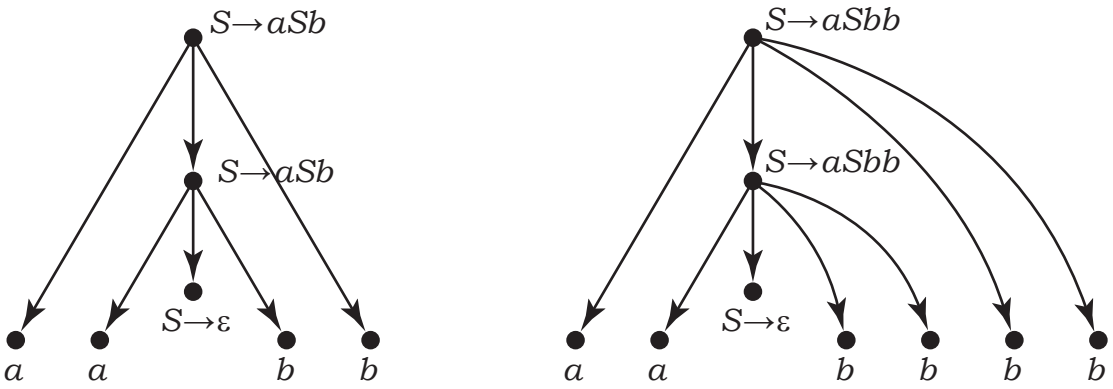

Figure 2.2: (left) A parse tree of the string *aabb* according to the grammar in Example 2.3; (right) A parse tree of *aabbbb* according to the grammar in Example 2.4.

Example 2.5. *The language* $\{ a^m b^n | m \geqslant 0, m \leqslant n \leqslant 2m \}$ *is described by the following grammar.*

$$S \to aSb \mid aSbb \mid \varepsilon$$

Example 2.6. *The language* $L = \{ w \mid w \in \{a,b\}^*,\ |w|_a = |w|_b \}$ *is described by the following grammar.*

$$S \to SS \mid aSb \mid bSa \mid \varepsilon$$

Proof. The claim that this grammar generates the given language requires an argument. All strings generated by the grammar are in L. To see that every string belonging to the language is generated by the grammar, let $w \in L$ and consider the function $f \colon \{0, 1, \dots, |w|\} \to \mathbb{Z}$ described by $f(|u|) = |u|_a - |u|_b$ for every partition $w = uv$. Then either it has an intermediate zero, thus w is obtained by the rule $S \to SS$, or it doesn't, in which case either all its values are positive, or all are negative. In the former case, it must begin with a and end with b, and therefore is generated by the rule $S \to aSb$. □

Example 2.7. *The language of palindromes* $\{ w \mid w \in \{a,b\}^*,\ w = w^R \}$ *is described by the following grammar.*

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

Example 2.8. *The language* $\{ a^m b^{m+n} a^n \mid m, n \geqslant 0 \}$ *is described by the following grammar, which treats a string* $a^m b^{m+n} a^n$ *as a concatenation* $(a^m b^m)(b^n a^n)$*, and defines the two parts separately, the first by* A *and the second by* B*.*

$$\begin{aligned} S &\to AB \\ A &\to aAb \mid \varepsilon \\ B &\to bBa \mid \varepsilon \end{aligned}$$

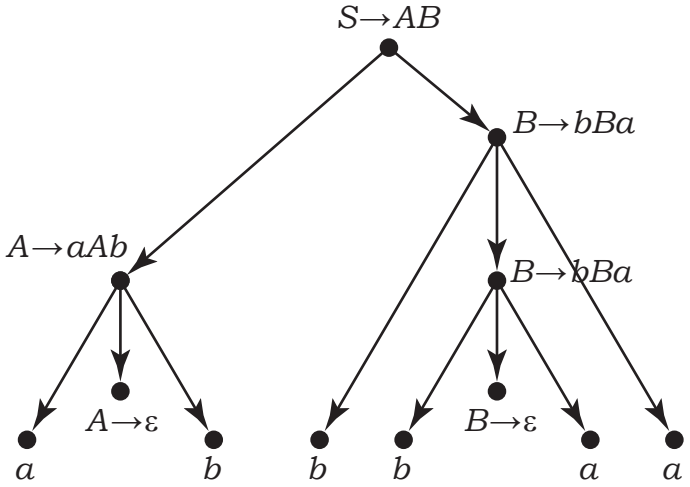

Figure 2.3: A parse tree of the string *abbbaa* according to the grammar in Example 2.8.

