Contents

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 $X_1 \ldots X_\ell$ of zero or more symbols symbols $X_1, \ldots, X_\ell \in \Sigma \cup N$, with $\ell \geqslant 0$.*

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

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

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

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

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 [4], 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 [4]). *Let $G = (\Sigma, N, R, S)$ be a grammar. Define a relation $\Longrightarrow$ of one-step rewriting on $(\Sigma \cup N)^*$ as follows.*

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

If there is a sequence of zero or more rewriting steps,

$$\alpha_0 \Longrightarrow \alpha_1 \Longrightarrow \ldots \Longrightarrow \alpha_\ell$$

*then *** is abbreviated by the following notation: $\alpha_0 \Longrightarrow^* \alpha_\ell$. Also there are symbols for one or more rewriting steps ($\alpha_0 \Longrightarrow^+ \alpha_\ell$, with $\ell \geqslant 0$) for exactly ℓ steps ($\alpha_0 \Longrightarrow^\ell \alpha_\ell$) and for at most ℓ steps ($\alpha_0 \Longrightarrow^{\leqslant \ell} \alpha_\ell$)*

The language generated by a string $\alpha \in (\Sigma \cup N)^*$ is the set of all strings over Σ obtained from it in zero or more 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 symbol for a rewriting step is marked with the name of the grammar ($\overset{G}{\Longrightarrow}$).

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

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

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.

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

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.*

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

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).

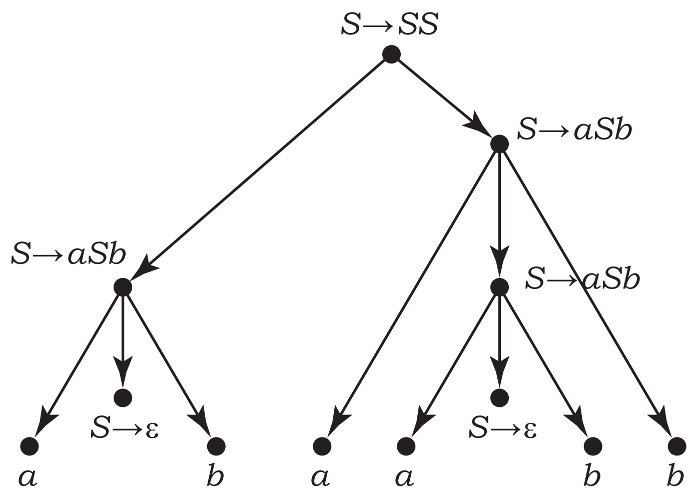


Figure 2.1: A parse tree of the string *abaabb* according to the grammar in Example 2.1.

2.1.4 Towards a definition by language 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 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. *A function on languages φ formed from variables and constant languages using concatenation and union is monotone and continuous.*

Proof. By induction on the structure of individual expressions. Variables and constants are monotone, union/concatenation of monotone functions is monotone. Same with continuity. □

Lemma 2.2 (attributed to Kleene)**.** *If φ is monotone and $\sqcup$-continuous, then the following vector of languages is least solution of the system $X = \varphi(X)$.*

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

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 equations

Definition 2.1(E) (Ginsburg and Rice [10]). *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*

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

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 $L = (\dots, L_B, \dots)_{B \in N}$, 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). $w \in \big[\bigsqcup_{k \geqslant 0} \varphi^k(\bot)\big]_X$.

Proof. $\boxed{\text{(R)} \Rightarrow \text{(D)}}$ 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

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

$\boxed{\text{(D)} \Rightarrow \text{(T)}}$ 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.

$\boxed{\text{(T)} \Rightarrow \text{(E)}}$ 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, $w_i \in \big[\bigsqcup_{k \geqslant 0} \varphi^k(\bot)\big]_{X_i}$. Concatenating these statements gives $w_1 \dots w_\ell \in \big[\bigsqcup_{k \geqslant 0} \varphi^k(\bot)\big]_{X_1} \cdots \big[\bigsqcup_{k \geqslant 0} \varphi^k(\bot)\big]_{X_\ell}$, 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 $w \in \big[\bigsqcup_{k \geqslant 0} \varphi^k(\bot)\big]_A$, as claimed.

$\boxed{\text{(E)} \Rightarrow \text{(R)}}$ If $w \in \big[\bigsqcup_{k \geqslant 0} \varphi^k(\bot)\big]_X$, 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

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

This means that there exists such a rule $A \to X_1 \ldots X_\ell$ and such a partition $w = w_1 \ldots w_\ell$, 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:

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

□

2.2 Examples

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.*

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

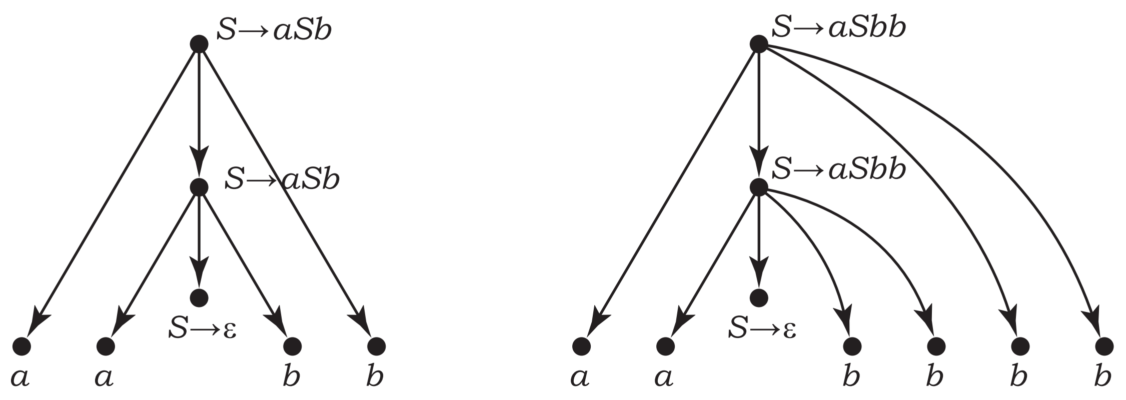


Figure 2.2: (left) A parse tree of the string *aabb* according to the grammar in Example 2.3; (right) A parse tree of *aabbbb* according to the grammar in Example 2.4.

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

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

Lemma 2.3. *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. □

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 $f \colon \{0, 1, \ldots, |w|\} \to \mathbb{Z}$ 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 $S \to aSb$. $\square$

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.

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

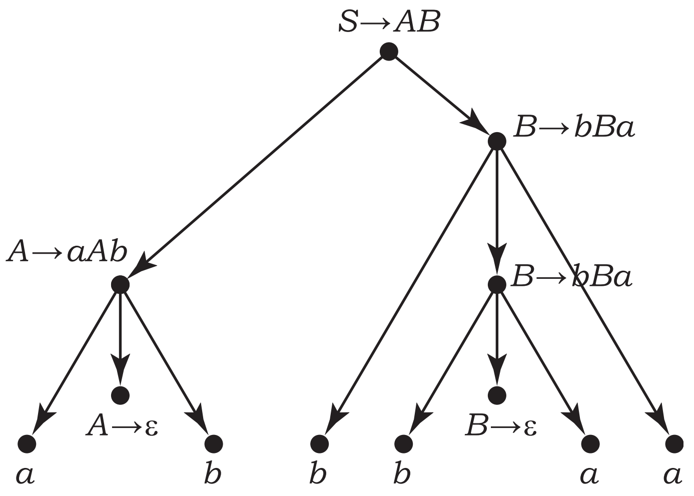


Figure 2.3: A parse tree of the string *abbbaa* according to the grammar in Example 2.8.

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)

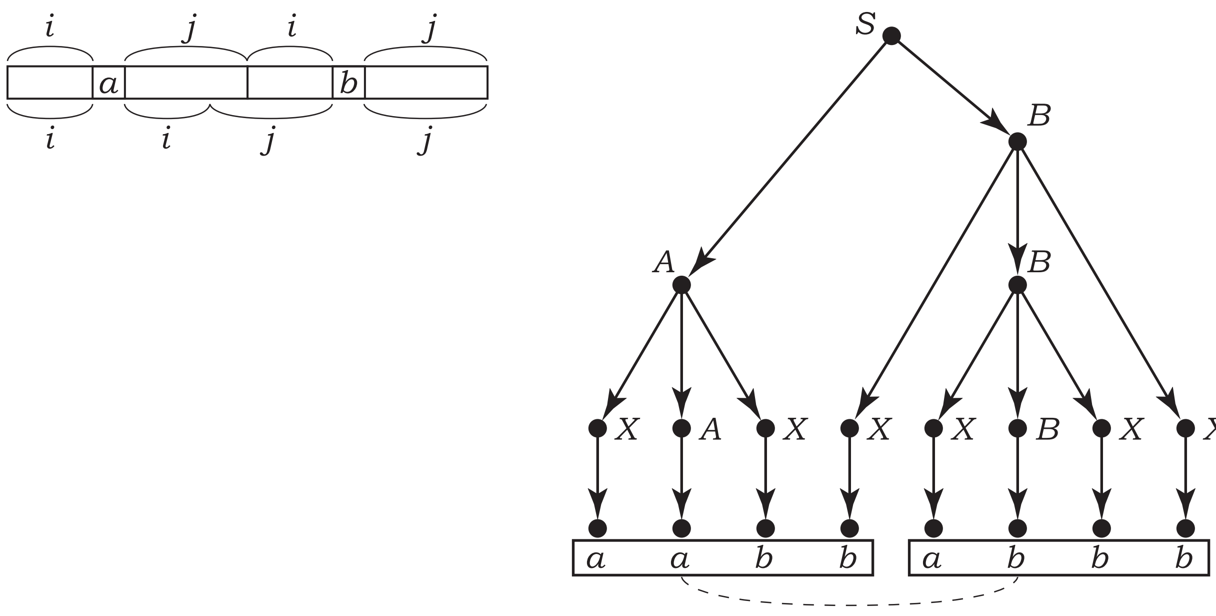


Figure 2.4: (left) How the grammar in Example 2.9 defines strings of the form uv with $|u| = |v|$ and $u \neq v$; (right) a parse tree of the string $aabbabbb$.

