Contents

Chapter 4

Algebraic Properties of Language Families

In this section we study the behaviour of languages under certain operations. Especially, we are interested in the question whether or not the application of some operation to languages of some language family yields a language of that family, again. The result will be used to present some characterizations of language families by operations. In addition, we also give a characterization of the set of regular languages by properties of associated congruence classes.

4.1 Closure Properties of Language Families

The basic definition for the behaviour of language families with respect to operation is the following one.

Definition 4.1 *We say that a family $\mathcal{L}$ of languages is closed under the n-ary operation τ if, for any languages $L_1, L_2, \ldots, L_n$ of $\mathcal{L}$, $\tau(L_1, L_2, \ldots, L_n) \in \mathcal{L}$.*

We first study the closure properties of the families of the Chomsky hierarchy under set-theoretic operations.

Lemma 4.2 *The families $\mathcal{L}(REG)$, $\mathcal{L}(LIN)$, $\mathcal{L}(CF)$, $\mathcal{L}(CS)$ and $\mathcal{L}(RE)$ are closed under union.*

Proof. Let L_1 and L_2 are two languages in $\mathcal{L}(X)$ with $X \in \{REG, LIN, CF, CS, RE\}$. Then there are grammars $G_1 = (N_1, T_1, P_1, S_1)$ and $G_2 = (N_2, T_2, P_2, S_2)$ of type X such that $L(G_1) = L_1$ and $L(G_2) = L_2$). Without loss of generality we assume that $N_1 \cap N_2 = \emptyset$ (if this should not be the case we rename some nonterminals such that the required emptiness is obtained). We construct the grammar

$$G = (\{S\} \cup N_1 \cup N_2, T_1 \cup T_2, \{S \to S_1, S \to S_2\} \cup P_1 \cup P_2, S),$$

where S is a new symbol not contained in $N_1 \cup N_2 \cup T_1 \cup T_2$. Obviously, G is of type X, too. Moreover, any derivation has the form

$$S \Longrightarrow S_i \overset{*}{\underset{G_i}{\Longrightarrow}} w \in L(G_i)$$

for some $i \in \{1, 2\}$ (because by $N_1 \cup N_2 = \emptyset$ the rules of P_i do not produce nonterminals of N_j, $j \neq i$, i. e., we cannot merge the productions of P_1 and P_2. Thus we can derive only words and all words of $L(G_1) \cup L(G_2) = L_1 \cup L_2$. Therefore $L_1 \cup L_2 = L(G) \in \mathcal{L}(X)$. $\square$

Lemma 4.3 *The families $\mathcal{L}(REG)$, $\mathcal{L}(CS)$ and $\mathcal{L}(RE)$ are closed under intersection. The families $\mathcal{L}(LIN)$ and $\mathcal{L}(CF)$ are not closed under intersection.*

Proof. $\mathcal{L}(REG)$. We have to show that, for two regular languages L_1 and L_2, their intersection $L_1 \cap L_2$ is a regular language, too. We only give the proof for the case that $\lambda \notin L_1 \cap L_2$ and leave the modifications for the general case to the reader.

Let
$$G_1 = (N_1, T_1, P_1, S_1) \quad \text{and} \quad G_2 = (N_2, T_2, P_2, S_2)$$
be two regular grammars with
$$L(G_1) = L_1 \quad \text{and} \quad L(G_2) = L_2.$$
By Theorem 2.28, we can assume that both grammar are in the normal form, i. e., the rules have the form $A \to aB$ or $A \to a$ with nonterminals A, B and terminal a. We consider the regular grammar
$$G = (N_1 \times N_2, T, P, (S_1, S_2))$$
with
$$\begin{aligned} P \;=\; & \{(A_1, B_1) \to a(A_2, B_2) : A_1 \to aA_2 \in P_1, B_1 \to aB_2 \in P_2\} \\ & \cup\{(A, B) \to a : A \to a \in P_1, B \to a \in P_2\}. \end{aligned}$$
It is easy to see that a derivation
$$(S_1, S_2) \Longrightarrow a_1(A_1, B_1) \Longrightarrow a_1a_2(A_2, B_2) \Longrightarrow \ldots \Longrightarrow a_1a_2 \ldots a_{n-1}(A_{n-1}, B_{n-1}) \Longrightarrow a_1a_2 \ldots a_{n-1}a_n$$
exists in G if and only derivations
$$S_1 \Longrightarrow a_1A_1 \Longrightarrow a_1a_2A_2 \Longrightarrow \ldots \Longrightarrow a_1a_2 \ldots a_{n-1}A_{n-1} \Longrightarrow a_1a_2 \ldots a_{n-1}a_n$$
and
$$S_2 \Longrightarrow a_1B_1 \Longrightarrow a_1a_2A_2 \Longrightarrow \ldots \Longrightarrow a_1a_2 \ldots a_{n-1}B_{n-1} \Longrightarrow a_1a_2 \ldots a_{n-1}a_n$$
exist in G_1 and G_2, respectively. Therefore $w \in L(G)$ holds if and only $w \in L(G_1)$ and $w \in L(G_2)$. Hence
$$L(G) = L(G_1) \cap L(G_2) = L_1 \cap L_2.$$
Since G is a regular grammar, $L_1 \cap L_2$ is a regular languages.

$\mathcal{L}(RE)$. Let $L_1 \in \mathcal{L}(RE)$ and $L_2 \in \mathcal{L}(RE)$ be given. By Theorem 3.19, there are deterministic Turing machines
$$M_1 = (X, Z_1, z_{01}, Q_1, \delta_1, Q_1) \quad \text{and} \quad M_2 = (X, Z_2, z_{02}, Q_2, \delta_2, Q_2)$$

with

$$T(M_1) = L_1 \quad \text{and} \quad T(M_2) = L_2.$$

Without loss of generality we can assume that $Z_1 \cap Z_2 = \emptyset$. We construct a Turing machine M which works as follows (the formal description is left to the reader). First the machine replaces any letter x of the input word by (x, x). Then it works as M_1 using only the letters of the first components; thus the input input word is stored in the second component (if a $*$ is read, then it is handled as $(*, *)$). If M reaches a state from Q_1, then it replaces all letters (a, b) by their second component b, i. e., the input word is at the tape, again. Now M starts to work as M_2 and stops if a state of Q_2 is reached.

According to this work we first check whether the input is accepted by M_1 and then whether the input is in $T(M_2)$. Thus M accepts a word W if and only w is accepted by M_1 as well as by M_2. Consequently,

$$T(M) = T(M_1) \cap T(M_2) = L_1 \cap L_2,$$

which proves that $L_1 \cap L_2 \in \mathcal{L}(RE)$ by Theorem 3.19.

$\mathcal{L}(CS)$. The proof can be given analogously to that for recursively enumerable languages, but we use linearly bounded automata and Theorem 3.23.

$\mathcal{L}(LIN)$ and $\mathcal{L}(CF)$. In order to prove the assertion it is sufficient to give two linear languages which have a non-context-free intersection. We consider the linear grammars

$$\begin{aligned} G_1 &= (\{S, A\},\ \{a, b, c\},\ \{S \to Sc,\ S \to Ac,\ A \to aAb,\ A \to ab\}, S), \\ G_1 &= (\{S, A\},\ \{a, b, c\},\ \{S \to aS,\ S \to aA,\ A \to bAc,\ A \to bc\}, S). \end{aligned}$$

It is easy to see that

$$L(G_1) = \{a^n b^n c^m \mid n \geq 1,\ m \geq 1\} \text{ and } L(G_2) = \{a^m b^n c^n \mid n \geq 1,\ m \geq 1\}.$$

Obviously, $L(G_1) \cap L(G_2) = \{a^n b^n c^n \mid n \geq 1\}$. By the proof of Theorem 16.13 we know that $L(G_1) \cap L(G_2)$ is not context-free. □

Lemma 4.4 *The families $\mathcal{L}(REG)$ and $\mathcal{L}(CS)$ are closed under complement. The families $\mathcal{L}(LIN)$, $\mathcal{L}(CF)$ and $\mathcal{L}(RE)$ are not closed under complement.*

Proof. $\mathcal{L}(REG)$. Let L be a regular language. Then there is a deterministic finite automaton $\mathcal{A} = (alph(L), Z, z_0, F, \delta)$ such that $L = T(\mathcal{A})$. Thus $w \in L$ if and only if $\delta^*(z_0, w) \in F$. Consequently, $w \in C(L)$ if and only if $\delta^*(z_0, w) \notin F$ if and only if $\delta^*(z_0, w) \in Z \setminus F$. Thus the automaton $\mathcal{A}' = ((alph(L), Z, z_0, Z \setminus F, \delta)$ accepts $C(L)$. Therefore $C(L)$ is regular.

$\mathcal{L}(CS)$. We omit the proof since it requires some knowledge not presented in this book and is relatively long. We refer to [32] and [17] and the original papers [13], [29].

$\mathcal{L}(RE)$. If $\mathcal{L}(RE)$ is closed under complement, then any recursively-enumerable language is recursive by Theorem 3.10, in contradiction to Theorem 3.11.

$\mathcal{L}(CF)$. Let us assume that $\mathcal{L}(CF)$ is closed under complement. Let L_1 and L_2 be two arbitrary context-free languages. We set

$$X = alph(L_1) \cup alph(L_2), \quad X_1 = X \setminus alph(L_1), \quad X_2 = X \setminus alph(L_2).$$

Let R_1 and R_2 be the sets of all words over X which contain at least one letter of X_1 and X_2, respectively. If $X_i = \emptyset$ for some $i \in \{1,2\}$, then R_i is the empty set, and therefore R_i is a regular set. If $X_i \neq \emptyset$, then the regular grammar

$$G_i = (\{S,A\}, X, \bigcup_{a\in alph(L_i)} \{S \to aS\} \cup \bigcup_{b\in X_i} \{S \to bA, S \to b\} \cup \bigcup_{x\in X} \{A \to xA, A \to x\}, S)$$

generates R_i (since we can only terminate from S or switch from S to A, if a letter from X_i is generated). Hence in all cases R_1 and R_2 are regular languages and therefore context-free, too. By our assumption and Lemma 4.2, for $i \in \{1,2\}$,

$$X^* \setminus L_i = ((alph(L_i))^* \setminus L_i) \cup R_i = C(L_i) \cup R_i$$

is a context-free language. Again, by Lemma 4.2,

$$R = (X^* \setminus L_1)) \cup (X^* \setminus L_2)$$

is context-free. Now our assumption gives the context-freeness of

$$L_1 \cap L_2 = X^* \setminus ((X^* \setminus L_1) \cup (X^* \setminus L_2)) = (alph(R))^* \setminus R$$

is a context-free languages, which means that the intersection of arbitrary context-free languages is context-free. Thus we have a contradiction to Lemma 4.3. Therefore our assumption is not valid, i. e. $\mathcal{L}(CF)$ is not closed under complement.