Example 2.9. *Let* $\Sigma = \{a,b\}$*. The language* $\overline{\{ ww \mid w \in \{a,b\}^* \}}$ *is described by the following grammar.*

$$\begin{aligned} S &\to AB \mid BA \mid O \\ A &\to XAX \mid a \\ B &\to XBX \mid b \\ X &\to a \mid b \\ O &\to XXO \mid X \end{aligned}$$

(see Figure 2.4)

Example 2.10. *The language* $\{ a^{k_1} b \dots a^{k_\ell} b \mid \ell \geqslant 1,\ k_1, \dots, k_\ell \geqslant 0,\ \exists i : k_i = \ell \}$ *is described by the following grammar.*

$$\begin{aligned} S &\to BC \\ A &\to aA \mid b \\ B &\to ABa \mid \varepsilon \\ C &\to aCA \mid ab \end{aligned}$$

(see Figure 2.5)

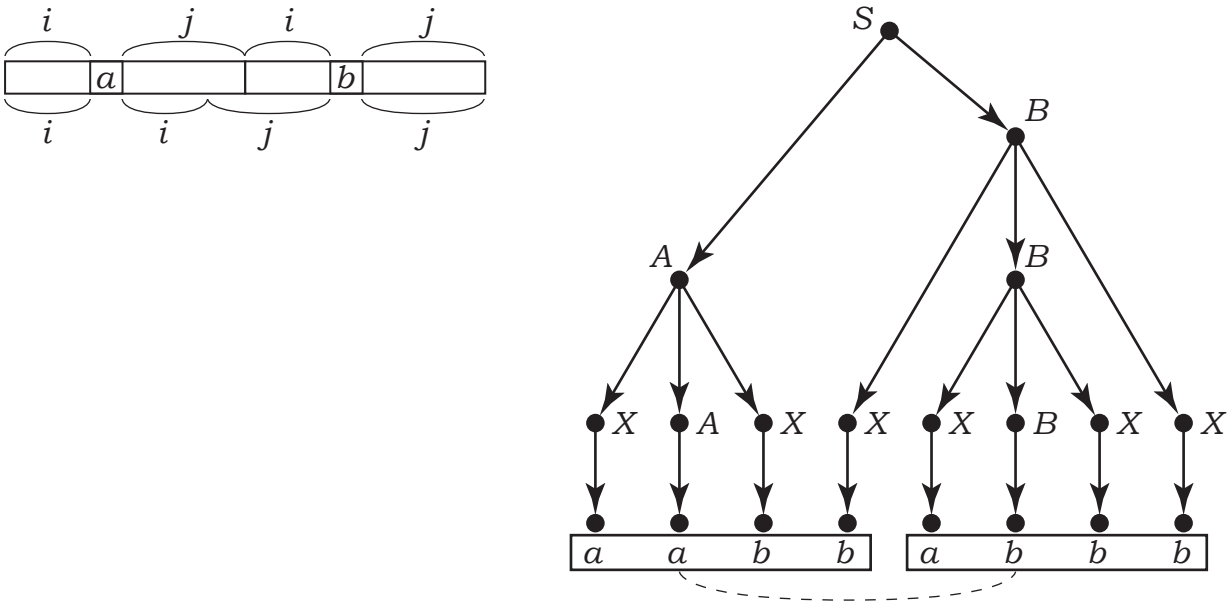

Figure 2.4: (left) How the grammar in Example 2.9 defines strings of the form uv with $|u| = |v|$ and $u \neq v$; (right) a parse tree of the string *aabbabbb*.

Lemma 2.2. *Every regular language is described by an ordinary formal grammar* $G = (\Sigma, N, R, S)$*, in which every rule is of the form* $A \to aB$*, with* $a \in \Sigma$ *and* $B \in N$*, or of the form* $A \to \varepsilon$*.*

Proof. by simulating a DFA. □

Exercises

2.2.1. Construct a grammar for $\{ w \mid w \in \{a,b\}^*,\ |w|_a < |w|_b \}$.

2.2.2. Construct a grammar for $\overline{\{ ww \mid w \in \{a,b,c\}^* \}}$.

2.2.3. Construct a grammar for $\{ a^{k_1} b \dots a^{k_\ell} b \mid \ell \geqslant 1,\ k_i \geqslant 0,\ \exists i : k_i = i \}$.

2.3 Limitations

2.3.1 The pumping lemma

Lemma 2.3 (The pumping lemma: Bar-Hillel, Perles and Shamir [2])**.** *For every ordinary language* $L \subseteq \Sigma^*$ *there exists a constant* $p \geqslant 1$*, such that for every string* $w \in L$ *with* $|w| \geqslant p$ *there exists a factorization* $w = xuyvz$*, where* $|uv| > 0$ *and* $|uyv| \leqslant p$*, such that* $xu^i yv^i z \in L$ *for all* $i \geqslant 0$*.*

Sketch of a proof. Let $G = (\Sigma, N, R, S)$ be a grammar generating L. Let $m = \max_{A \to \alpha \in R} |\alpha|$ and define $p = m^{|N|} + 1$.