Example 2.10. *The language* $\{ a^{k_1}b \ldots a^{k_\ell}b \mid \ell \geqslant 1,\ k_1, \ldots, k_\ell \geqslant 0,\ \exists i : k_i = \ell \}$ *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)

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 $\{ a^{k_1}b \ldots a^{k_\ell}b \mid \ell \geqslant 1,\ k_i \geqslant 0,\ \exists i : k_i = i \}$.

2.3 Limitations

2.3.1 The pumping lemma

Lemma 2.4 (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 $xu^iyv^iz \in L$ 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 $p = m^{|N|} + 1$.

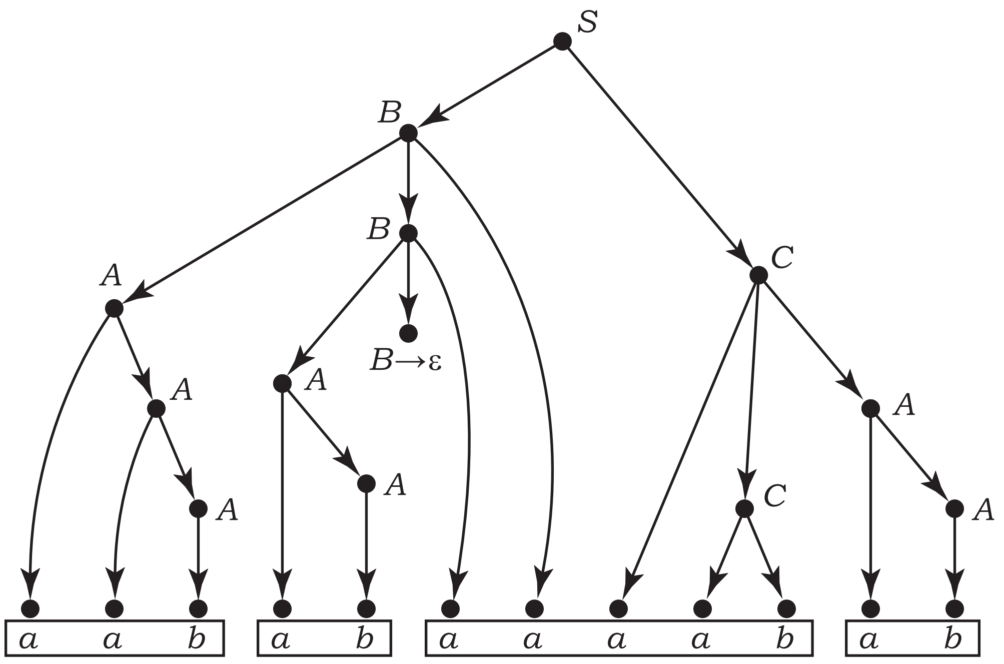


Figure 2.5: A parse tree of the string $w = aababaaaabab$ according to the grammar in Example 2.10.

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).

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. □

Used to prove that a language is not described by any ordinary grammar, as follows. Assuming that a language L is described by one, the pumping lemma provides a certain constant p, and then claims something about every sufficiently long string in L. In order to obtain a contradiction, it is sufficient to construct any single string w (depending on p), and then prove that for every factorization $w = xuyvz$ satisfying the conditions in the lemma, there is a string of the form xu^iyv^iz that does not belong to L.

Example 2.11. *The language $L = \{ a^nb^nc^n \mid n \geqslant 0\}$ is not described by any ordinary grammar.*

Proof. Suppose it is, and let $p \geqslant 1$ be the constant given by the pumping lemma. Consider $w = a^pb^pc^p$. 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 [8], 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.

2.3.2 Ogden's lemma and its variants

Lemma 2.5 (Ogden's lemma, Ogden [17]). *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. □

The standard pumping lemma is Ogden's lemma with every position distinguished.

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

Proof. Suppose it is, and let p be the constant given by Ogden's 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 2.6 (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

Proposition 2.1. *The ordinary languages are closed under union, concatenation and star.*

Proof. If $G_i = (\Sigma, N_i, R_i, S_i)$ with $i \in \{1, 2\}$ and $N_1 \cap N_2 = \varnothing$ are ordinary grammars, then the grammars $(\Sigma, N_1 \cup N_2 \cup \{S\}, R_1 \cup R_2 \cup \{S \to S_1,\ S \to S_2\}, S)$ and $(\Sigma, N_1 \cup N_2 \cup \{S\}, R_1 \cup R_2 \cup \{S \to S_1S_2\}, S)$ generate $L(G_1) \cup L(G_2)$ and $L(G_1)L(G_2)$, respectively, while the grammar $(\Sigma, N_1 \cup \{S\}, R_1 \cup \{S \to S_1S,\ S \to \varepsilon\}, S)$ generates $L(G_1)^*$.

The correctness of each construction can be immediately proved using language equations. □

Theorem 2.2 (Scheinberg [20]). *The ordinary languages are not closed under intersection and complementation.*

Proof. Consider the following two grammars, which generate the languages $L_1 = \{\, a^ib^\ell c^\ell | i, \ell \geqslant 0\}$ and $L_2 = \{\, a^mb^mc^j \mid j, m \geqslant 0\}$, respectively.

$$\begin{array}{rcl} S_1 & \to & aS_1 \mid A \\ A & \to & bAc \mid \varepsilon \end{array} \qquad\qquad \begin{array}{rcl} S_2 & \to & S_2c \mid B \\ B & \to & aBb \mid \varepsilon \end{array}$$

Suppose the ordinary languages are closed under intersection. Then the intersection

$$L_1 \cap L_2 = \{\, a^ib^\ell c^\ell \mid i, \ell \geqslant 0\} \cap \{\, a^mb^mc^j \mid j, m \geqslant 0\} = \{\, a^nb^nc^n \mid n \geqslant 0\}$$

should be ordinary as well, which contradicts Example 2.11.

Turning to the complementation, consider the following grammar generating the language $\{\, a^kb^\ell c^m \mid k \neq \ell \text{ or } \ell \neq m\}$.

$$\begin{array}{rcl} S & \to & AD \mid EC \\ A & \to & aA \mid \varepsilon \\ B & \to & bB \mid \varepsilon \\ C & \to & cC \mid \varepsilon \\ D & \to & aDb \mid aA \mid bB \\ E & \to & bEc \mid bB \mid cC \end{array}$$

The following language is ordinary as well, as a union of two ordinary languages.

$$L = \{\, a^kb^\ell c^m \mid k \neq \ell \text{ or } \ell \neq m\} \cup \overline{a^*b^*} = \overline{\{\, a^nb^nc^n \mid n \geqslant 0\}}$$

Then, if the ordinary languages are closed under complementation, then the language $\overline{L}$ must be ordinary as well, which is known to be untrue. □

Though an intersection of two ordinary languages is not necessarily ordinary, it is ordinary in case one of these languages is regular.

Theorem 2.3 (Bar-Hillel, Perles and Shamir [2]). *For every ordinary language L and for every regular language K, the language $L \cap K$ is ordinary.*

Proof. Let $G = (\Sigma, N, R, S)$ be a grammar and let $M = (\Sigma, Q, q_0, \delta, F)$ be a DFA. Construct a new grammar $G' = (\Sigma, N', R', S')$ with a set of nonterminals $N' = \{S'\} \cup \{\, A_{q,q'} \mid A \in N,\ q, q' \in Q\}$. The idea is that $L_{G'}(A_{q,q'}) = L_G(A) \cap \{\, w \mid \delta(q, w) = q'\}$.