$\mathcal{L}(LIN)$ We repeat the proof for $\mathcal{L}(CF)$ (word by word), but replace context-free in all cases by linear and $\mathcal{L}(CF)$ by $\mathcal{L}(LIN)$. □

Lemma 4.5 *The families $\mathcal{L}(REG)$ and $\mathcal{L}(CS)$ are closed under set-theoretic difference. The families $\mathcal{L}(LIN)$, $\mathcal{L}(CF)$ and $\mathcal{L}(RE)$ are not closed under set-theoretic difference.*

Proof. Let X and Y be two languages and $V = alph(X) \cup alph(Y)$. Let us assume that $alph(X) \setminus alph(Y)$ is not empty (the easy modifications for $alph(X) \subseteq alph(Y)$ are left to the reader). From the proof of Lemma 4.4, we know that $V^* \setminus (alph(Y))^*$ is in $\mathcal{L}(REG)$ and therefore in $\mathcal{L}(CS)$, too. Because

$$\begin{aligned} X \setminus Y &= (V^* \setminus Y) \cap X = ((V^* \setminus (alph(Y))^*) \cup ((alph(Y))^* \setminus Y)) \cap X \\ &= ((V^* \setminus (alph(Y))^*) \cup C(Y)) \cap X, \end{aligned}$$

the first assertion follows by Lemmas 4.2 – 4.4.

Since X^* is a regular language and belongs to all language families under consideration, the complement is a special case of difference. Thus the second statement of Lemma 4.4 implies the second assertion. □

We now mention a special case of intersection; we require that the language of the family under consideration has to intersected with a regular set.

Lemma 4.6 *The families $\mathcal{L}(REG)$, $\mathcal{L}(LIN)$, $\mathcal{L}(CF)$, $\mathcal{L}(CS)$, and $\mathcal{L}(RE)$ are closed under intersection with regular languages.*

Proof. The statement holds trivially for $\mathcal{L}(REG)$, $\mathcal{L}(CS)$, and $\mathcal{L}(RE)$, because any of these language families is closed by intersection (see Lemma 4.3) and contains all regular languages (see Theorem 2.37).

In order to prove the statement for $\mathcal{L}(CF)$ we construct a pushdown automaton $\mathcal{M}$ which accepts $L \cap R$ for a given context-free language L and a given regular language R. Let

$$\mathcal{M}_1 = (X, Z_1, \Gamma, z_{0,1}, F_1, \delta_1) \text{ and } \mathcal{A}_2 = (X, Z_2.z_{0,2}, F_2, \delta_2)$$

be a pushdown automaton and a finite automaton, respectively, such that $T(\mathcal{M}_1) = L$ and $T(\mathcal{A}_2) = R$. We construct the pushdown automaton

$$\mathcal{M} = (X, Z_1 \times Z_2, \Gamma, (z_{0,1}, z_{0,2}), F_1 \times F_2, \delta)$$

where

$$\begin{aligned} ((z_1', z_2'), R, \beta) \in \delta((z_1, z_2), a, \gamma) \quad &\text{if} \quad (z_1', \beta) \in \delta_1(z_1, a, \gamma) \text{ and } \delta_2(z_2, a) = z_2', \\ ((z_1', z_2), N, \beta) \in \delta((z_1, z_2), a, \gamma) \quad &\text{if} \quad (z_1', \beta) \in \delta_1(z_1, a, \gamma). \end{aligned}$$

By definition $\mathcal{M}$ behaves on the first component of the state and the pushdown tape as $\mathcal{M}_1$ and on the second component of the state as $\mathcal{A}_2$ (where a letter is only read by $\mathcal{A}_2$, if $\mathcal{M}_1$ moves to the right). Hence $\mathcal{M}$ accepts a word w if and only if w is accepted by $\mathcal{M}_1$ as well as by $\mathcal{A}_2$. Thus $T(\mathcal{M}) = L \cap R$.

For the family of linear languages, we only notice that the construction of $\mathcal{M}$ from $\mathcal{M}_1$ gives a 1-turn pushdown automaton if $\mathcal{M}_1$ is a 1-turn pushdown automaton. □

We now study the algebraically motivated operations concatenation and Kleene closure and those operations related to homomorphisms.

Lemma 4.7 *The families $\mathcal{L}(REG)$, $\mathcal{L}(CF)$, $\mathcal{L}(CS)$, and $\mathcal{L}(RE)$ are closed under concatenation. $\mathcal{L}(LIN)$ is not closed under concatenation.*

Proof. $\mathcal{L}(CF)$. Again, we start with two context-free grammars

$$G_1 = (N_1, T, P_1, S_1) \quad \text{and} \quad G_2 = (N_2, T, P_2, S_2)$$

with $N_1 \cap N_2 = \emptyset$ and show that the grammar

$$G = (N_1 \cup N_2 \cup \{S\}, T, P_1 \cup P_2 \cup \{S \to S_1S_2\}, S)$$

generates $L(G_1)$
$cdot L(G_2)$. It is sufficient to mention that – up to the order of the applications of rules – any derivation in G has the form

$$S \Longrightarrow S_1S_2 \overset{*}{\Longrightarrow} w_1S_2 \overset{*}{\Longrightarrow} w_1w_2$$

where, for $i \in \{1, 2\}$, $S_i \overset{*}{\Longrightarrow} w_i$ is a derivation in G_i (i. e., the derivation only uses rules of P_i). Since G is a context-free grammar, $L(G_1)$
$cdot L(G_2)$ is a context-free language.

$\mathcal{L}(CS)$ and $\mathcal{L}(RE)$. We repeat the proof for $\mathcal{L}(CF)$ where we suppose without loss of generality that the grammars are in the Kuroda normal form (see Theorem 2.19. This ensures that the derivations in G_1 and G_2 cannot be influenced by the contexts of the other part. Furthermore, we have to take care of the empty word in case of $\mathcal{L}(CS)$, which requires to represent the concatenation as a union by languages without the empty word and the language only consisting of the empty word; e. g., if $\lambda \in L(G_1)$ and $\lambda \in L(G_2)$, then

$$L(G_1) \cdot L(G_2) = ((L(G_1) \setminus \{\lambda\}) \cdot (L(G_2) \setminus \{\lambda\})) \cup (L(G_1) \setminus \{\lambda\}) \cup (L(G_2) \setminus \{\lambda\}) \cup \{\lambda\}.$$

The details are left to the reader.

$\mathcal{L}(REG)$. The above proof (for $\mathcal{L}(CF)$) does not work for regular languages since the newly introduced rule $S \to S_1S_2$ has not the required form.

Let $G_1 = (N_1, T_1, P_1, S_1)$ and $G_2 = (N_2, T_2, P_2, S_2)$ be regular grammars such that $L(G_1) = L_1$, $L(G_2) = L_2$ and $N_1 \cap N_2 = \emptyset$. Then we construct the grammar

$$G = (N_1 \cup N_2, T, P_1' \cup P_2, S_1)$$

where

$$P_1' = \{A \to wB : A \to wB \in P_1, B \in N_1\} \cup \{A \to wS_2 : A \to w \in P_1, w \in T^*\}.$$

According to this construction, all derivations in G have the form

$$S_1 \stackrel{*}{\Longrightarrow} w'A \Longrightarrow w'wS_2 \stackrel{*}{\Longrightarrow} w'ww_2$$

where

$$S_1 \stackrel{*}{\Longrightarrow} w'A \Longrightarrow w'w = w_1 \quad \text{and} \quad S_2 \stackrel{*}{\Longrightarrow} w_2$$

are derivations in G_1 and G_2, respectively. Hence

$$L(G) = \{w_1w_2 : w_1 \in L(G_1), w_2 \in L(G_2)\} = L(G_1) \cdot L(G_2).$$

$\mathcal{L}(LIN)$ The method used for $\mathcal{L}(REG)$ does not work since the derivation of the first grammar can end somewhere in the middle of the word and not at the end as in the case of regular grammars.

By Example 2.5, $L = \{a^nb^n \mid n \geq 1\}$ is a linear language. However, the language $L \cdot L = \{a^nb^na^mb^m \mid n \geq 1,\ m \geq 1\}$ is not linear as we have shown in the proof of Theorem 2.34. □

Lemma 4.8 *The families $\mathcal{L}(REG)$, $\mathcal{L}(CF)$, $\mathcal{L}(CS)$, and $\mathcal{L}(RE)$ are closed under (positive) Kleene closure. $\mathcal{L}(LIN)$ is not closed under (positive) Kleene closure.*

Proof. We first prove the statement for positive Kleene closure.

$\mathcal{L}(CF)$. Let L be a context-free language. Let $G = (N, T, P, S)$ be a context-free grammar which generates L. We set

$$G' = (N \cup \{S'\}, T, P \cup \{S' \to SS', S' \to S\}, S')$$

(where S' is an additional symbol, again). Up to the order of the application of the rules, any derivation in G' has the form

$$\begin{aligned} S' &\Longrightarrow SS' \stackrel{*}{\Longrightarrow} w_1S' \Longrightarrow w_1SS' \stackrel{*}{\Longrightarrow} w_1w_2S' \Longrightarrow w_1w_2SS' \Longrightarrow \dots \\ &\Longrightarrow w_1w_2\dots w_{n-1}S' \Longrightarrow w_1w_2\dots w_{n-1}S \stackrel{*}{\Longrightarrow} w_1w_2\dots w_{n-1}w_n, \end{aligned}$$

where, for $1 \leq i \leq n$, each derivation $S \overset{*}{\Longrightarrow} w_i$ uses only rules of P. Thus we have $w_i \in L(G) = L$ for $1 \leq i \leq n$. Hence $w_1 w_2 \dots w_n \in L^n$. It is obvious that any word $w \in L^n$ and only words of L^m with $m \geq 1$ can be generated. Therefore

$$L(G') = \bigcup_{n \geq 1} L^n = L^+,$$

which proves the context-freeness of L^+.

$\mathcal{L}(CS)$ and $\mathcal{L}(RE)$. Let L be a language of $\mathcal{L}(X)$, $X \in \{CS, RE\}$. Then L can be generated by a grammar $G = (N, T, P, S)$ in Kuroda normal form (see Theorem 2.19). We set

$$G' = (N \cup \{S', S''\}, T, P \cup P', S')$$

where P' consists of the rules

$$\begin{aligned} &S' \to S,\ S' \to SS'', \\ &xS'' \to xSS'',\ xS'' \to xS \quad \text{for } x \in T. \end{aligned}$$

By these it is ensured that the subderivations starting from S can not influence each other by context (since a new derivation can only be started if the preceding one has already produced the last terminal letter). Now we get $L(G') = L^+$ as above. The details of the proof are left to the reader.