Consider any string $w \in L$ of length at least p and consider its parse tree. Let an internal node s in the tree be called *non-trivial* if the partitition of its subtree induced by its sons non-trivially divides the leaves (that is, it is not the case that one of the sons has all the leaves and the others have none).

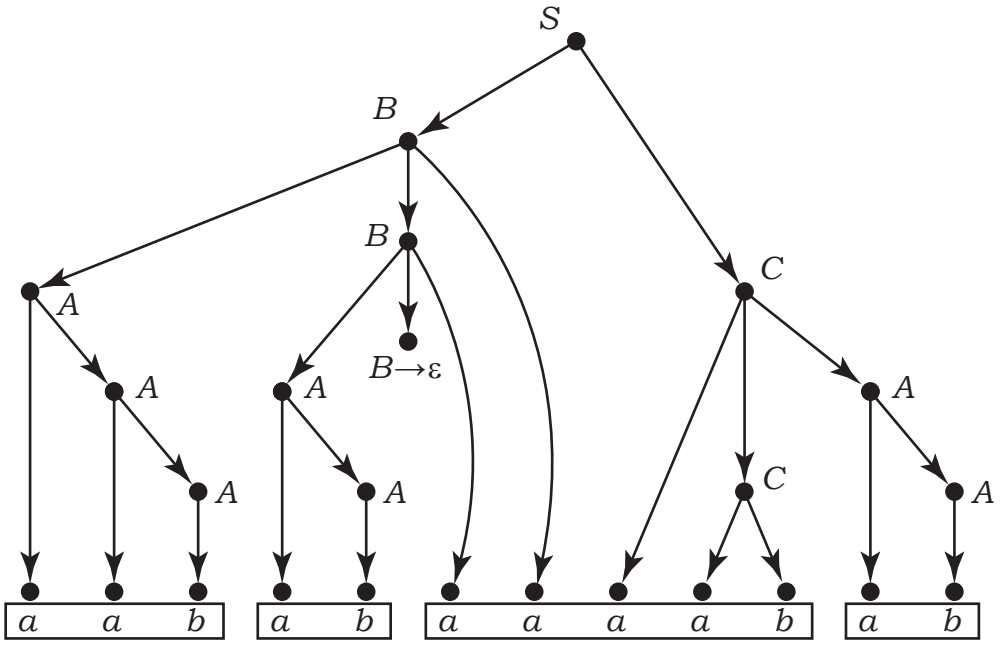

Figure 2.5: (left) A parse tree of the string $w = aababaaaabab$ according to the grammar in Example 2.10.

Then the parse tree should contain a path with at least $|N|+1$ non-trivial nodes, some symbol $A \in N$ must repeat twice in this path and the section of the tree between these two instances of A can be repeated 0 or more times, thus obtaining parse trees of xu^iyv^iz. This section of the tree represents a derivation of uAv from A. □

Example 2.11. *The language* $L = \{\, a^n b^n c^n \mid n \geqslant 0\}$ *is not ordinary.*

Proof. Suppose it is, and let $p \geqslant 1$ be the constant given by the pumping lemma. Consider $w = a^p b^p c^p$. Then there exists a factorization $w = xuyvz$. There are several cases:

- Either u or v is not in $a^* \cup b^* \cup c^*$, that is, the string spans over the boundary between as, bs and cs. Then $xu^2yv^2z \notin a^*b^*c^*$ and cannot be in L.
- If $u, v \in a^*$, then $xyz = a^{p-|uv|}b^pc^p \notin L$. The cases of $u, v \in b^*$ and $u, v \in c^*$ are similar.
- If $u \in a^*$ and $v \in b^*$, then $xu^0yv^0z = a^{p-|u|}b^{p-|v|}c^p$, which is not in L because $p - |u| \neq p$ or $p - |v| \neq p$. The case of $u \in b^*$ and $v \in c^*$ is similar.

In each case a contradiction is obtained. □

Example 2.12 (Floyd [7], here "ported" from Algol 60 to C)**.** *Consider that the following string is a valid C program if and only if* $i = j = k$*.*

$$\texttt{main() \{ int } \underbrace{\texttt{x...x}}_{i\geqslant 1}\texttt{; } \underbrace{\texttt{x...x}}_{j\geqslant 1} \texttt{ = } \underbrace{\texttt{x...x}}_{k\geqslant 1}\texttt{; \}}$$

Then there is no ordinary grammar for the set of well-formed programs in C.