For every rule $A \to X_1 \dots X_\ell \in R$, and for every sequence of intermediate states $p_0, p_1, \dots, p_\ell \in Q$, where $p_0 = q$, $p_\ell = q'$ and $X_i = a_i \in \Sigma$ implies $\delta(p_{i-1}, a_i) = p_i$, the new grammar contains a rule

$$A_{q,q'} \to X_1'X_2' \dots X_\ell', \quad \left(\text{where } X_i' = \begin{cases} A_{p_{i-1},p_i}, & \text{if } X_i = A \in N \\ a, & \text{if } X_i = a \in \Sigma \end{cases}\right)$$

The rules for the new initial symbol are

$$S' \to S_{q_0,q} \quad (q \in F)$$

□

One more very simple closure result. Let Σ and Γ be alphabets. A mapping $h\colon \Sigma^* \to \Gamma^*$ is called a *homomorphism* if $h(uv) = h(u) \cdot h(v)$ for all $u, v \in \Sigma^*$. This definition, in particular, implies that $h(\varepsilon) = \varepsilon$, and that a homomorphism is completely defined by the images of all symbols from Σ.

Theorem 2.4. *Let Σ and Γ be alphabets, let $h\colon \Sigma^* \to \Gamma^*$ be a homomorphism and let $G = (\Sigma, N, R, S)$ be any ordinary grammar. Define $h(A) = A$ for all $A \in N$. Then the grammar $G' = (\Gamma, N, R', S)$ with $R' = \{\, A \to h(\alpha) \mid A \to \alpha \in R\}$ generates the language $h(L(G))$.*

Another operation is *cyclic shift*, defined as $\text{SHIFT}(L) = \{\, vu \mid u, v \in \Sigma^*,\ uv \in L\}$. The closure of ordinary languages under this operation was independently obtained by Oshiba [18], by Maslov [16] and by Hopcroft and Ullman [14, solution to Ex. 6.4c].

Theorem 2.5 (Oshiba [18]; Maslov [16]; Hopcroft, Ullman [14, solution to Ex. 6.4c])**.** *Let $G = (\Sigma, N, R, S)$ be any ordinary grammar and consider the grammar $G' = (\Sigma, N \cup \widetilde{N} \cup \{S'\}, R \cup R', S')$, with $\widetilde{N} = \{\, \widetilde{A} \mid A \in N\}$, which contains all rules from the original grammar, and the following additional rules in R'.*

$$\begin{aligned} &S' \to \theta\widetilde{A}\eta \quad (A \to \eta\theta \in R) \\ &\widetilde{B} \to \beta\widetilde{A}\alpha \quad (A \to \alpha B\beta \in R) \\ &\widetilde{S} \to \varepsilon \end{aligned}$$

Then $L_{G'}(A) = L_G(A)$ and $L_{G'}(\widetilde{A}) = \{\, vu | u, v \in \Sigma^*, S \overset{G}{\Longrightarrow}{}^* uAv\}$ for all $A \in N$, and accordingly $L(G') = \text{SHIFT}(L(G))$.

Sketch of a proof. Consider a parse tree of a string uv according to the original grammar G. Define the main path leading to the boundary between u and v. The parse tree is turned inside out in relation to the main path, as shown in Figure 2.6. □

Theorem 2.6 (Ginsburg and Rose [11])**.** *The ordinary languages are closed under inverse homomorphisms.*

Example 2.15 (Ginsburg [9])**.** *Let $K = a\{\, b^\ell a^\ell \mid \ell \geqslant 1\}^*$ and $L = \{\, a^m b^{2m} \mid m \geqslant 1\}^*$. Then $K^{-1}L \cap b^* = \{\, b^{2^n} \mid n \geqslant 1\}$. Therefore, the ordinary languages are not closed under the quotient operation.*

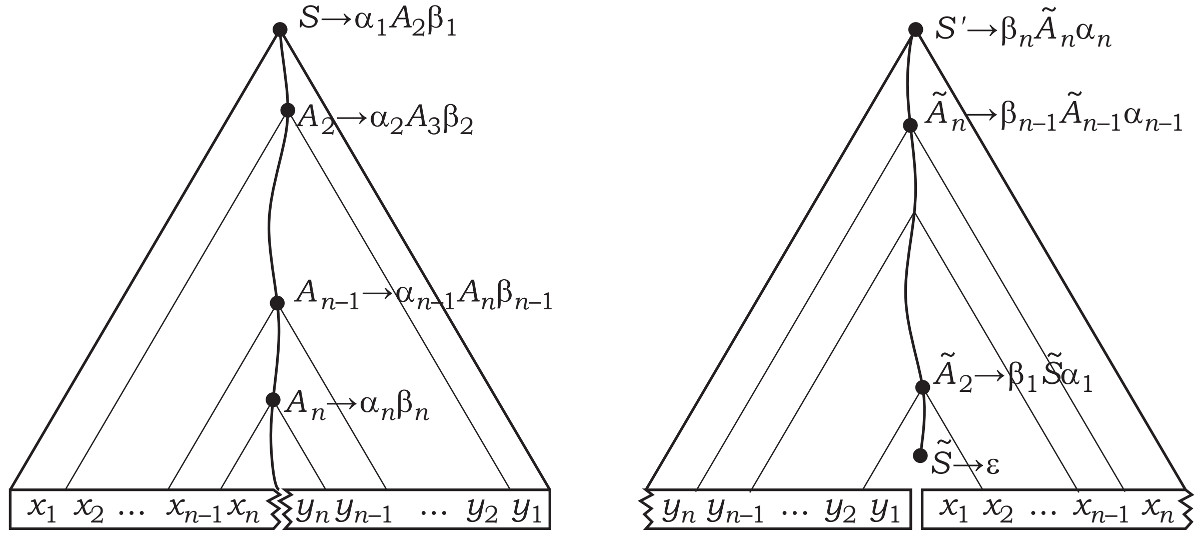
Example 2.16. *Let $L = \{\, a^m b^m c^n d^{3n+2} \mid m, n \geqslant 0\}$. Then $(\frac{1}{2}L) \cap a^*b^*c^*d = \{\, a^n b^n c^n d \mid n \geqslant 0\}$, and therefore the ordinary languages are not closed under the $\frac{1}{2}$ operation.*

Exercises

2.4.1. Consider the following quasi-square-root operation on formal languages.

$$\sqrt{L} = \{\, w \mid ww \in L\}$$

It is certainly *not an inverse of concatenation*, and should be regarded as a kind of joke. Prove that the ordinary languages are not closed under this operation.



$S \to \alpha_1 A_2 \beta_1$
$A_2 \to \alpha_2 A_3 \beta_2$
$A_{n-1} \to \alpha_{n-1} A_n \beta_{n-1}$
$A_n \to \alpha_n \beta_n$
$x_1\ x_2 \dots x_{n-1} x_n$
$y_n\ y_{n-1} \dots y_2\ y_1$
$S' \to \beta_n \tilde{A}_n \alpha_n$
$\tilde{A}_n \to \beta_{n-1} \tilde{A}_{n-1} \alpha_{n-1}$
$\tilde{A}_2 \to \beta_1 \tilde{S} \alpha_1$
$\tilde{S} \to \varepsilon$
$y_n\ y_{n-1} \dots y_2\ y_1$
$x_1\ x_2 \dots x_{n-1}\ x_n$

Figure 2.6: Construction for the cyclic shift: (left) a parse tree of a string uv in the original grammar; (right) turning this tree inside out to obtain a parse tree of vu in the new grammar.

2.5 Normal forms

No ε rules, no chain rules: Chomsky nf. No left recursion (and even left chains): Greibach nf. No left or right recursion (and chains): Rosenkrantz nf.

2.5.1 Eliminating null rules

Example 2.17. *Consider the following grammar that defines the language $\{ab, b\}$.*

$$\begin{aligned} S &\to AB \\ A &\to \varepsilon \mid a \\ B &\to b \end{aligned}$$

It generates the string b by producing ε from A and b from B. Once the rule $A \to \varepsilon$ is eliminated, it is no longer possible to obtain ε from A in the concatenation AB, but the same effect can be achieved by not referring to A at all, that is, by including a new rule $S \to B$, which means essentially AB with A replaced by ε.

$$\begin{aligned} S &\to AB \mid B \\ A &\to a \\ B &\to b \end{aligned}$$

To do that, one first has to determine, which symbols generate ε.