$\mathcal{L}(REG)$. Let $G = (N, T, V, P, S)$ be a regular grammar with $L(G) = L$. We construct the regular grammar $G' = (N, T, P', S)$ where P' is obtained by adding all rules of the forms

$$A \to wS \text{ for } A \to w \in P, w \in T^*$$

to P. Then the derivations of G' have the form

$$\begin{aligned} S &\overset{*}{\Longrightarrow} w_1' A_1 \Longrightarrow w_1' w_1'' S \overset{*}{\Longrightarrow} w_1' w_2'' w_2' A_2 \Longrightarrow w_1' w_1'' w_2' w_2'' S \\ &\overset{*}{\Longrightarrow} w_1' w_1'' \dots w_{n-1}' w_{n-1}'' S \overset{*}{\Longrightarrow} w_1' w_1'' \dots w_{n-1}' w_{n-1}'' w_n, \end{aligned}$$

where $w_i' w_i'' \in L(G)$ for $1 \leq i \leq n-1$ and $w_n \in L(G)$. Now $L(G') = L^+$ can easily be proved.

Kleene closure. If $\lambda \in L$, then $L^* = L^+$ and we can use the above constructions. If $\lambda \notin L$, then $L^* = L^+ \cup \{\lambda\}$; because a grammar with the only rule $S \to \lambda$, generates the language which only consists of the empty word, the assertion follows by the above result for L^* and Lemma 4.2.

$\mathcal{L}(LIN)$. We consider the linear language $L(G_2) = \{a^n b^n \mid n \geq 1\}$ from Example 2.5. It is easy to see that

$$L(G_2)^+ = \{a^{n_1} b^{n_1} a^{n_2} b^{n_2} \dots a^{n_t} b^{n_t} \mid t \geq 1,\ n_i \geq 1,\ 1 \leq i \leq t\}.$$

Let us assume that $L(G_2)^+$ is linear. Because $R = \{a^p b^q a^r b^s \mid p, q, r, s \geq 1\}$ is regular (the verification is left to the reader), then

$$L(G_2)^+ \cap R = \{a^n b^n a^m b^m \mid n, m \geq 1\}$$

is also linear by Lemma 4.6. However, as an application of the pumping lemma for linear languages we have shown that $L(G_2)^+ \cap R$ is not linear. This contradiction shows that our above assumption is wrong, i. e., $L(G_2)^+$ is not a linear languages. Thus we have shown the non-closure of the family of linear languages under positive Kleene closure. The analogous statement for the Kleene closure follows as above taking into consideration that $L(G_2)^* \cap R = L(G_2)^+ \cap R$. □

Lemma 4.9 *The families $\mathcal{L}(REG)$, $\mathcal{L}(LIN)$, $\mathcal{L}(CF)$, and $\mathcal{L}(RE)$ are closed under homomorphisms.*

Proof. Let h be homomorphism which maps T^* to Y^*.

$\mathcal{L}(CF)$. Let L be a context-free language. Then there is a context-free grammar $G = (N, T, P, S)$ in Chomsky normal form such that $L(G) = L$ (see Theorems 2.26). Therefore all rules are of the form $A \to BC$ or $A \to a$ with $A, B, C \in N$ and $a \in T$. Moreover, we can arrange the order of the applications of rules such that any derivation has the form

$$S \stackrel{*}{\Longrightarrow} A_1A_2 \ldots A_k \Longrightarrow a_1A_2A_3 \ldots A_k \Longrightarrow a_1a_2A_3A_4 \ldots A_k \Longrightarrow \ldots \Longrightarrow a_1a_2 \ldots a_k$$

(where we apply only rules of the form $A \to BC$ in the subderivation $S \stackrel{*}{\Longrightarrow} A_1A_2 \ldots A_k$. We now construct the grammar $G' = (N, Y, P', S)$ where P' is obtained from P by a replacement of any rule of the form $A \to a \in P$ by $A \to h(a)$. Then it follows that – without loss of generality – the derivations in G' have the form

$$\begin{aligned} S &\stackrel{*}{\Longrightarrow} A_1A_2 \ldots A_k \Longrightarrow h(a_1)A_2A_3 \ldots A_k \Longrightarrow h(a_1)h(a_2)A_3A_4 \ldots A_k \Longrightarrow \ldots \\ &\Longrightarrow h(a_1)h(a_2) \ldots h(a_k) = h(a_1a_2 \ldots a_k). \end{aligned}$$

Thus we have $w \in L(G)$ if and only if $h(w) \in L(G')$ and therefore $L(G') = h(L(G)) = h(L)$. Furthermore, G' is a context-free grammar. Hence $\mathcal{L}(CF)$ is closed under homomorphisms.

$\mathcal{L}(RE)$. We repeat the proof for $\mathcal{L}(CF)$ but use the Kuroda normal form instead of the Chomsky normal form.

$\mathcal{L}(LIN)$. Let L be a linear grammar. Then there is a linear grammar $G = (N, T, P, S)$ generating L. Moreover, any derivation in G has the form

$$\begin{aligned} S &\to w_1A_1v_1 \Longrightarrow w_1w_2A_2v_2v_1 \Longrightarrow \ldots \Longrightarrow w_1w_2 \ldots w_kA_kv_kv_{k-1} \ldots v_1 \\ &\Longrightarrow w_1w_2 \ldots w_kuv_kv_{k-1} \ldots v_1 \end{aligned}$$

where the rules $S \to w_1A_1v_1$, $A_i \to w_{i+1}A_{i+1}v_{i+1}$ for $1 \le i \le k-1$, and $A_k \to u$ are applied.

We now define the grammar $G = (N, Y, P', S)$ by

$$P' = \{A \to h(w)Bh(v) \mid A \to wBv \in P\} \cup \{A \to h(w) \mid A \to w \in P\}.$$

Any derivation in G' has the form

$$\begin{aligned} S &\to h(w_1)A_1h(v_1) \Longrightarrow h(w_1)h(w_2)A_2h(v_2)h(v_1) \\ &\Longrightarrow \ldots \Longrightarrow h(w_1)h(w_2) \ldots h(w_k)A_kh(v_k)h(v_{k-1}) \ldots h(v_1) \\ &\Longrightarrow h(w_1)h(w_2) \ldots h(w_k)h(u)h(v_k)h(v_{k-1}) \ldots h(v_1) \\ &= h(w_1w_2 \ldots w_kuv_kv_{k-1} \ldots v_1). \end{aligned}$$

Again, we have $z \in L(G)$ if and only if $h(z) \in L(G')$ and therefore $L(G') = h(L(G)) = h(L)$. The assertion follows because G' is linear.

$\mathcal{L}(REG)$. The construction given in the proof for $\mathcal{L}(LIN)$ gives a regular grammar G', if G is regular. □

We have not given the closure property of $\mathcal{L}(CS)$ under homomorphisms. This will be added in Chapter 5.

Lemma 4.10 *The families $\mathcal{L}(REG)$, $\mathcal{L}(LIN)$, $\mathcal{L}(CF)$, $\mathcal{L}(CS)$, and $\mathcal{L}(RE)$ are closed under inverse homomorphisms.*

Proof. $\mathcal{L}(REG)$. Let L be a regular language. Then there is a deterministic finite automata $\mathcal{A} = (X, Z, z_0, F, \delta)$ such that $T(\mathcal{A}) = L$. Now let $h : Y^* \to X^*$ be a homomorphism. Then $a_1a_2 \dots a_n \in h^{-1}(L)$, $a_i \in Y$ for $1 \le i \le n$ if and only if $h(a_1a_2 \dots a_n) = h(a_1)h(a_2) \dots h(a_n) \in L$. We construct the automaton $\mathcal{A}' = (Y, Z, z_0, F, \delta')$ by setting

$$\delta'(z, a) = \delta^*(z, h(a)) \text{ for } a \in Y.$$

By definition of δ', we immediately have

$$\delta'(z_0, a_1a_2 \dots a_n) = \delta(z_0, h(a_1)h(a_2) \dots h(a_n) \in F.$$

Therefore $a_1a_2 \dots a_n \in T(\mathcal{A}')$ if and only if $h(a_1)h(a_2) \dots h(a_n) \in T(\mathcal{A}')$. This implies that $\mathcal{A}'$ accepts $h^{-1}(T(\mathcal{A})) = h^{-1}(L)$. Hence $h^{-1}(L)$ is regular.

$\mathcal{L}(CF)$. Let L be a context-free language and $\mathcal{M} = (X, Z, \Gamma, z_0, F, \delta)$ be a pushdown automaton. Moreover, let $h : Y^* \to X^*$ be a homomorphism. For any letter $a \in Y$ with $h(a) = b_1b_2 \dots b_{r_a}$, we introduce new symbols (a, i), $1 \le i \le r_a + 1$. Let Z' be the set of all new symbols. Then we consider the pushdown automaton

$$\mathcal{M}' = (Y, \{(z, z) \mid z \in Z\} \cup (Z \times Z'), z_0, \{(z, z) \mid z \in F\}, \delta'),$$

where δ' is defined as follows:

$$\begin{array}{rcl}
\delta'((z,z), a, \#) &=& \{(z, (a,1)), \lambda)\} \quad \text{for} \quad z \in Z,\ a \in Y, \\
\delta'((z,z), a, \gamma) &=& \{(z, (a,1)), \gamma)\} \quad \text{for} \quad z \in Z,\ a \in Y, \gamma \in \Gamma, \\
\delta'((z, (a,i)), \lambda, \gamma) &=& \{(z', (a, i+1)), \beta) \mid (z', \beta) \in \delta(z, b_i, \gamma)\} \\
&& \qquad\qquad \text{for} \quad z \in Z, a \in Y, 1 \le i \le r_a, \gamma \in \Gamma \cup \{\#\}, \\
\delta'((z, (a,i)), \lambda, \gamma) &=& \{(z', (a, i)), \beta) \mid (z', \beta) \in \delta(z, \lambda, \gamma)\} \\
&& \qquad\qquad \text{for} \quad z \in Z, a \in Y, 1 \le i \le r_a, \gamma \in \Gamma \cup \{\#\}, \\
\delta'((z, (a, r_{a+1})), \lambda, \gamma) &=& \{((z,z), \gamma)\} \quad \text{for} \quad z \in Z, a \in Y, \gamma \in \Gamma, \\
\delta'((z, (a, r_{a+1})), \lambda, \#) &=& \{(z,z), \lambda)\} \quad \text{for} \quad z \in Z, a \in Y,
\end{array}$$

After reading a letter a in state (z, z), we change to $(z, (a, 1))$ and simulate the work of $\mathcal{M}$ on $h(a) = b_1b_2 \dots b_{r_a}$ by changing the first component according to $\mathcal{M}$ and moving to $(a, i+1)$ if b_i is "read". The (z', a_{r_a+1}) says that the work on $h(a)$ is simulated and we enter (z', z'). Therefore the pushdown automaton $\mathcal{M}'$ accepts $a_1a_2 \dots a_n$ if and only if we obtain (q,q) for some $q \in F$ on the input $a_1a_2 \ldots a_n$ if and only the simulation on $h(a_1)h(a_2)\ldots h(a_n)$ leads to $q \in F$. Thus $a_1a_2\ldots a_n \in T(\mathcal{M}')$ if and only if $h(a_1)h(a_2)\ldots h(a_n) \in T)\mathcal{M} = L$ if and only if $a_1a_2\ldots a_n \in h^{-1}(L)$.