Lemma 2.4 (Ogden's lemma [16])**.** *For every ordinary language* $L \subseteq \Sigma^*$ *there exists a constant* $p \geqslant 1$*, such that for every string* $w \in L$ *with* $|w| \geqslant p$ *and for every set* $P \subseteq \{1, \ldots, |w|\}$ *of distinguished positions in* w*, with* $|P| \geqslant p$*, there exists a factorization* $w = xuyvz$*, where*

- *uv contains at least one distinguished position,*
- *uyv contains at most p distinguished positions,*

such that $xu^iyv^iz \in L$ for all $i \geqslant 0$.

Proof. By the same argument, in which non-trivial nodes are defined by having a non-trivial partition of distinguished leaves in a subtree. □

Example 2.13. *The language $L = \{ a^m b^n c^n \mid m, n \geqslant 0,\ m \neq n \}$ is not described by any ordinary grammar.*

Proof. Suppose it is, and let p be the constant given by the pumping lemma. Consider the string $w = a^{p+p!}b^pc^p$ with distinguished positions b^p. □

Example 2.14 (Bader and Moura [1])**.** *The language $\{ ab^p \mid p \text{ is prime} \} \cup \overline{ab^*}$ satisfies Ogden's lemma, but is not described by any ordinary grammar.*

Proof. TBW. □

This example motivates an even stronger pumping lemma, which also features excluded positions.

Lemma 2.5 (Bader and Moura [1])**.** *For every ordinary language $L \subseteq \Sigma^*$ there exists a constant $p \geqslant 1$, such that for every string $w \in L$ and for every two sets $P, Q \subseteq \{1, \ldots, |w|\}$ of distinguished and excluded positions in w, which satisfy $|P| \geqslant p^{|Q|+1}$, there exists a factorization $w = xuyvz$, where*

- *uv contains at least one distinguished position and no excluded positions;*
- *if d is the number of distinguished positions in uyv and e is the number of excluded positions in uyv, then $d \leqslant n^{e+1}$,*

and $xu^iyv^iz \in L$ for all $i \geqslant 0$.

2.4 Closure properties

2.5 Normal forms

Chapter 13

Selected theoretical topics

13.1 Homomorphic characterizations

13.2 Inverse homomorphic characterizations

Bibliography

[1] C. Bader, A. Moura, "A generalization of Ogden's lemma", *Journal of the ACM*, 29:2 (1982), 404–407.

[2] Y. Bar-Hillel, M. Perles, E. Shamir, "On formal properties of simple phrase-structure grammars", *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14 (1961), 143–177.

[3] N. Chomsky, "Three models for the description of language", *IRE Transactions on Information Theory* 2:3 (1956), 113–124.

[4] N. Chomsky, "On certain formal properties of grammars", *Information and Control*, 2:2 (1959), 137–167.

[5] N. Chomsky, M. P. Schützenberger, "The algebraic theory of context-free languages", in: Braffort, Hirschberg (Eds.), *Computer Programming and Formal Systems*, North-Holland Publishing Company, Amsterdam, 1963, 118–161.

[6] J. Engelfriet, "An elementary proof of double Greibach normal form", *Information Processing Letters*, 44:6 (1992), 291–293.

[7] R. W. Floyd, "On the non-existence of a phrase structure grammar for ALGOL 60", *Communications of the ACM*, 5 (1962), 483–484.

[8] S. Ginsburg, *The Mathematical Theory of Context-Free Languages*, McGraw-Hill, 1966.

[9] S. Ginsburg, H. G. Rice, "Two families of languages related to ALGOL", *Journal of the ACM*, 9 (1962), 350–371.

[10] S. Ginsburg, G. Rose, , "Operations which preserve definability in languages", *Journal of the ACM*, 10:2 (1963), 175–195.

[11] S. A. Greibach, "A new normal-form theorem for context-free phrase structure grammars", *Journal of the ACM*, 12 (1965), 42–52.

[12] S. A. Greibach, "The hardest context-free language", *SIAM Journal on Computing*, 2:4 (1973), 304–310.

[13] J. E. Hopcroft, J. D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, 1979.

[14] R. Kowalski, *Logic for Problem Solving*, North-Holland, Amsterdam, 1979.

[15] A. N. Maslov, "Cyclic shift operation for languages", *Problems of Information Transmission*, 9 (1973), 333–338.

[16] W. F. Ogden, "A helpful result for proving inherent ambiguity", *Mathematical Systems Theory* 2:3 (1968), 191–194.

[17] T. Oshiba, "Closure property of the family of context-free languages under the cyclic shift operation", *Transactions of IECE*, 55D (1972), 119–122.

[18] D. J. Rozenkrantz, "Matrix equations and normal forms for context-free grammars", *Journal of the ACM*, 14:3 (1967), 501–507.

[19] S. Scheinberg, "Note on the boolean properties of context free languages", *Information and Control*, 3 (1960), 372-375.

[20] F. J. Urbanek, "On Greibach normal form construction", *Theoretical Computer Science*, 40 (1985), 315–317.