Lemma 2.7. *There exists an algorithm that, given a grammar $G = (\Sigma, N, R, S)$, constructs the set of category symbols generating the empty string.*

$$\textsc{Nullable}(G) = \{\, A \mid A \in N,\ \varepsilon \in L_G(A) \}$$

Proof. Nested set construction:

$$\begin{aligned} \textsc{Nullable}_0(G) &= \varnothing, \\ \textsc{Nullable}_{i+1}(G) &= \{\, A \in N \mid \text{there is a rule } A \to X_1 \dots X_\ell \text{ with } X_1, \dots, X_\ell \in \textsc{Nullable}_i(G) \}. \end{aligned}$$

This sequence actually represents the sequence $\varphi^k(\bot)$ taken modulo $\{\varepsilon\}$. Correctness: $A \in \textsc{Nullable}_k(G)$ if and only if $\varepsilon \in [\varphi^k(\bot)]_A$. $\square$

Lemma 2.8. *Let $G = (\Sigma, N, R, S)$ be a grammar and let $\textsc{Nullable}(G)$ be as in Lemma 2.7. Construct another grammar $G' = (\Sigma, N, R', S)$, where R' contains all rules of the form*

$$A \to X_1 \ldots X_\ell \qquad \text{(if there is a rule } A \to \theta_0 X_1 \theta_1 \ldots X_\ell \theta_\ell \text{ in } R, \text{ with } \ell \geqslant 1 \text{ and } \theta_0, \ldots, \theta_\ell \in \textsc{Nullable}(G)^*\text{)}.$$

Then, for every $A \in N$, $L_{G'}(A) = L_G(A) \setminus \{\varepsilon\}$.

Proof. $\ominus\!\!\!\!\Rightarrow$ Let a string $w \in \Sigma^*$ be representable as $X \in \Sigma \cup N$ in the grammar G', ($\vdash_{G'} X(w)$). It is claimed that w is then representable as X in the grammar G ($\vdash_G X(w)$), and also that $w \neq \varepsilon$. Induction in the number of steps in the proof $\vdash_{G'} X(w)$.

Basis: none. Then $X = w = a$ and $\vdash_G a(a)$, as claimed.

Induction step. Let $X = A \in N$ and consider the last step of the deduction, which is $X_1(w_1), \ldots, X_\ell(w_\ell) \vdash_{G'} A(w)$ for some rule $A \to X_1 \ldots X_\ell \in R'$ and for some partition $w = w_1 \ldots w_\ell$. Such a rule exists, because the original grammar has a rule $A \to \theta_0 X_1 \theta_1 \ldots X_\ell \theta_\ell$ in R, with $\theta_0, \ldots, \theta_\ell \in \textsc{Nullable}(G)^*$. Let $\theta_i = Y_{i,1} \ldots Y_{i,m_i}$ with $Y_{i,j} \in \textsc{Nullable}(G)$. Then, by the definition of $\textsc{Nullable}$, $\vdash_G Y_{i,j}(\varepsilon)$. By the induction hypothesis for $X_i(w_i)$, it follows that $\vdash_G X_i(w_i)$. Using these items as premises allows the following deduction by the rule $A \to \theta_0 X_1 \theta_1 \ldots X_\ell \theta_\ell$:

$$\{X_i(w_i)\}_{i=1}^n, \{Y_{i,j}(\varepsilon)\}_{i,j} \vdash_G A(w).$$

Furthermore, the induction hypothesis asserts that each w_i is non-empty, and since $\ell \geqslant 1$, the string w must be non-empty as well.

$\Leftarrow\!\!\!\!\ominus$ Assume that $\vdash_G X(w)$ and $w \neq \varepsilon$. The claim is $\vdash_{G'} X(w)$, proved by induction on the length of the proof.

Basis: $X = w = a$, $\vdash_G a(a)$ and $\vdash_{G'} a(a)$.

Induction step: ***TBW*** $\square$

2.5.2 Eliminating unit rules

Lemma 2.9. *Let $G = (\Sigma, N, R, S)$ be a grammar, in which all rules are of the form $A \to \alpha$ with $|\alpha| \geqslant 1$, and accordingly $\varepsilon \notin L_G(A)$ for all $A \in N$. Then the grammar $G' = (\Sigma, N, R', S)$, where R' contains a rule of the form $A \to \alpha$ if and only if there is a sequence of rules $A_0 \to A_1$, $A_1 \to A_2$, …, $A_{k-1} \to A_k$, $A_k \to \alpha$ in R, where $A_0 = A$ and $k \geqslant 0$.*

More or less like transformation of ε-NFA to NFA.

Together with Lemma 2.7 and Lemma 2.8, this gives a proof of Theorem 2.7.

2.5.3 The Chomsky normal form

Definition 2.2. *An ordinary grammar $G = (\Sigma, N, R, S)$ is said to be in Chomsky normal form, if each rule in R is of one of the following forms.*

$$\begin{array}{ll} A \to BC & (B, C \in N) \\ A \to a & (a \in \Sigma) \\ S \to \varepsilon & (\textit{only if } S \textit{ never occurs in the right-hand sides of any rules}) \end{array}$$

Actually, Chomsky [5] presented a transformation to a slightly stronger form than the one in Definition 2.2. However, the name *Chomsky normal form* is universally applied to the form given in Definition 2.2.

Theorem 2.7 (Chomsky [5, Thm. 5]). *For every grammar there exists and can be effectively constructed a grammar in the Chomsky normal form that defines the same language.*

Sketch of a proof. Preparation: ensure right-hand sides are at most two symbols long.
Step 1: remove all *null rules* (or ε-*rules*) of the form $A \to \varepsilon$.
Step 2: remove *unit rules* of the form $A \to B$.
Last step: move symbols from Σ to separate rules. □

2.5.4 Left recursion and Greibach normal form

Left dependence $A \to B\alpha$. May end up in a left recursion $A \to B\alpha$, $B \to A\beta$. Direct left recursion $A \to A\alpha$. Relevant for left-to-right parsing. A normal form that rules out left recursion by disallowing any left dependencies.

Definition 2.3. *An ordinary grammar $G = (\Sigma, N, R, S)$ is said to be in the Greibach normal form, if each rule in R is of one of the following forms.*

$$\begin{array}{ll} A \to a\alpha & (a \in \Sigma,\ \alpha \in (\Sigma \cup N)^*) \\ S \to \varepsilon & (\textit{only if } S \textit{ never occurs in the right-hand sides of any rules}) \end{array}$$

Theorem 2.8 (Greibach [12]). *For every ordinary grammar there exists and can be effectively constructed a grammar in the Greibach normal form that describes the same language.*

The transformation works through elementary transformations of two types. The first transformation eliminates direct left recursion for a chosen symbol A.

Lemma 2.10 (Elimination of direct left recursion). *Let*

$$A \to A\alpha_1 \mid \ldots \mid A\alpha_m \mid \beta_1 \mid \ldots \mid \beta_n$$

be all rules for A, where none of $\beta_1, \ldots, \beta_n$ begins with A. Then they can be replaced with the rules

$$\begin{aligned} A &\to \beta_1 \mid \ldots \mid \beta_n \mid \beta_1 A' \mid \ldots \mid \beta_n A' \\ A' &\to \alpha_1 A' \mid \ldots \mid \alpha_m A' \mid \alpha_1 \mid \ldots \mid \alpha_m \end{aligned}$$

where A' is a new category symbol.

Note that this transformation does not introduce any new left dependencies, because the right-hand sides of the new rules for A begin with the same strings $\beta_1, \ldots, \beta_n$ as some of the original rules for A. Furthermore, no symbol in the grammar has a left dependence on A', and therefore any left recursion through A' is impossible.

Example 2.18. *Consider the following left-recursive grammar for arithmetical expressions.*

$$E \to E\underbrace{+E}_{\alpha_1} \mid E\underbrace{*E}_{\alpha_2} \mid \underbrace{(E)}_{\beta_1} \mid \underbrace{1}_{\beta_2}$$

According to Lemma 2.10, it is equivalently transformed to the grammar below.

$$\begin{aligned} E &\to (E) \mid 1 \mid (E)E' \mid 1E' \\ E' &\to +EE' \mid *EE' \mid +E \mid *E \end{aligned}$$

Lemma 2.11 (Substitution)**.** *Let*

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

be all rules for A. Then a rule $B \to \beta A \gamma$, with $B \neq A$ and $\beta, \gamma \in (\Sigma \cup N)^*$, can be replaced with the following collection of rules.

$$B \to \beta\alpha_1\gamma \mid \dots \mid \beta\alpha_m\gamma$$

With the two types of transformation rules defined, the transformation to the Greibach normal form proceeds as follows.

Proof of Theorem 2.8. Assume that the given grammar $G = (\Sigma, N, R, S)$ is already in the Chomsky normal form. The first stage of the construction eliminates left recursion in the grammar. and the following second stage removes all remaining left dependencies.

Fix any enumeration of the category symbols: $N = \{A_1, \dots, A_n\}$. At the **first stage**, these symbols shall be processed one by one. For every subsequent symbol A_i processed, a new symbol A'_i is created, with all rules of the form $A'_i \to \alpha$, where $\alpha \in (\Sigma \cup N)^+$. No symbol has a left dependence on A'_i.