We omit the proofs for $\mathcal{L}(LIN)$, $\mathcal{L}(CS)$, and $\mathcal{L}(RE)$ which can be given analogously, i. e., the automaton for $h^{-1}(L)$ simulates the work of the automaton for L. □

The proof of the following theorem is left to the reader (see Exercise ???).

Lemma 4.11 *The families $\mathcal{L}(REG)$, $\mathcal{L}(LIN)$, $\mathcal{L}(CF)$, $\mathcal{L}(CS)$, and $\mathcal{L}(RE)$ are closed under reversal.* □

We summarize the closure properties of the families of the Chomsky hierarchy in the table given in Figure 4.1 where a + or − in the meet of the column associated with a family $\mathcal{L}$ and the row associated with an operation τ means that $\mathcal{L}$ is closed or not closed under τ, respectively.

	$\mathcal{L}(RE)$	$\mathcal{L}(CS)$	$\mathcal{L}(CF)$	$\mathcal{L}(LIN)$	$\mathcal{L}(REG)$
union	+	+	+	+	+
intersection	+	+	−	−	+
intersection with regular sets	+	+	+	+	+
complement	−	+	−	−	+
product	+	+	+	−	+
(positive) Kleene closure	+	+	+	−	+
homomorphisms	+	−	+	+	+
non-erasing homomorphisms	+	+	+	+	+
inverse homomorphisms	+	+	+	+	+
reversal	+	+	+	+	+

Figure 4.1: Table of closure properties

We now show that a family of languages which is closed under certain operations is also closed under some further operations. In order to shorten the statements we give the following notation.

Definition 4.12 *A family $\mathcal{L}$ of languages is called an* abstract family of languages *(abbreviated by AFL) if*

– *it contains at least one non-empty language,*
– *it is closed under union, product, positive Kleene closure, non-erasing homomorphisms, inverse homomorphisms and intersections with regular languages.*

The family $\mathcal{L}$ is called a full AFL *if, in addition, it is closed under (arbitrary) homomorphisms.*

By Figure 4.1, $\mathcal{L}(REG)$, $\mathcal{L}(CF)$, $\mathcal{L}(CS)$, and $\mathcal{L}(RE)$ are AFLs; $\mathcal{L}(REG)$, $\mathcal{L}(CF)$, and $\mathcal{L}(RE)$ are full AFLs; $\mathcal{L}(LIN)$ is not an abstract family of languages.

Lemma 4.13 *Any full AFL is closed under Kleene closure.*

Proof. Since $L^* = L^+ \cup \{\lambda\}$ and any full AFL is closed under positive Kleene closure and union, it is sufficient to show that any full AFL contains $\{\lambda\}$.

Let $\mathcal{L}$ be an AFL. We first show that $\{\lambda\} \in \mathcal{L}$. By defition, $\mathcal{L}$ contains a non-empty language K. If $K = \{\lambda\}$, then the assertion holds. If $K \neq \{\lambda\}$, then K contains a non-empty word z. We define the homomorphism $h : (alph(K))^* \to (alph(K))^*$ by $h(a) = \lambda$ for all $a \in alph(K)$. Then

$$\{\lambda\} = h(K \cap \{w\}).$$

Because $\mathcal{L}$ is closed under intersections with regular sets and homomorphisms, we obtain $\{\lambda\} \in \mathcal{L}$. □

Theorem 4.14 *Any AFL is closed under set-theoretic subtraction of regular languages.*

Proof. Let $\mathcal{L}$ be an AFL. For a language $L \subseteq X^*$ from $\mathcal{L}$ and a regular set $R \subseteq X^*$, $L \setminus R = L \cap (X^* \setminus R)$. Since the complement of a regular set is regular, too (see Theorem 4.4), $L \setminus R$ is an intersection of a languages in $\mathcal{L}$ with a regular set. Thus $L \setminus R \in \mathcal{L}$ by the closure properties required for an AFL. □

Theorem 4.15 *Any full AFL is closed under left and right quotients by regular sets, i. e., for any language L of the AFL $\mathcal{L}$ and any regular set R, the quotients $D_l(L,R)$ and $D_r(L,R)$ belong to $\mathcal{L}$.*

Proof. We only give the proof for the left quotient; the proof for the right quotient is analogous.

Let $\mathcal{L}$ be an AFL, L a language in $\mathcal{L}$, and R a regular set. Furthermore, let

$$X = alph(L) \cup alph(R) \text{ and } X' = \{a' \mid a \in X\}.$$

We define the homomorphisms

$$h : X^* \to X^*,\ h_1 : (X \cup X')^* \to X^* \text{ and } h_2 : (X \cup X')^* \to X^*$$

by

$$h(a) = a', h_1(a') = a, h_1(a) = a, h_2(a') = \lambda, h_2(a) = a \text{ for } a \in X.$$

Additionally, we consider the set

$$Q = h(R)(alph(L))^*.$$

By the closure of $\mathcal{L}(REG)$ under homomorphisms and concatenation (see Theorems 4.7 and 4.9), Q is regular. Because

$$\begin{aligned} h_2(h_1^{-1}(L) \cap Q) &= h_2(\{w'v \mid w' \in h(R), v \in (alph(L))^*, wv \in L\}) \\ &= h_2(\{w'v \mid w \in R, v \in (alph(L))^*, wv \in L\}) \\ &= \{h_2(w')h_2(v) \mid w \in R, v \in (alph(L))^*, wv \in L\} \\ &= \{v \mid wv \in L \text{ for some } w \in R\}, \end{aligned}$$

we have

$$D_l(L,R) = h_2(h_1^{-1}(L) \cap Q).$$

By the closure properties of an AFL, we obtain $D_l(L,R) \in \mathcal{L}$. □

Theorem 4.16 *Any full AFL is closed under substitutions by regular sets.*

Proof. Let $\mathcal{L}$ be an AFL, $L \subseteq X^*$ a language of L and $\tau : X^* \to Y^*$ a substitution such that $\tau(a)$ is a regular set for any $a \in X$. Let $X = \{a_1, a_2, \ldots, a_n\}$ and $\tau(a_i) = R_i \in \mathcal{L}(REG)$ for $1 \le i \le n$. We define

$$X' = \{a' \mid a \in X\},$$
$$h_1 : (X' \cup Y)^* \to X^* \text{ by } h_1(x') = x \text{ for } x \in X \text{ and } h_1(y) = \lambda \text{ for } y \in Y,$$
$$h_2 : (X' \cup Y)^* \to Y^* \text{ by } h_2(x') = \lambda \text{ for } x \in X \text{ and } h_2(y) = y \text{ for } y \in Y,$$
$$R = \bigcup_{i=1}^{n} a_i' R_i.$$

Then we get

$$\begin{aligned}
h_1^{-1}(L) &= \{u_0x_1'u_1x_2'u_2 \ldots x_r'u_r \mid x_1x_2\ldots x_r \in L,\ u_i \in Y^* \text{ for } 1 \le i \le r\},\\
h_1^{-1}(L) \cap R &= \{x_1'u_1x_2'u_2 \ldots x_r'u_r \mid x_1x_2\ldots x_r \in L,\ u_i \in \tau(x_i) \text{ for } 1 \le i \le r\},\\
h_2(h_1^{-1}(L) \cap R) &= \{u_1u_2 \ldots u_r \mid x_1x_2\ldots x_r \in L,\ u_i \in \tau(x_i) \text{ for } 1 \le i \le r\},
\end{aligned}$$

and finally,

$$\tau(L) = h_2(h_1^{-1}(L) \cap R).$$

By the closure properties required for a full AFL, we obtain $\tau(L) \in \mathcal{L}$. □

4.2 Algebraic Characterizations of Language Families

4.2.1 Characterizations of Language Families by Operations

The aim of this section is to present some characterizations of language families by algebraic means. We start with characterizations by closure properties under certain operations and containments of very special languages.

Definition 4.17 *Regular expressions over an alphabet X are inductively defined as follows:*

1. *$\emptyset$, λ and x with $x \in X$ are regular expressions.*
2. *If R_1, R_2 and R are regular expressions, then $(R_1 + R_2)$, $(R_1 \cdot R_2)$ and R^* are also regular expressions.*

With any regular expression we associate a regular language.

Definition 4.18 *For a regular expression U over the alphabet X, the associated set $M(U)$ is inductively defined by the following settings:*

1. *$M(\emptyset) = \emptyset$, $M(\lambda) = \{\lambda\}$ uand $M(x) = \{x\}$ for $x \in X$,*

2. *If R_1, R_2 and R are regular expressions, then*

$$\begin{aligned} M((R_1+R_2)) &= M(R_1)\cup M(R_2), \\ M((R_1\cdot R_2)) &= M(R_1)\cdot M(R_2), \\ M(R^*) &= (M(R))^* . \end{aligned}$$

Example 4.19 Let $X=\{a,b,c\}$. By condition 1. of Definition 4.17,

$$R_0=\lambda,\ R_1=a,\ R_2=b,\ R_3=c$$

are regular expressions over X. By condition 2. of Definition 4.17, the following constructs are also regular expressions:

$$\begin{aligned} R_1' &= (R_1\cdot R_1)=(a\cdot a)\,, \\ R_1'' &= (R_1'\cdot R_1)=((a\cdot a)\cdot a)\,, \\ R_2' &= R_2^*=b^*\,, \\ R_2'' &= (R_2'+R_1'')=(b^*+((a\cdot a)\cdot a)))\,, \\ R_3' &= R_3^*=c^*\,, \\ R_3'' &= (R_3\cdot R_3')=(c\cdot c^*)\,, \\ R_4 &= (R_2''\cdot R_3'')=((b^*+((a\cdot a)\cdot a)))\cdot(c\cdot c^*)), \\ R_5 &= (R_0+R_4)=(\lambda+((b^*+((a\cdot a)\cdot a)))\cdot(c\cdot c^*)))\,. \end{aligned}$$

According to Definition 4.18 we obtain the following associated sets (where obvious simplifications are done):

$$\begin{aligned} M(R_0) &= \{\lambda\},\ M(R_1)=\{a\},\ M(R_2)=\{b\},\ M(R_3)=\{c\}, \\ M(R_1') &= =M((R_1\cdot R_1))=\{a\}\cdot\{a\}=\{a^2\}\,, \\ M(R_1'') &= M((R_1'\cdot R_1))=\{a^2\}\cdot\{a\}=\{a^3\}\,, \\ M(R_2') &= M(R_2^*)=\{b\}^*=\{b^m : m\geq 0\}\,, \\ M(R_2'') &= M((R_2'+R_1''))=\{b^m : m\geq 0\}\cup\{a^3\}\,, \\ M(R_3') &= M(R_3^*)=\{c\}^*=\{c^n : n\geq 0\}\,, \\ M(R_3'') &= M((R_3\cdot R_3'))=\{c\}\{c^n : n\geq 0\}=\{c^n : c\geq 1\}\,, \\ M(R_4) &= M((R_2''\cdot R_3''))=(\{b^m : m\geq 0\}\cup\{a^3\})\cdot\{c^n : n\geq 1\} \\ &= \{b^mc^n : m\geq 0, n\geq 1\}\cup\{a^3c^n : n\geq 3\}\,, \\ M(R_5) &= M((R_0+R_4))=\{\lambda\}\cup(\{b^mc^n : m\geq 0, n\geq 1\}\cup\{a^3c^n : n\geq 3\}) \\ &= \{\lambda\}\cup\{b^mc^n : m\geq 0, n\geq 1\}\cup\{a^3c^n : n\geq 3\}. \end{aligned}$$

If $U=((\ldots((R_1+R_2)+R_3)+\ldots)+R_n)$, then to shorten the notation we write

$$U=\sum_{i=1}^{n} R_i\,.$$

Obviously,

$$M(U)=\bigcup_{i=1}^{n} M(R_i)\,.$$

In an analogous way we use sums and unions over certain sets of indexes.