Denote by (A_i, A_j) the condition that the grammar contains any rules of the form $A_i \to A_j\alpha$. Such a pair can be *processed* to remove all these rules as follows. A pair of two identical symbols (A_i, A_i) is processed according to Lemma 2.10, with $A = A_i$: this removes direct left recursion in A_i and creates a new symbol A'_i with the same left dependencies as A_i. Each pair (A_i, A_j), with $i \neq j$, is processed by taking every rule $A_i \to A_j\alpha$ (for any α), and replacing A_j with all its rules using Lemma 2.11. Then the sequence of transformations used to eliminate left recursion from the grammar is given below.

$$\begin{array}{cccccc}
(A_1, A_1), & (A_2, A_1), & (A_3, A_1), & \dots, & (A_{n-1}, A_1), & (A_n, A_1), \\
 & (A_2, A_2), & (A_3, A_2), & \dots, & (A_{n-1}, A_2), & (A_n, A_2), \\
 & & (A_3, A_3), & \dots, & (A_{n-1}, A_3), & (A_n, A_3), \\
 & & & \ddots & & \vdots \\
 & & & & (A_{n-1}, A_{n-1}), & (A_n, A_{n-1}), \\
 & & & & & (A_n, A_n).
\end{array}$$

For example, the first line of the table begins with applying Lemma 2.10 to $A = A_1$, thus eliminating direct left recursion in A_1. Then, for all remaining rules with a left dependence on A_1, which are of the form $A_j \to A_1\alpha$ for $j \in \{2, \dots, n\}$, Lemma 2.11 is applied to substitute $A = A_1$ in this rule. The resulting grammar has no rules of the form $A_i \to A_1\alpha$, for any $i \in \{1, \dots, n\}$.

The second line similarly ensures that there are no rules of the form $A_i \to A_2\alpha$, for any $i \in \{1, \dots, n\}$. This process continues until the last line, which eliminates the last remaining left recursion in A_n. After the last line is processed, all rules with a left dependencies are either of the form $A_i \to A_j\alpha$, with $j > j$, or of the form $A'_i \to A_j\alpha$, for any $i, j \in \{1, \dots, n\}$.

At the **second stage** of the transformation, these left dependencies are eliminated in the following order. The last symbol A_n already has no left dependencies. Then, all left dependencies are removed for A_{n-1} (which can only depend on A_n), then from A_{n-2}, and so on until A_1. After that, all remaining left dependencies are in the rules $A'_i \to A_j\alpha$, which can be removed in any order. □

2.5.5 The Rosenkrantz normal form

A strengthened version of the Greibach normal form, with symbols concatenated on both sides.

Definition 2.4 (Rosenkrantz [19]). *An ordinary grammar $G = (\Sigma, N, R, S)$ is said to be in the Rosenkrantz normal form, if each rule in R is of one of the following forms.*

$$\begin{array}{ll} A \to a\alpha d & (a, d \in \Sigma,\ \alpha \in (\Sigma \cup N)^*) \\ A \to a & (a \in \Sigma) \\ S \to \varepsilon & (\textit{only if } S \textit{ never occurs in the right-hand sides of any rules}) \end{array}$$

Theorem 2.9 (Rosenkrantz [19]). *For every ordinary grammar there exists and can be effectively constructed a grammar in the Rosenkrantz normal form that describes the same language.*

Engelfriet [7] defined a different transformation.

First, need better understanding of Lemma 2.10. It is actually based on the fact that the grammar

$$A \to b \mid Ac,$$

generates the regular language bc^*. The same language is generated by another grammar

$$\begin{aligned} A &\to b \mid bA' \\ A' &\to cA' \mid c \end{aligned}$$

Expanded version of this transformation. First, there is a grammar

$$A \to aA \mid b \mid Ad \mid AcA,$$

and it generates the language $a^*bc^*(da^*bc^*)^*$, which is still regular. There is the following alternative grammar for this language:

$$\begin{aligned} A_{bb} &\to aA_{bb}d \mid aA_{bc}b \mid bA_{cb}d \mid bA_{cc}b \mid b \\ A_{bc} &\to aA_{bc}a \mid aA_{bb}c \mid bA_{cc}a \mid bA_{cb}c \mid bc \mid a \\ A_{cb} &\to dA_{cb}d \mid dA_{cc}b \mid cA_{bb}d \mid cA_{bc}b \mid cb \mid d \\ A_{cc} &\to dA_{cc}a \mid dA_{cb}c \mid cA_{bc}a \mid cA_{bb}c \mid c \end{aligned}$$

Lemma 2.12 (Elimination of direct left and right recursion). *Let $A \in N$, and consider the following general form of the list of rules rules for A, where none of α_i begins with A, none of γ_i ends with A, and none of β_i begins or ends with A.*

$$A \to \alpha_1 A \mid \ldots \mid \alpha_k A \mid \mid \beta_1 \mid \ldots \mid \beta_\ell \mid A\gamma_1 \mid \ldots \mid A\gamma_m \mid A\delta_1 A \mid \ldots \mid A\delta_n A$$

Then these rules can be replaced with the following collection of rules (for all applicable pairs of i and j), where A', A'' and A''' are three new category symbols.

$$\begin{aligned} A &\to \alpha_i A\gamma_j \mid \alpha_i A'\beta_j \mid \beta_i A''\gamma_j \mid \beta_i A'''\beta_j \mid \beta_i \\ A' &\to \alpha_i A'\alpha_j \mid \alpha_i A\delta_j \mid \beta_i A'''\alpha_j \mid \beta_i A''\delta_j \mid \beta_i\delta_j \mid \alpha_i \\ A'' &\to \gamma_i A''\gamma_j \mid \gamma_i A'''\beta_j \mid \delta_i A\gamma_j \mid \delta_i A'\beta_j \mid \delta_i\beta_j \mid \gamma_i \\ A''' &\to \gamma_i A'''\alpha_j \mid \gamma_i A''\delta_j \mid \delta_i A'\alpha_j \mid \delta_i A\delta_j \mid \delta_i \end{aligned}$$

Used in the proof of Theorem 2.9.

2.5.6 Generalized normal form theorem

Theorem 2.10 (Blattner and Ginsburg [3]). *Let $(k_0, \ldots, k_\ell)$ with $\ell \geqslant 2$ be any finite sequence of non-negative integers. Then for every ordinary grammar G there exists and can be effectively constructed an ordinary grammar $G' = (\Sigma, N, R, S)$ generating the same language, in which every rule is of one of the following two forms.*

$$\begin{aligned} &A \to u_0 B_1 u_1 \ldots B_\ell u_\ell && (u_i \in \Sigma^*,\ B_i \in N,\ |u_i| = k_i) \\ &A \to w && (w \in \Sigma^*) \end{aligned}$$

The same result was independently obtained by Maurer, Salomaa and Wood.

Chapter 13

Selected theoretical topics

13.1 Homomorphic characterizations

Representing every ordinary language $L \subseteq \Sigma^*$ as a homomorphic image of an intersection of a Dyck language D_k with a regular language M.

$$L = h(D_k \cap M)$$

In the best versions of this characterization, the homomorphism is symbol-to-symbol, that is, a *renaming* of brackets to the alphabet Σ. Thus, in loose terms, the Chomsky–Schützenberger theorem states that *every ordinary language is a regular structure of well-nested brackets, renamed to the target alphabet.*

13.1.1 The Chomsky–Schützenberger theorem in the even form

All strings in the Dyck language are of even length, and so are their images under any symbol-to-symbol homomorphisms. Therefore, the most obvious version of the Chomsky–Schützenberger theorem applies only to languages of strings of an even length. This is a technical restriction that will be lifted in the subsequent variants of this theorem.

For any number $k \geqslant 1$, consider the alphabet of k pairs of brackets,

$$\Omega_k = \{a_1, b_1, \ldots a_k, b_k\},$$

where a_i represents a left bracket and b_i is its matching right bracket. Define the Dyck language on k pairs of brackets, $D_k \subseteq \Omega_k^*$. by the following grammar.

$$S \to SS \mid a_1 S b_1 \mid \ldots \mid a_k S b_k \mid \varepsilon$$

Theorem 13.1. *A language $L \subseteq (\Sigma^2)^*$ is ordinary if and only if there exists a number $k \geqslant 1$, a regular language $R \subseteq \Omega_k^*$ and a symbol-to-symbol homomorphism $h \colon \Omega_k \to \Sigma$, such that $L = h(D_k \cap M)$.*

The converse implication of Theorem 13.1 can be taken for granted, because the ordinary languages are closed under homomorphisms (Theorem 2.4) and under intersection with regular languages (Theorem 2.3). The task is to prove the forward implication of the theorem, that is, to construct k, R and h for a given language L. It is convenient to assume that the language L is described by a grammar in the following special variant of the Rosenkrantz normal form.

Lemma 13.1. *Every ordinary language $L \subseteq (\Sigma^2)^+$ is generated by a grammar (Σ, N, R, S) with all rules of the form*

$$A \to bC_1 \ldots C_\ell d \qquad\qquad (b, d \in \Sigma,\ \ell \geqslant 0,\ C_1, \ldots, C_\ell \in N),$$

where the symbols $C_1, \ldots, C_\ell$ in every such rule are pairwise distinct.

Proof. Let G be a grammar generating the given ordinary language $L \subseteq (\Sigma^2)^+$. Consider the alphabet $\Sigma \times \Sigma = \{ (a,b) \mid a,b \in \Sigma\}$ and define the homomorphism $h\colon (\Sigma \times \Sigma)^* \to \Sigma^*$ by $h\big((a,b)\big) = ab$. Then, by the closure of the ordinary languages under inverse homomorphisms (Theorem 2.6), there exists a grammar G' over the alphabet $\Sigma \times \Sigma$ generating the language

$$L(G') = h^{-1}(L(G)) = \{ (a_1,b_1)(a_2,b_2)\dots(a_n,b_n) \mid a_1b_1a_2b_2\dots a_nb_n \in L(G)\}.$$

By Theorem 2.9, this grammar can be transformed to a grammar G'' generating the same language over $\Sigma \times \Sigma$, with all rules of the form

$$A \to (b,b')C_1 \dots C_\ell(d,d') \qquad (b,b',d,d' \in \Sigma,\ \ell \geqslant 0,\ C_1,\dots,C_\ell \in N) \tag{13.1a}$$
$$A \to (a,a') \qquad (a,a' \in \Sigma) \tag{13.1b}$$

Construct a grammar G''' over the alphabet Σ, with the following rules:

$$A \to bb'C_1 \dots C_\ell dd', \qquad \text{for each “long” rule (13.1a) in } G'', \tag{13.2a}$$
$$A \to aa', \qquad \text{for each “short” rule (13.1b) in } G''. \tag{13.2b}$$

By construction, $L(G''') = h(L(G''))$, and hence $L(G''') = h(h^{-1}(L(G))) = L(G)$.

Once each “long” rule (13.2a) in G''' is replaced with two rules $A \to bXd'$ and $X \to b'C_1 \dots C_\ell d$, the resulting grammar still generates $L(G)$ and all rules are of the desired form.

Finally, the requirement that every such rule has pairwise distinct symbols $C_1,\dots,C_\ell$, can be met by making duplicate copies of any repeated category symbols. That is, if a rule refers to two instances of C, then the second instance can be replaced with a new symbol C', which has exactly the same rules as C. This completes the construction. $\square$

Consider an arbitrary language $L \subseteq (\Sigma^2)^*$. By Lemma 13.1, the language $L \setminus \{\varepsilon\}$ is described by a grammar $G = (\Sigma, N, R, S)$ with each rule of the form

$$A \to bC_1 \dots C_\ell d \qquad (b,d \in \Sigma,\ \ell \geqslant 0,\ C_1,\dots,C_\ell \in N),$$

where the symbols $C_1 \dots C_\ell$ are pairwise distinct. Due to the latter condition, given a rule $A \to bC_1 \dots C_\ell d$ and one of the symbols C_i, one can identify the position i.

Each pair of brackets in D_k is labelled with two rules of the grammar: *the current rule* $A \to bC_1 \dots C_\ell d$ and *the previous rule* $X \to \xi$, where ξ contains an instance of A. If the current rule is the rule for the initial symbol used at the root of the parse tree, then the previous rule is replaced with a dash $(-)$. Formally, one can set $k = (|R|+1)\cdot|R|$ and consider the Dyck language D_k. For the rest of this argument, a left and a right bracket of this kind shall be denoted by $(^{X\to\xi}_{A\to bC_1\dots C_\ell d}$ and $)^{X\to\xi}_{A\to bC_1\dots C_\ell d}$, respectively (rather than by a_i and b_i for some i).

Define the regular language R^0_G as the set of all strings $w \in \big(\Omega_k\big)^*$ that have all 2-symbol substrings of the following form: for some rule $A \to bC_1 \dots C_\ell d \in R$ with $k \geqslant 1$ and for $\Xi \in R \cup \{-\}$ being either a rule referring to A or “$-$”,

$$(^{\Xi}_{A\to bC_1\dots C_\ell d}(^{A\to bC_1\dots C_\ell d}_{C_1\to\gamma_1}, \qquad \text{with } C_1 \to \gamma_1 \in R,$$
$$)^{A\to bC_1\dots C_\ell d}_{C_i\to\gamma_i}(^{A\to bC_1\dots C_\ell d}_{C_{i+1}\to\gamma_{i+1}}, \qquad \text{with } i \in \{1,\dots,k-1\},\ C_i \to \gamma_i, C_{i+1} \to \gamma_{i+1} \in R,$$
$$)^{A\to bC_1\dots C_\ell d}_{C_\ell\to\gamma_k})^{\Xi}_{A\to bC_1\dots C_\ell d}, \qquad \text{with } C_\ell \to \gamma_k \in R,$$

or, for some rule $A \to bd \in R$, and for $\Xi \in R \cup \{-\}$ as above,

$$(^{\Xi}_{A\to bd})^{\Xi}_{A\to bd}.$$

Let R_G be the subset of R_G^0 containing the strings that begin with a symbol $(^{-}_{S\to\sigma}$ and end with a symbol $)^{-}_{S\to\sigma}$. Furthermore, if $\varepsilon \in L$, then R_G contains ε as well.

Define the homomorphism $h\colon \Omega_k \to \Sigma$ as follows:

$$h\Big((^{\Xi}_{A\to bC_1\ldots C_\ell d}\Big) = b \qquad\qquad h\Big()^{\Xi}_{A\to bC_1\ldots C_\ell d}\Big) = d$$

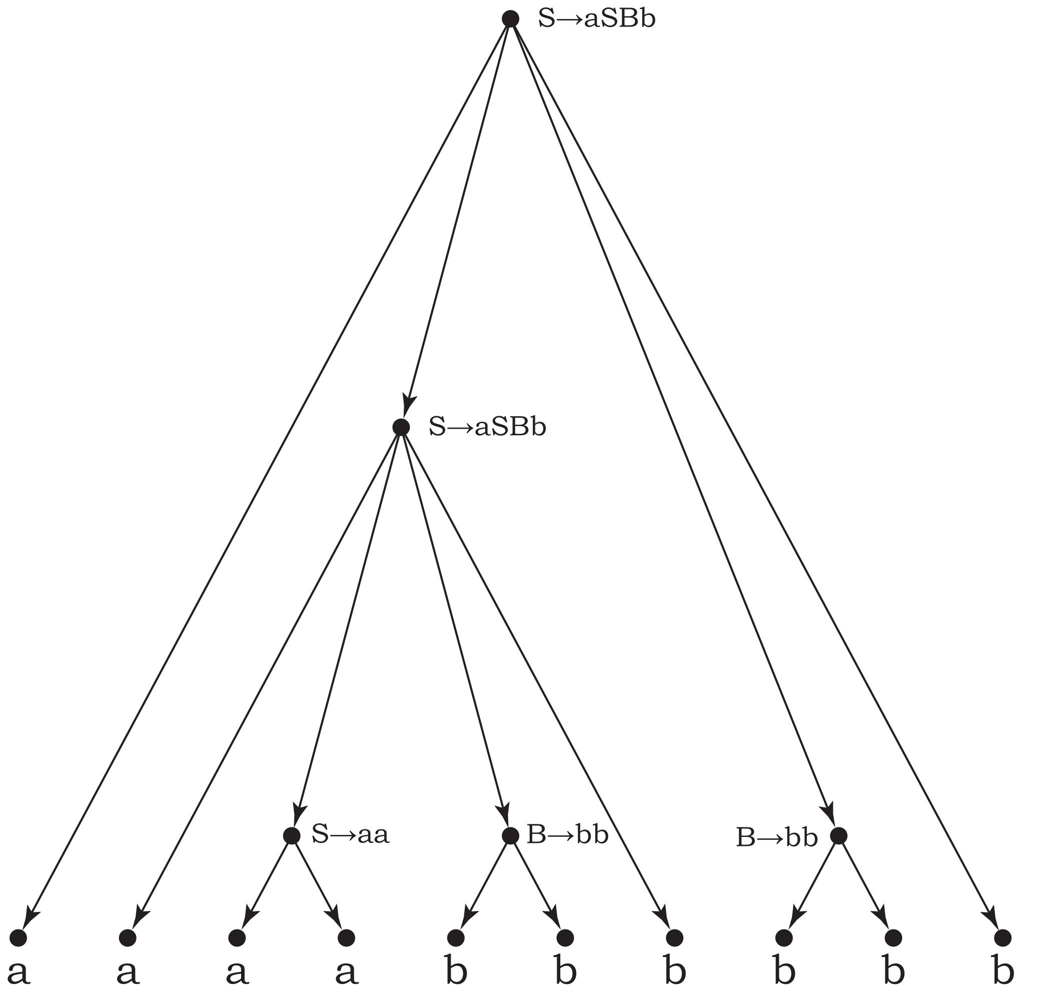
Example 13.1. *Consider the grammar*

$$\begin{aligned} S &\to aSBb \mid aa \\ B &\to bb \end{aligned}$$

generating the language $\{\, a^{n+2}b^{3n} \mid n \geqslant 0\}$. *Then the string* $w = aaaabbbbbb$ *is obtained as the homomorphic image of the following sequence of brackets in* $\Omega_k \cap M_G$:

$$(^{-}_{S\to aSBb}\ (^{S\to aSBb}_{S\to aSBb}\ (^{S\to aSBb}_{S\to aa}\ )^{S\to aSBb}_{S\to aa}\ (^{S\to aSBb}_{B\to bb}\ )^{S\to aSBb}_{B\to bb}\ )^{S\to aSBb}_{S\to aSBb}\ (^{S\to aSBb}_{B\to bb}\ )^{S\to aSBb}_{B\to bb}\ )^{-}_{S\to aSBb}$$