Theorem 4.20 *A language L is regular if and only if there is a regular expression R such that $M(R) = L$.*

Proof. $\Longleftarrow$) We show inductively that, for any regular expression U, the associated set $M(U)$ is regular.

If U is a regular expression by condition 1. of Definition 4.17, then all associated sets $M(\emptyset) = \emptyset$, $M(\lambda) = \{\lambda\}$ and $M(x) = \{x\}$ with $x \in X$ are finite and therefore regular (see Exercise ???).

Now let U be a regular expression, which is obtained from regular expressions R_1, R_2, and R according to condition 2. of Definition 4.17, and let $M(R_1)$, $M(R_2)$, and $M(R)$ be the sets associated with R_1, R_2, and R, respectively. By induction hypotheses, $M(R_1)$, $M(R_2)$, and $M(R)$ are regular. If $U = (R_1 + R_2)$, then $M(U) = M(R_1) \cup M(R_2)$. By Theorem 4.2, $M(U)$ is regular. If $U = (R_1 \cdot R_2)$ or $U = R^*$, then the associated sets $M(U) = M(R_1) \cdot M(R_2)$ or $M(U) = (M(R))^*$, respectively, so sind nach den are also regular by Theorems 4.7 and 4.8,respectively.

$\Longrightarrow$) Let L be a regular language. Then there is a finite deterministic automaton $\mathcal{A} = (X, Z, z_0, F, \delta)$ with $T(\mathcal{A}) = L$. Without loss of generality we can assume that

$$Z = \{0, 1, 2, \dots r\} \quad \text{and} \quad z_0 = 0$$

for some $r \geq 0$. For $i, j \in Z$ and $0 \leq k \leq r + 1$, by $L^k_{i,j}$ we denote the set of all words w satisfying the following two conditions: Eigenschaften:

(a) $\delta(i, w) = j$,

(b) for any $u \neq \lambda$ with $w = uu'$ and $|u| < |w|$, we have $\delta(i, u) < k$.

Obviously,

$$L = T(\mathcal{A}) = \bigcup_{j \in F} L^{r+1}_{0,j}. \tag{4.1}$$

We now prove that, for any set $L^k_{i,j}$, $i, j \in Z$, $0 \leq k \leq r + 1$, there is a regular expression $R^k_{i,j}$ with $M(R^k_{i,j}) = L^k_{i,j}$. The proof will be given by induction on k.

Let $k = 0$. For $i \neq j$, by definition, $L^0_{i,j}$ consists of all words w, which directly transform the state i into state j, because by condition (b) no intermediate states occur. Thus w is a word of length 1.Therefore

$$L^0_{i,j} = \{x : x \in X, \delta(i, x) = j\}.$$

This can be written as

$$L^0_{i,j} = \bigcup_{\substack{x \in X \\ \delta(i,x)=j}} \{x\}.$$

Thus we also have

$$L^0_{i,j} = M(\sum_{\substack{x \in X \\ \delta(i,x)=j}} x) = \bigcup_{\substack{x \in X \\ \delta(i,x)=j}} \{x\},$$

which proves our assertion. If $i = j$, in addition to the words of length 1 which transform i into i, the empty word is in $L^0_{i,i}$. Hence

$$L^0_{i,j} = M(\lambda + \sum_{\substack{x \in X \\ \delta(i,x)=i}} x) = \{\lambda\} \cup \bigcup_{\substack{x \in X \\ \delta(i,x)=j}} \{x\}$$

is regular.

Let $k \geq 1$ and let us assume (by induction hypotheses) that, for all sets $L_{i,j}^s$ with $s < k$, there is a regular expression $R_{i,j}^s$ such that $L_{i,j}^s = M(R_{i,j}^s)$. We first show that

$$L_{i,j}^k = L_{i,k-1}^{k-1}(L_{k-1,k-1}^{k-1})^* L_{k-1,j}^{k-1} \cup L_{i,j}^{k-1}\,. \tag{4.2}$$

Let $w = x_1x_2\dots x_n$ be a word of $L_{i,j}^k$. For $1 \leq p \leq n-1$, we set

$$z_p = \delta(i, x_1x_2\dots x_p).$$

If $z_p < k-1$ for $1 \leq p \leq n-1$, then w is in $L_{i,j}^{k-1}$, too. Thus we get $w \in L_{i,j}^{k-1}$.
If there exist integers $t \geq 1$ and $1 \leq p_1 \leq p_2 \leq \dots \leq p_t \leq n-1$ such that

$$z_{p_1} = z_{p_2} = \dots = z_{p_t} = k-1 \quad \text{and} \quad z_p < k-1 \ \text{ for } \ p \notin \{p_1, p_2, \dots, p_t\},$$

then we have

$$\begin{aligned}
&\delta(i, x_1x_2\dots x_{p_1}) = k-1,\\
&\delta(k-1, x_{p_q+1}x_{p_q+2}\dots x_{p_{q+1}}) = k-1 \quad \text{for} \quad 1 \leq q \leq t-1,\\
&\delta(k-1, x_{p_t}x_{p_t+1}\dots x_n) = j.
\end{aligned}$$

Furthermore, $k-1$ is not an intermediate state in each of these transformations. Therefore we obtain

$$\begin{aligned}
&x_1x_2\dots x_{p_1} \in L_{i,k-1}^{k-1},\\
&x_{p_q}x_{p_q+1}x_{p_q+2}\dots x_{p_{q+1}} \in L_{k-1,k-1}^{k-1} \quad \text{für} \quad 1 \leq q \leq t-1,\\
&x_{p_t}x_{p_t+1}x_{p_t+2}\dots x_n \in L_{k-1,j}^{k-1}.
\end{aligned}$$

and

$$w = x_1\dots x_{p_1}\dots x_{p_2}...x_{p_t}\dots x_n \in L_{i,k-1}^{k-1}(L_{k-1,k-1}^{k-1})^* L_{k-1,j}^{k-1}.$$

Consequently,

$$L_{i,j}^k \subseteq L_{i,k-1}^{k-1}(L_{k-1,k-1}^{k-1})^* L_{k-1,j}^{k-1} \cup L_{i,j}^{k-1}\,.$$

The converse inclusion and thus the equality in (4.2) follow by analogous arguments.

The equation (4.2) yields immediately

$$\begin{aligned}
L_{i,j}^k &= M(R_{i,k-1}^{k-1})M(R_{k-1,k-1}^{k-1})^* M(R_{k-1,j}^{k-1}) \cup M(L_{i,j}^{k-1})\\
&= M((((R_{i,k-1}^{k-1} \cdot [R_{k-1,k-1}^{k-1}]^*) \cdot R_{k-1,j}^{k-1}) + R_{i,j}^{k-1}))\,,
\end{aligned}$$

which proves that, any set $L_{i,j}^k$ can be described by a regular expression $R_{i,j}^k$.

If we take into consideration the relation

$$L = \bigcup_{j \in F} L_{0,j}^{r+1} = M(\sum_{j \in F} R_{0,j}^{r+1})$$

which follows from (4.1), then the second implication of our statement is shown. □

We present another formulation of Theorem 4.20 where we use immediately the operations instead of the regular expressions.

Theorem 4.21 *A language L over the alphabet X is regular if and only if it can be generated by an iterated application of union, product, and Kleene closure from the sets $\emptyset$, $\{\lambda\}$ and $\{x\}$ for $x \in X$.* □

Theorem 4.20 (or equivalently, Theorem 4.21) was first shown by the American mathematician Stephen Cole Kleene[1] in the paper [16], and therefore it is often called Kleene's Theorem. We want to mention that in the original paper essentially events in nerve nets are characterized by union, product, and Kleene closure, and a relation to automata is only mentioned. Thus the paper gives a very early relation between biology and formal languages. Further examples of such a relation are discussed in Chapters **??**, **??**, and **??**.

We conclude the considerations concerning Kleene's Theorem by an example.

Example 4.22 We consider the finite automaton $\mathcal{A}$ of Example 3.43 and construct for the language accepted by $\mathcal{A}$ the representation by union, product, and Kleene closure. To simplify the notation we write i instead of z_i. We obtain

$$\begin{aligned}
T(\mathcal{A}) &= L^4_{0,2}\\
&= L^3_{0,3}(L^3_{3,3})^*L^3_{3,2} \cup L^3_{0,2}\\
&= L^3_{0,2}(\text{wegen } L^3_{3,2} = \emptyset)\\
&= L^2_{0,2}(L^2_{2,2})^*L^2_{2,2} \cup L^2_{0,2}\\
&= L^2_{0,2}(L^2_{2,2})^*(\text{wegen } \lambda \in L^2_{0,2})\\
&= (L^1_{0,1}(L^1_{1,1})^*L^1_{1,2} \cup L^1_{0,2})(L^1_{2,1}(L^1_{1,1})^*L^1_{1,2} \cup L^1_{2,2})^*\\
&= L^1_{0,1}\{a\} \cdot (L^1_{2,1}\{a\})^*\text{wegen } L^1_{1,2} = \{a\}, L^1_{1,1} = L^1_{0,2} = L^1_{2,2} = \emptyset)\\
&= (L^0_{0,0}(L^0_{0,0})^*L^0_{0,1} \cup L^0_{0,1})\{a\} \cdot ((L^0_{2,0}(L^0_{0,0})^*L^0_{0,1} \cup L^0_{2,1})\{a\})^*\\
&= (\{\lambda, c\}\{\lambda, c\}^*\{a\} \cup \{a\})\{a\} \cdot ((\{c\}\{\lambda, c\}^*\{a\})\{a\})^*,
\end{aligned}$$

which finally yields the representation

$$T(\mathcal{A}) = ((((((\lambda + c) \cdot (\lambda + c)^*) \cdot a) + a) \cdot a) \cdot (((c \cdot (\lambda + c)^*) \cdot a) \cdot a^*)). \tag{4.3}$$

In Example 3.43 we have shown that

$$T(\mathcal{A}) = \{c^{n_1}aac^{n_2}aa \ldots c^{n_k}aa : k \geq 1, n_1 \geq 0, n_i \geq 1, 2 \leq i \leq k\}.$$

Because

$$\{x\}^* = \{x^n : n \geq 0\} \quad \text{and} \quad \{x\}^+ = \{x^n : n \geq 1\} = \{x\}\{x\}^*,$$

we have also the representation

$$T(\mathcal{A}) = \{c\}^*\{a\}\{a\}(\{c\}\{c\}^*\{a\}\{a\})^*. \tag{4.4}$$

Since the representations of $T(\mathcal{A})$ given in (4.3) and (4.4) are different, this example shows that the representation and therefore the regular expression for a regular set are not uniquely determined.