Figure 13.1 illustrates how this sequence of brackets encodes the parse tree of w.



$$(^{-}_{S\to \mathbf{a}SBb}\ (^{S\to a\mathbf{S}Bb}_{S\to \mathbf{a}SBb}\ (^{S\to a\mathbf{S}Bb}_{S\to \mathbf{a}a}\ )^{S\to a\mathbf{S}Bb}_{S\to a\mathbf{a}}\ (^{S\to aS\mathbf{B}b}_{B\to \mathbf{b}b}\ )^{S\to aS\mathbf{B}b}_{B\to b\mathbf{b}}\ )^{S\to a\mathbf{S}Bb}_{S\to aSB\mathbf{b}}\ (^{S\to aS\mathbf{B}b}_{B\to \mathbf{b}b}\ )^{S\to aS\mathbf{B}b}_{B\to b\mathbf{b}}\ )^{-}_{S\to aSB\mathbf{b}}$$

Figure 13.1: The parse tree of a string in Example 13.1, encoded in a sequence of brackets.

It remains to prove that the constructed Dyck language D_k, regular language R_G and homomorphism h satisfy the equality $h(D_k \cap M_G) = L$. This is achieved in the below series of statements.

Claim 13.1.1. *Let $x = (^{\Xi}_{A\to\alpha} y)^{\Xi}_{A\to\alpha} \in R^0_G$, where $y \in D_k$. Then, $h(x) \in L_G(\alpha)$.*

Proof. Induction on the length of x.

Basis: $|x| = 2$. Then the rule $A \to \alpha$ must have $\alpha = bd$ with $b, d \in \Sigma$, and accordingly $x = (^{\Xi}_{A\to bd})^{\Xi}_{A\to bd}$. Then $h(x) = bd$, and the condition $h(x) \in L_G(\alpha)$ holds true.

Induction step. Let $x = (^{\Xi}_{A\to bC_1\ldots C_\ell d} y)^{\Xi}_{A\to bC_1\ldots C_\ell d}$ with $|y| > 0$. Since $y \in D_k$, it is a concatenation $y = y_1 \ldots y_\ell$ of $\ell \geqslant 1$ non-empty strings from D_k, none of which can be further factored into elements of D_k. Because $x \in R^0_G$ and its first symbol is $(^{\Xi}_{A\to bC_1\ldots C_\ell d}$, its next symbol, which is the first symbol of y_1, must be $(^{A\to bC_1\ldots C_\ell d}_{C_1\to\gamma_1}$, for some rule $C_1 \to \gamma_1 \in R$. Then the last symbol of y_1 is $)^{A\to bC_1\ldots C_\ell d}_{C_1\to\gamma_1}$, and the next symbol of x must be $(^{A\to bC_1\ldots C_\ell d}_{C_2\to\gamma_2}$ for some rule $C_2 \to \gamma_2 \in R$. This symbol is accordingly the first symbol of y_2. Continuing the same argument leads to $y_i = (^{A\to bC_1\ldots C_\ell d}_{C_i\to\gamma_i} z_i)^{A\to bC_1\ldots C_\ell d}_{C_i\to\gamma_i}$ for all $i \in \{1, \ldots, k\}$; applying the induction hypothesis to each y_i gives that $h(y_i) \in L_G(\gamma_i)$.

Finally, the condition $x \in R^0_G$ ensures that the last symbol of y_k, which is $)^{A\to bC_1\ldots C_\ell d}_{C_\ell\to\gamma_k}$, must be followed by a symbol $)^{\Xi'}_{A\to bC_1\ldots C_\ell d}$, for some $\Xi' \in R\cup\{-\}$. Since this cannot be the first symbol of y_{k+1}, it follows that $\ell = k$, $\Xi' = \Xi$ and $x = (^{\Xi}_{A\to bC_1\ldots C_\ell d} y_1 \cdots y_k)^{\Xi}_{A\to bC_1\ldots C_\ell d}$. Therefore,

$$h(x) = b \cdot h(y_1) \cdot \ldots \cdot h(y_k) \cdot d \in L_G(b\gamma_1 \ldots \gamma_k d) \subseteq L_G(bC_1 \ldots C_\ell d),$$

as claimed. □

Claim 13.1.2. *If $w \in L_G(\alpha)$ for some rule $A \to \alpha$ and $\Xi \in R \cup \{-\}$, then $w = h(x)$ for some string $x = (^{\Xi}_{A\to\alpha} y)^{\Xi}_{A\to\alpha} \in R^0_G$ with $y \in D_k$.*

Proof. Induction on the length of w. Let $\alpha = bC_1 \ldots C_\ell d$ and accordingly let $w = bu_1 \ldots u_k d$ with $u_i \in L_G(C_i)$. For each $i \in \{1, \ldots, k\}$, let $C_i \to \gamma_i$ with $u_i \in L_G(\gamma_i)$ be the rule used to generate u_i. By the induction hypothesis, each u_i is representable as $h(y_i)$, for $y_i = (^{A\to bC_1\ldots C_\ell d}_{C_i\to\gamma_i} z_i)^{A\to bC_1\ldots C_\ell d}_{C_i\to\gamma_i} \in R^0_G$ with $z_i \in D_k$.

Define $y = y_1 \ldots y_k$. Then $y \in D_k$, because it is a concatenation of k elements of D_k, each enclosed in a pair of matching brackets. To see that $x \in R^0_G$, it is sufficient to check all two-symbol substrings at the boundaries of y_i. If $k \geqslant 1$, these are $(^{\Xi}_{A\to bC_1\ldots C_\ell d}(^{A\to bC_1\ldots C_\ell d}_{C_1\to\gamma_1}$, $)^{A\to bC_1\ldots C_\ell d}_{C_i\to\gamma_i}(^{A\to bC_1\ldots C_\ell d}_{C_{i+1}\to\gamma_{i+1}}$ for all $i \in \{1, \ldots, k-1\}$, and $)^{A\to bC_1\ldots C_\ell d}_{C_i\to\gamma_i})^{\Xi}_{A\to bC_1\ldots C_\ell d}$, and all of them are allowed by the definition of R^0_G. If $k = 0$, there is a unique substring $(^{\Xi}_{A\to bd})^{\Xi}_{A\to bd}$, which is also allowed. □

Now the statement of Theorem 13.1 is proved as follows.

If a non-empty string w is in in $L(G)$, then there is a rule $S \to \sigma$ with $w \in L_G(\sigma)$, and, by Claim 13.1.2, $w = h(x)$ for some $x = (^{-}_{S\to\sigma} y)^{-}_{S\to\sigma} \in D_k \cap M^0_G$. By its first and its last symbol, x is in R_G. Accordingly, $w \in h(D_k \cap M_G)$.

Conversely, assume that $w \in h(x)$ for some $x \in D_k \cap M_G$. Then the first symbol of x is $(^{-}_{S\to\sigma}$, its last symbol is $)^{-}_{S\to\sigma}$, and these two symbols have to be matched to each other, because R^0_G does not allow such symbols in the middle of a string. Then $x = (^{-}_{S\to\sigma} y)^{-}_{S\to\sigma}$ for some $y \in D_k$. By Claim 13.1.1 for this string, $h(x) \in L_G(\sigma) \subseteq L(G)$.

Finally, by definition, the empty string is in R_G if and only if it is in L. Altogether, $h(D_k \cap M_G) = L$, as claimed.

13.1.2 Further forms of the Chomsky–Schützenberger theorem

For arbitrary languages, where the length of the strings is not restricted to be even, this theorem has several possible statements.

One of them relies on a variant of the Dyck language equipped with neutral symbols. For any numbers k and ℓ, this language, $\widehat{D}_{k,\ell}$, is defined over the alphabet

$$\Omega_{k,\ell} = \{a_1, b_1, \ldots, a_k, b_k, c_1, \ldots, c_\ell\}$$

by the following grammar:

$$S \to SS \mid a_1Sb_1 \mid \ldots \mid a_kSb_k \mid c_1 \mid \ldots \mid c_\ell \mid \varepsilon$$

Denote $\widehat{D}_{k,\ell} = \widehat{D}_{\{1,\ldots,k\},\,\{1,\ldots,\ell\}}$ and let $\Omega_{k,\ell}$ be the alphabet, over which it is defined.

Theorem 13.2 (The Chomsky–Schützenberger theorem in the form with neutral symbols)**.** *A language $L \subseteq \Sigma^*$ is ordinary if and only if there exists numbers $k, \ell \geqslant 1$, a regular language $R \subseteq \Omega_{k,\ell}^*$ and a symbol-to-symbol homomorphism $h\colon \Omega_{k,\ell} \to \Sigma$, such that $L = h(\widehat{D}_{k,\ell} \cap M)$.*

Another variant retains the Dyck language without neutral symbols, but at the expence of using erasing homomorphisms.

Theorem 13.3 (The Chomsky–Schützenberger theorem in the erasing form)**.** *A language $L \subseteq \Sigma^*$ is ordinary if and only if there exists a number k, a regular language $R \subseteq \Omega_k^*$ and homomorphism $h\colon \Omega_k \to \Sigma^*$, such that $L = h(D_k \cap M)$.*

Originally, Chomsky and Schützenberger [6] proved the theorem in the erasing form, and used a proof in which every right bracket b_i was erased.

Finally, there is another statement using the standard Dyck language without neutral symbols and *non-erasing homomorphisms.* This variant has to exclude one-symbol strings, because they cannot be obtained as non-erasing homomorphic images of any strings in the Dyck language.

Theorem 13.4. *A language $L \subseteq \Sigma^* \setminus \Sigma$ is ordinary if and only if there exists a number k, a regular language $R \subseteq \Omega_k^*$ and a non-erasing homomorphism $h\colon \Omega_k \to \Sigma^+$, such that $L = h(D_k \cap M)$.*

13.2 Inverse homomorphic characterizations

13.2.1 Greibach's "hardest language"

Theorem 13.5 (Greibach [13])**.** *There exist such an alphabet Σ_0 and such an ordinary language $L_0 \subseteq \Sigma_0^*$, that for every ordinary language L over any alphabet Σ there is such a homomorphism $h\colon \Sigma \to \Sigma_0^*$, that $L = h^{-1}(L_0)$ if $\varepsilon \notin L$ and $L = h^{-1}(L_0 \cup \{\varepsilon\})$ if $\varepsilon \in L$.*

The image of every symbol $a \in \Sigma$ under the homomorphism h will contain basically the entire grammar for the language L, and the language L_0 will check the rules of the grammar given within the images of symbols.

Let $G = (\Sigma, N, R, S)$ be any grammar in the Greibach normal form that defines the language L. Assume that $N = \{A_1, A_2, \ldots, A_{|N|}\}$, $S = A_1$ and each rule in R is of the form

$$A_i \to sA_{j_1} \ldots A_{j_\ell} \qquad (s \in \Sigma,\ \ell \geqslant 0,\ i, j_1, \ldots, j_\ell \in \{1, \ldots, |N|\}) \tag{13.3}$$

Let $\Sigma_0 = \{a, b, c, d, \#\}$ be the alphabet used for encoding grammars. The symbols in Σ_0 have the following meaning.

- The symbol a is used to represent each symbol A_j in the right-hand side of any rule as a^j.

- The symbol b is used to denote A_i in the left-hand side of a rule as b^i.
- The symbol c represents concatenation in the right-hand side of any rule, so that a rule (13.3) is represented by the following string in $\{a, b, c\}^*$:

$$\sigma(A_i \to sA_{j_1} \dots A_{j_\ell}) = b^i ca^{j_\ell} ca^{j_{\ell-1}} c \dots ca^{j_2} ca^{j_1}.$$

- The symbol d is used to separate multiple rules with the right-hand sides beginning with the same symbol s.
- The image $h(s)$ of any symbol $s \in \Sigma$ is concluded with the separator symbol $\# \in \Sigma_0$.

Consider a symbol $s \in \Sigma$, and let $\{\rho_1, \dots \rho_k\} \subseteq R$ be all rules with the right-hand side beginning with s. Then define the image of s under h as

$$h_G(s) = \sigma(\rho_1) d\sigma(\rho_2) d \dots d\sigma(\rho_k)\#,$$

and if $k = 0$, then let $h_G(s) = \#$.

Example 13.2. *Let $\Sigma = \{s, t\}$ and consider a grammar with the rules*

$$\begin{aligned} A_1 &\to sA_1A_2 \mid t \\ A_2 &\to tA_2 \mid t \end{aligned}$$

which generates the language $L = \{\, s^m t^n \mid 0 \leqslant m < n\}$. Then

$$\begin{aligned} h_G(s) &= bca^2ca\# \qquad \textit{and} \\ h_G(t) &= bdb^2ca^2db^2\#, \end{aligned}$$

and accordingly the string $sttt \in L$ has the following image:

$$h_G(sttt) = bca^2ca\#bdb^2ca^2db^2\#bdb^2ca^2db^2\#bdb^2ca^2db^2\#.$$

This string over Σ_0 contains sufficient information to verify that $sttt$ is in $L(G)$.

The goal is to construct a grammar that will generate the image $h_G(w)$ of a string $w \in \Sigma^*$ if and only if $w \in L(G)$. This grammar has to check the following condition.

Definition 13.1. *Let a string over $\Sigma_0 \in \{a, b, c, d, \#\}$ be called m-**correct**, with $m \geqslant 1$, if it is of the form $x_1dx_2d \dots dx_k\#y_2\# \dots \#y_n\#$, where $k, n \geqslant 1$, $x_i \in \{a, b, c\}^+$, $y_j \in \{a, b, c, d\}^+$ and there exists a number $i \in \{1, \dots, k\}$, for which $x_i = b^m a^{m_\ell} ca^{m_{\ell-1}} c \dots ca^{m_1}$ and there are numbers $1 = n_1 < n_2 < \dots < n_\ell < n_{\ell+1} = n$ satisfying the following: for each $j \in \{1, \dots, \ell\}$ the substring $y_{n_j+1}\# \dots y_{n_{j+1}}\#$ is m_j-correct.*

Define $L_0 \subseteq \Sigma_0^*$ as the language of all 1-correct strings.

Lemma 13.2. *The language L_0 is generated by the following grammar.*

$$\begin{aligned} S &\to XdS \mid bA\# \\ X &\to aX \mid bX \mid cX \mid a \mid b \mid c \\ A &\to cCA \mid B \\ C &\to aCb \mid aDb \\ D &\to DXd \mid A\# \\ B &\to dXB \mid \varepsilon \end{aligned}$$

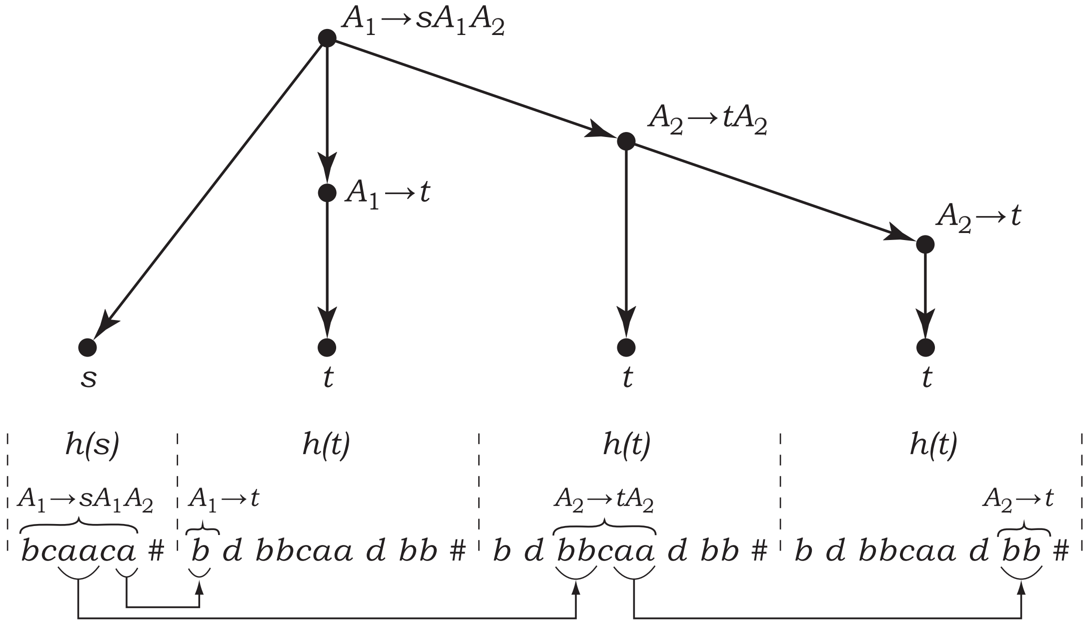


Figure 13.2: A parse of a string $w = sttt$ decoded from its homomorphic image $h_G(sttt)$, as in Example 13.2.

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] M. Blattner, S. Ginsburg, "Position-restricted grammar forms and grammars", *Theoretical Computer Science*, 17:1 (1982), 1–27.

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

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

[6] 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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Index