[1]born in 1909, died in 1994)

We now construct a regular grammar which generates $T(\mathcal{A})$. We start with the representation given in (4.4). Obviously, for all grammar given in this construction the terminal alphabet T is the input alphabet $\{a, b, c\}$ of $\mathcal{A}$.

We first construct grammars, which generate the necessary seven sets consisting of a single word. Moreover, we use the notation in such a way that the alphabets of nonterminals are disjunct since this was supposed in the constructions of grammars generating the union and product. Thus we start with

$$\begin{aligned} G_i &= (\{S_i\}, T, \{S_i \to c\}, S_i) \quad \text{für } i \in \{1, 4, 5\} \\ G_j &= (\{S_j\}, T, \{S_j \to a\}, S_j) \quad \text{für } i \in \{2, 3, 6, 7\} \end{aligned}$$

which generate

$$L(G_i) = \{c\} \text{ for } i \in \{1, 4, 5\} \quad \text{and} \quad L(G_j) = \{a\} \text{ for } i \in \{2, 3, 6, 7\}.$$

Therefore

$$T(\mathcal{A}) = L(G_1)^* L(G_2) L(G_3) (L(G_4) L(G_5)^* L(G_6) L(G_7))^*.$$

We now follow the constructions given in the proofs of the Lemmas 4.7 and 4.8. The following table gives the generated language, the rules and the axiom (the nonterminals can be seen from the rules and the terminal set is $\{a, b, c\}$):

$L(G_1)^* = \{a\}^*$	$S_1' \to \lambda, S_1' \to S_1, S_1 \to cS_1, S_1 \to c$	S_1'
$L(G_1)^* L(G_2)$	$S_1' \to S_2, S_1' \to S_1, S_1 \to cS_1, S_1 \to cS_2,$ $S_2 \to a$	S_1'
$L(G_1)^* L(G_2) L(G_3)$	$S_1' \to S_2, S_1' \to S_1, S_1 \to cS_1, S_1 \to cS_2,$ $S_2 \to cS_3, S_3 \to c$	S_1'
$L(G_5)^*$	$S_5' \to \lambda, S_5' \to S_5, S_5 \to cS_5, S_5 \to c$	S_5'
$L(G_4) L(G_5)^*$	$S_4 \to cS_5', S_5' \to \lambda, S_5' \to S_5, S_5 \to cS_5,$ $S_5 \to c$	S_4
$L(G_4) L(G_5)^* L(G_6) L(G_7)$	$S_4 \to cS_5', S_5' \to S_6, S_5' \to S_5, S_5 \to cS_5,$ $S_5 \to cS_6, S_6 \to aS_7, S_7 \to a$	S_4
$(L(G_4) L(G_5)^* L(G_6) L(G_7))^*$	$S_4' \to \lambda, S_4' \to S_4, S_4 \to cS_5', S_5' \to S_6,$ $S_5' \to S_5, S_5 \to cS_5, S_5 \to cS_6, S_6 \to aS_7,$ $S_7 \to a$	S_4'
$T(\mathcal{A})$	$S_1' \to S_2, S_1' \to S_1, S_1 \to cS_1, S_1 \to cS_2,$ $S_2 \to cS_3, S_3 \to cS_4', S_4' \to \lambda, S_4' \to S_4,$ $S_4 \to cS_5', S_5' \to S_6, S_5' \to S_5, S_5 \to cS_5,$ $S_5 \to cS_6, S_6 \to aS_7, S_7 \to a$	S_1'

We now present a further characterization of the family of regular languages by operations, more precisely we show that $\mathcal{L}(REG)$ is the only minimal abstract family of languages (with respect to inclusion).

Theorem 4.23 *For any AFL $\mathcal{L}$, we have $\mathcal{L}(REG) \subseteq \mathcal{L}$.*

Proof. Let $\mathcal{L}$ be a full AFL, and let $R \subseteq X^*$ be an arbitrary regular set.

By the first condition of Definition 4.12, $\mathcal{L}$ contains a non-empty language L. Let $Y = alph(L)$ and w be a word from L. Because the finite language $\{w\}$ is regular, we get $L \cap \{w\} = \{w\} \in \mathcal{L}$. For each $a \in X$, we define the homomorphism $h_a : X^* \to Y^*$ by $h_a(a) = w$ and $h_a(b) = aw$ for $b \in Y$, $b \neq a$. Then $h_a^{-1}(\{w\} = \{a\}$. By the closure properties required for an AFL, $\{a\} \in \mathcal{L}$ for any $a \in X$. Moreover, using the homomorphism $h : X^* \to X^*$ with $h(a) = \lambda$ for any $a \in X$, we get $h(\{a\}) = \{\lambda\} \in \mathcal{L}$. Furthermore, for $a, b \in X$ with $a \neq b$, $\{a\} \cap \{b\} = \emptyset$. Therefore the empty set belongs to $\mathcal{L}$ since $\{a\} \in \mathcal{L}$ and $\{b\} \in \mathcal{L}(REG)$. Thus all sets associated with the basic regular expression over X belong to $\mathcal{L}$. Hence, by Theorem 4.21, R can be obtained by applications of union, product and Kleene closure. Since any AFL is closed under union, product and Kleene closure, we get $R \in cL$. □

Corollary 4.24 *The family $\mathcal{L}(REG)$ is the smallest full AFL (with respect to inclusion).* □

We now present some characterizations of other language families by (iterated) applications operations to some languages of certain language families.

Theorem 4.25 *For any recursively enumerable language L, there are two context-free languages L_1 and L_2 and a homomorphism such that $L = h(L_1 \cap L_2)$.*

Proof. Let L be a recursively enumerable language. Let $G' = (N, T, P, S)$ be a grammar such that $L(G) = L$. We construct the grammar $G = (N, T, P \cup \{S \to S\}, S)$. It is obvious that $L(G) = L$ also holds. Moreover, G has the property that any word $w \in L$ can be generated by a derivation of odd length. This follows from the fact that a derivation D of w of even length can be transformed in a derivation of odd length as follows: we start with $S \to S$ and perform then D.

Let $T' = \{a' \mid a \in T\}$ be a set of primed versions of letters of T. Let c be an additional letter not in $N \cup T \cup T'$. Furthermore, let $g : (N \cup T)^* \to (N \cup T')^*$ be the homomorphism given by $g(A) = A$ for $A \in N$, $g(a) = a'$ for $a \in T$.

We consider the languages

$$\begin{aligned} U_1 &= \{g(y^R u^R x^R)\, c\, g(xvy) \mid u \to v \in P \cup \{S \to S\}\}, \\ U_2 &= \{g(y^R u^R x^R)\, c\, xvy \mid u \to v \in P \cup \{S \to S\}\}, \\ U_3 &= \{g(xuy)\, c\, g(y^R v^R x^R) \mid u \to v \in P \cup \{S \to S\}\}. \end{aligned}$$

We note that $h(w^R)\, c\, h(w')$ is in U_1 if and only if $w \Longrightarrow w'$ holds in G. Moreover, $h(w^R)\, c\, w'$ is in U_2 as well as $h(w)\, c\, h((w')^R)$ is in U_3 if and only if $w \Longrightarrow w'$ holds in G.

Now we define

$$L_1 = (U_1\{c\})^* U_2 \text{ and } L_2 = \{Sc\}(U_3\{c\})^* T^*.$$

Then a word

$$g(w_0)\, c\, g(w_1)\, c\, g(w_2)\, c\, g(w_3)\, c \ldots c\, g(w_{2n})\, c\, w_{2n+1}$$

is in L_1 if and only if $w_{2i}^R \Longrightarrow w_{2i+1}$ for $0 \le i \le n$ and

$$S\, c\, g(w_1)\, c\, g(w_2)\, c\, g(w_3)\, c\, g(w_4)\, c \ldots c\, g(w_{2n-1})\, c\, g(w_{2n})\, c\, w_{2n+1}$$

is in L_2 if and only if $w_{2i+1} \Longrightarrow w_{2i+2}^R$ for $0 \le i \le n-1$ and w_{2n+1} is in T^*. Thus a word

$$g(w_0)\, c\, g(w_1)\, c\, g(w_2)\, c\, g(w_3)\, c \ldots c\, g(w_{2n})\, c\, w_{2n+1}$$

is in the intersection $L_1 \cap L_2$ if and only if

$$g(w_0) = S, w_{2n+1} \in T^*, w_{2i}^R \Longrightarrow w_{2i+1} \text{ for } 0 \le i \le n, w_{2i+1} \Longrightarrow w_{2i+2}^R \text{ for } 0 \le i \le n-1.$$

Hence there is a derivation

$$S \Longrightarrow w_1 \Longrightarrow w_2^R \Longrightarrow w_3 \Longrightarrow w_4^R \Longrightarrow \ldots \Longrightarrow w_{2n-1} \Longrightarrow w_{2n}^R \Longrightarrow w_{2n+1} \in T^*$$

in G. Therefore $w_{2n+1} \in L(G) = L$.

Conversely, if

$$S \Longrightarrow v_1 \Longrightarrow v_2 \Longrightarrow v_3 \Longrightarrow v_4 \Longrightarrow \ldots \Longrightarrow v_{2n-1} \Longrightarrow v_{2n} \Longrightarrow v_{2n+1} \in T^*$$

is a derivation in G (remember that, without loss of generality, it has odd length), then the words

$$S\, c\, g(v_1)\, c\, g(v_2^R)\, c\, g(v_3)\, c \ldots c\, g(v_{2n}^R)\, c\, v_{2n+1}$$

is in the intersection of L_1 and L_2.

Let now $h : (N \cup T' \cup T \cup \{c\})^* \to T^*$ be the homomorphism given by $h(a) = a$ for $a \in T$ and $h(X) = \lambda$ for $X \in N \cup T' \cup \{c\}$. Then the application of h to $L_1 \cup L_2$ cancels all letters which are not in T, i. e. the word behind the last c remains. Thus, by the above considerations, we get exactly the words of L.

It remains to show that L_1 and L_2 are context-free. By the closure properties of the family of context-free languages it is sufficient to show that U_1, U_2 and U_3 are context-free.

The context-free grammar $H = (\{A, A', A''\}, N \cup T', P', A)$, where

$$\begin{aligned} P &= \{Y \to XYX \mid Y \in \{A, A'\}, X \in N \cup T'\} \cup \{A' \to c\} \\ &\quad \cup \{A \to u^R A' v \mid u \to v \in P \cup \{S \to S\}, \end{aligned}$$

generates U_1 since any derivation has the form

$$\begin{aligned} A &\Longrightarrow x_1 A x_1 \Longrightarrow^* x_1 x_2 \ldots x_n A x_n x_{n-1} \ldots x_2 x_1 \\ &\Longrightarrow x_1 x_2 \ldots x_n u^R A v x_n x_{n-1} \ldots x_2 x_1 \\ &\Longrightarrow x_1 x_2 \ldots x_n u^R y_1 A' y_1 v x_n x_{n-1} \ldots x_2 x_1 \\ &\quad \vdots \\ &\Longrightarrow x_1 x_2 \ldots x_n u^R y_1 y_2 \ldots y_m A' y_m y_{m-1} \ldots y_1 v x_n x_{n-1} \ldots x_2 x_1 \\ &\Longrightarrow x_1 x_2 \ldots x_n u^R y_1 y_2 \ldots y_m c y_m y_{m-1} \ldots y_1 v x_n x_{n-1} \ldots x_2 x_1. \end{aligned}$$

It is left to the reader to construct analogous context-free grammars for U_2 and U_3. □

Theorem 4.26 *For any recursively enumerable language L, there are context-free languages L_1, L_2, L_3, and L_4 such that*

$$L = \{v \mid uv \in L_1 \text{ for some } u \in L_2\} \text{ and } L = \{u \mid uv \in L_1 \text{ for some } v \in L_2\}.$$

Proof. Let L be a recursively enumerable language. Let $G = (N, T, P, S)$ be a grammar such that $L(G) = L$. Without loss of generality, we assume that $S \to S$ is in P in order to ensure that each word of L has a derivation of length at most 2.We now consider the two languages

$$\begin{aligned} L_1 &= \{w_n\, c\, w_{n-1}\, c \dots c\, w_1\, cc\, w'_1\, c\, w'_2 \dots c\, w'_{n-1} ccc w'_n \mid \\ &\qquad n \geq 2,\ w_i = y_i^R u_i^R x_i^R,\ w'_i = x_i v_i y_i,\ x_i \in (N \cup T)^*,\ y_i \in (N \cup T)^*, \\ &\qquad u_i \to v_i \in P,\ 1 \leq i \leq n,\ w'_n \in T^*\} \end{aligned}$$

(by definition, $w_i^R \Longrightarrow w_i$ holds in G) and

$$L_2 = \{z_m^R c z_{m-1}^R c \dots c z_1^R c S c c z_1 c z_2 \dots c z_{m-1} c z_m ccc \mid m \geq 1, z_i =\in (N \cup T)^*, 1 \leq i \leq n\}.$$

Assume that $w \in L_1$ has a decomposition $w = uv$ with $u \in L_2$. Then we get

$$\begin{aligned} w &= z_m^R\, c\, z_{m-1}^R\, c \dots c\, z_1^R\, c\, S\, cc\, z_1\, c\, z_2 \dots c\, z_m ccc w'_{m+1}, && (4.5)\\ u &= z_m^R\, c\, z_{m-1}^R\, c \dots c\, z_1^R\, c\, S\, cc\, z_1\, c\, z_2 \dots c\, z_m ccc, && (4.6)\\ v &= w'_n && (4.7) \end{aligned}$$

with the additional relations $S \Longrightarrow z_1$, $z_i \Longrightarrow z_{i+1}$ for $1 \leq i \leq m-1$ (since $(z_i^R)^R = z_i$, $z_m \Longrightarrow w'_{m+1}$, and $w'_{m+1} \in T^*$. Therefore

$$S \Longrightarrow z_1 \Longrightarrow z_2 \Longrightarrow \dots \Longrightarrow z_m \Longrightarrow w'_{m+1} \qquad (4.8)$$

is a terminating derivation in G, i. e., $w'_{m+1} \in L(G) = L$. Therefore the set

$$L' = \{v \mid uv \in L_1 \text{ for some } u \in L_2\}$$

is contained in L.

Conversely, each terminating derivation (4.8) can be transformed into words $w \in L_1$, $u \in L_2$, and v with (4.5), (4.6), and (4.7) which implies that $L \subseteq L'$.

Thus the first relation of the statement is shown.

If L is a recursively enumerable language, then $L^R \in \mathcal{L}(RE)$ also holds. Then there are languages L_1 and L_2 such that

$$L^R = \{v \mid uv \in L_1 \text{ and } u \in L_2\}.$$

Hence

$$\begin{aligned} L &= \{v^R \mid uv \in L_1 \text{ and } u \in L_2\} \\ &= \{v^R \mid v^R u^R \in L_1^R, u^R \in L_2^R\}. \end{aligned}$$

If we choose $L_3 = L_1^R$ and $L_4 = L_2^R$ we get the desired second relation of the statement.

Since $\mathcal{L}(CF)$ is closed under reversal, it remains to prove that L_1 and L_2 are context-free. We present context-free grammars G_1 and G_2 which generate L_1 and L_2, respectively, as can be seen easily. We set

$$\begin{aligned}
G_1 &= (\{S, A, B, C\},\ N \cup T \cup \{c\},\ P_1,\ S), \\
P_1 &= \{S \to aSa \mid a \in T\} \cup \{S \to u^R A v \mid u \to v \in P, v \in T^*\} \\
&\quad\ \cup \{A \to aAa \mid a \in T\} \cup \{A \to cBccc\} \\
&\quad\ \cup \{B \to aBa \mid a \in T\} \cup \{B \to u^R C v \mid u \to v \in P\} \\
&\quad\ \cup \{C \to aCa \mid a \in T\} \cup \{C \to cBc, C \to cc\}, \\
G_2 &= (\{S', A', B'\},\ N \cup T \cup \{c\},\ P_2,\ S'), \\
P_2 &= \{S' \to A'ccc, A' \to cB'c, B' \to cB'c, B' \to cScc\} \cup \{A' \to aA'a \mid a \in T\} \\
&\quad\ \cup \{B' \to aB'a \mid a \in T\}
\end{aligned}$$

□

Theorem 4.27 *For any recursively enumerable language $L \subset V^*$, there is a context-sensitive language L' and letters c_1 and c_2 not contained in V such that $L' \subseteq L\{c_1\}\{c_2\}^*$ and, for any $w \in L$, there is a number $i \geq 1$ such that $wc_1c_2^i \in L'$.*

Proof. Let L be a recursively enumerable language, and let $G = (N, T, P, S)$ be a phrase structure grammar generating L. We construct the monotone grammar

$$G' = (N \cup \{C, S'\}, T \cup \{c_1, c_2\}, P', S')$$

where P' consists of all rules of the following forms:

- $S' \to Sc_1$
  (this rule introduces the start symbol of G and the additional symbol c_1),
- $\alpha \to \beta$ where $\alpha \to \beta \in P$ and $|\alpha| \leq |\beta|$,
  $\alpha \to \beta C^p$ where $\alpha \to \beta \in P$ and $|\alpha| - |\beta| = p > 0$
  (these monotone rules simulate the rules of P),
- $Ca \to aC$ for $a \in N \cup T \cup \{c_1\}$
  (by these rules, C can be shifted to the right),
- $C \to c_2$
  (terminating rules for C).

By the explanations added to the rules, it is obvious that $v \in L(G')$ if and only if $v = c_2^{r_1} w_1 c_2^{r_2} w_2 \dots c_2^{r_k} w_k c_2^s$ where $r_i \geq 0$ for $1 \leq i \leq k$, $s \geq 0$ and $w_1 w_2 \dots w_n = wc_1$ for some $w \in L$. Since $L(G) \in \mathcal{L}(CS)$ and $\mathcal{L}(CS)$ is closed under intersection (with regular sets), $L' = L(G') \cap T^*\{c_1\}\{c_2\}^*$ is a context-sensitive language, too. It is easy to see that L' has the properties required in the statement. □

4.2.2 Characterizations of Regular Language Families by Congruence Relations

Before we present a further characterization of regular languages, we recall some notions on equivalence and congruence relations.

A binary relation R on a set M is a subset of $M \times M$. Instead of $(a, b) \in R$ we often write aRb.

A binary relation $\sim$ It is called an *equivalence relation* if it satisfies the following three properties:

– for all $a \in M$, $a \sim a$ (reflexivity),
– for all $a, b \in M$, $a \sim b$ implies $b \sim a$ (symmetry),
– for all $a, b, c \in M$, $a \sim b$ and $b \sim c$ imply $a \sim c$ (transitivity).

For any element $a \in M$ and any equivalence relation $\sim$ on M, we define the *equivalence class*

$$K_\sim(a) = \{b \mid b \in M,\ a \sim b\}.$$

A subset M' of M is called an equivalence class of the equivalence relation $\sim$ on M if $M' = K_\sim(a)$ for some $a \in M$. It is well-known that the equivalence classes of an equivalence relation form a partition of the set M (i. e., they are non-empty sets and pairwise disjunct, and their union is M).

The *index* of an equivalence relation $\sim$ is the cardinality of the set of its equivalence classes and is denoted by $Ind(\sim)$. An equivalence relation $\sim$ is said to be of finite index if $Ind(\sim)$ is finite.

Let the set M be equipped with an operation $\circ$. An equivalence relation $\sim$ on M is called a congruence if, for all $a, b, c \in M$, $a \sim b$ implies $a \circ c \sim b \circ c$. If $\sim$ is a congruence, then equivalence is preserved if the operation is applied.

An equivalence relation $\sim$ on M is called a refinement of a set $R \subset M$ if, for all $a, b \in R$ with $a \sim b$, we have $a \in R$ if and only if $b \in R$. This means that $a \in R$ implies that all elements equivalent to a belong to R, too. Thus any equivalence class of an element of R is a subset of R. By this fact, the notation refinement of R is justified.

Let R be a subset of M. An equivalence relation is called an R-relation if it is a congruence relation of finite index and a refinement of R.

Example 4.28 Let $\mathcal{A} = (X, Z, z_0, \delta, F)$ be a finite automaton and $R = T(\mathcal{A})$. Without loss of generality we assume that each state of Z is accessible from the initial state, i. e., for any $z \in Z$ there is a word $x \in X^*$ such that $\delta^*(z_0, x) = z$ (if this is not the case we cancel all states which are not accessible). We define on X^* the relation $\sim_\mathcal{A}$ by

$$x \sim_\mathcal{A} y \quad \text{if and only if} \quad \delta^*(z_0, x) = \delta^*(z_0, y).$$

Obviously, $\sim_\mathcal{A}$ is an equivalence relation. We show that $\sim_\mathcal{A}$ is an R-relation.

Let $x \sim_\mathcal{A} y$. By definition of $\sim_\mathcal{A}$, $\delta^*(z_0, x) = \delta^*(z_0, y)$. Thus we get

$$\delta^*(z_0, xw) = \delta(\delta^*(z_0, x), w) = \delta(\delta^*(z_0, y), w) = \delta^*(z_0, yw)$$

and therefore $xw \sim_\mathcal{A} yw$ for any $w \in X^*$, which proves that $\sim_\mathcal{A}$ is a congruence.

Again, let $x \sim_\mathcal{A} y$. Moreover, let $x \in R$. Then $\delta^*(z_0, x) = \delta^*(z_0, y)$ und $\delta^*(z_0, x) \in F$. Hence $\delta^*(z_0, y) \in F$, which implies $y \in R$. Analogously, $y \in R$ implies $x \in R$. Thus $\sim_\mathcal{A}$ is a refinement of R.

Let $x \in X^*$. Furthermore, let $\delta^*(z_0, x) = z$. Then, for the equivalence class $K_{\sim_\mathcal{A}}(x)$ of $x \in X^*$, we obtain the following relations

$$\begin{aligned} K_{\sim_\mathcal{A}}(x) &= \{y \mid x \sim_\mathcal{A} y\} \\ &= \{y \mid \delta^*(z_0, x) = \delta^*(z_0, y)\} \\ &= \{y \mid \delta^*(z_0, y) = z\}. \end{aligned}$$

Moreover,

$$\begin{aligned}\{y \mid \delta^*(z_0, y) = z\} &= \{y \mid \delta^*(z_0, y) = \delta^*(z_0, x)\}\\ &= \{y \mid y \sim_{\mathcal{A}} x\}\\ &= K_{\sim_{\mathcal{A}}}(x).\end{aligned}$$

Therefore there is a one-to-one function from the equivalence classes of $\sim_{\mathcal{A}}$ to the states of $\mathcal{A}$. Hence the number of states and the number of equivalence classes coincide. Since the number of states is finite, the index of $\sim_{\mathcal{A}}$ is finite, too.

Example 4.29 For a language $R \subseteq X^*$ we define the relation $\sim_R$ as follows: $x \sim_R y$ holds if and only if, for all words $w \in X^*$, the word xw is in R if and only if yw is in R.

We prove that $\sim_R$ is a congruence which refines R.

Let $x \sim_R y$, $a \in X$ und $w \in X^*$. Then $aw \in X^*$ and , by definition of $\sim_R$, $xaw \in R$ if and only if $yaw \in R$. Since w can be arbitrarily chosen, we get $xa \sim_R ya$. Hence $\sim_R$ is a congruence.

If we choose $w = \lambda$, by definition of $\sim_R$, $x \sim_R y$ implies that $x \in R$ if and only if $y \in R$. Therefore $\sim_R$ is a refinement of R.

We note that $\sim_R$ has not necessarily a finite index. To see this we consider

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

and two words a^k und a^ℓ with $k \neq \ell$. Because $a^k b^k \in R$ and $a^\ell b^k \notin R$, a^ℓ and a^k are not equivalent. Thus there are at most as many equivalence classes as powers of a and thus as many as natural numbers. Thus the index of $\sim_R$ is infinite.

We now present the characterization of regular sets. It was first shown by J. Myhill in [23] and A. Nerode[2] in [24], and therefore it is often called Myhill-Nerode theorem.

Theorem 4.30 *The following three statements are equivalent for a language $R \subseteq X^*$.*

i) *R is regular.*
ii) *There is an R-relation.*
iii) *The relation $\sim_R$ (of Example 4.29) has finite index.*

Proof. i) $\Longrightarrow$ ii). If R is a regular language, then there is a deterministic finite automaton $\mathcal{A}$ such that $R = T(\mathcal{A})$ (see Theorem 3.48). Then we construct the relation $\sim_{\mathcal{A}}$ according to Example 4.28. By Example 4.28, $\sim_{\mathcal{A}}$ is an R-relation.

ii) $\Longrightarrow$ iii). By supposition there is an R-relation $\sim$ on X^* which has finite index. We now prove that $Ind(\sim_R) \leq Ind(\sim)$.

Let $x \sim y$ and $w \in X^*$. Then $xw \sim yw$ because $\sim$ is a congruence. Furthermore, since $\sim$ is an R-relation, we get $xw \in R$ if and only if $yw \in R$. By definition of $\sim_R$, we get $x \sim_R y$. Thus we have shown that $x \sim y$ implies $x \sim_R y$. Therefore

$$\{y \mid y \sim x\} \subseteq \{y \mid y \sim_R x\}.$$

Hence any equivalence class of $\sim$ is contained in an equivalence class of $\sim_R$ which implies $Ind(\sim_R) \leq Ind(\sim)$. Because $\sim$ is of finite index, $\sim_R$ is of finite index,too.

[2] John R. Myhill (1923–1987) and Anil Nerode (*1932), both North-American mathematicians

iii) $\Longrightarrow$ i). We construct an automaton where the finitely many equivalence classes of $\sim_R$ are taken as states and the input $a \in X$ transforms an equivalence class $K_{\sim_R}(x)$, $x \in X^*$, into the equivalence class $K_{\sim_R}(xa)$. Formerly, we set

$$\mathcal{A} = (X, \{K_{\sim_R}(x) \mid x \in X^*\}, K_{\sim_R}(\lambda), \delta, \{K_{\sim_R}(y) \mid y \in R\})$$

where

$$\delta(K_{\sim_R}(x), a) = K_{\sim_R}(xa).$$

We first note that the definition of $\mathcal{A}$ is correct since $\sim_R$ is an R-relation of finite index (the set of states is finite, and $K_{\sim_R}(x) = K_{\sim_R}(y)$ implies $K_{\sim_R}(xa) = K_{\sim_R}(ya)$ because $\sim_R$ is a congruence, i. e., $K_{\sim_R}(x) = K_{\sim_R}(y)$ or equivalently $x \sim_R y$ implies $xa \sim ya$ which is equivalent to $K_{\sim_R}(xa) = K_{\sim_R}(ya)$). By induction on the length of w, it is easy to show that $\delta^*(K_{\sim_R}(\lambda), x) = K_{\sim_R}(x)$ holds for all $x \in X^*$. Thus we get

$$T(\mathcal{A}) = \{x \mid \delta^*(K_R(\lambda), x) \in \{K_R(y) \mid y \in R\}\} = \{x \mid K_R(x) \in \{K_R(y) \mid y \in R\}\} = R.$$

Consequently, R is regular. $\square$

Finally, we mention that in the part ii) $\Longrightarrow$ iii) of the proof we have shown the following corollary.

Corollary 4.31 *Let R be a regular language. Then $Ind(\sim) \geq Ind(\sim_R)$ holds for any R-relation $\sim$.* $\square$

Chapter 5

Decision Problems for Formal Languages

In this section we ask whether or not a given grammar or given grammars have a certain property. First of all we are interested whether it is decidable that the grammars have the property. If the affirmative answer is positive, we also estimate the complexity of the decision procedure.

One of the basic questions in the theory of formal languages is the membership problem which can be stated as follows.

Membership problem:

Given:	a grammar $G = (N, T, P, S)$ and a word $w \in T^*$
Question:	Does $w \in L(G)$ hold?

The membership problem occurs very natural in programming languages. If the grammar G describes a programming languages, then the words w under consideration are written programs, and we ask whether or not the written program is syntactically correct. Therefore the membership problem has to be solved in the parsing process and the compilation with respect to a programming language. We note that we do not discuss the semantical correctness of the program in the context of the membership problem.

Obviously, the given version of the membership problem assumes that the language L under consideration is given by a grammar, i. e., $L = L(G)$. We know from Chapter 3 that languages can also be obtained as the accepted set of words of some automaton. Hence we also have the following formulation of the membership problem.

Membership problem:

Given:	an automaton $\mathcal{A}$ with input set X and a word $w \in X^*$
Question:	Does $w \in T(\mathcal{A})$ hold?

If we are only interested in the question whether or not the membership of a word of a language L is decidable, then it is not of importance which formulation we use, because we have seen in Chapter 3 that we can algorithmically transform a given grammar into an automaton and vice versa. Therefore if we study the status of decidability, then we shall use that description which is the most appropriate (where we prefer the version based on grammars). However, if we are interested in the complexity of the deciding procedure, then it is essential which description of the language is given.

An analogous situation holds if we consider different types of grammars which are able to generate the same family of languages, e. g., context-sensitive and monotone grammars.

Besides the membership problem we study the following problems in this section, where we present only the grammatical version of the formulation.

Emptiness problem:
Given: a grammar $G = (N, T, P, S)$
Question: Is $L(G)$ the empty language?

Finiteness problem:
Given: a grammar $G = (N, T, P, S)$
Question: Is $L(G)$ a finite language?

Equivalence problem:
Given: two grammars $G = (N, T, P, S)$ and $G' = (N', T, P', S')$
Question: Does $L(G) = L(G')$ hold?

The given formulations are very general since they do not restrict the type of the grammar (or of the automaton). In the sequel we discuss the problems for grammars of special types, e. g., for regular or context-free grammars. This means that the given grammars are of the type under consideration.

We start with two statements on the decidability of the membership problem for arbitrary and context-sensitive (or equivalently monotone) grammars.

Theorem 5.1 *The membership problem for (arbitrary) phase structure grammars is undecidable.*

Proof. Assume that the membership problem for arbitrary grammars is decidable. Let a Turing machine $\mathcal{M}$ be given. Without loss of generality we can assume that $\mathcal{M}$ halts on an input word w if and only if w is accepted (see Lemma 3.8). From $\mathcal{M}$ we can construct a phrase structure grammar G such that $L(G) = T(\mathcal{M})$ (see Lemma 3.12. Therefore $\mathcal{M}$ halts on w if and only if $w \in L(G)$. Since we can decide $w \in L(G)$ by assumption, we can decide whether or not $\mathcal{M}$ halts on w. This contradicts Theorem 3.31.

Therefore our assumption is false, i. e., the membership problem for arbitrary grammars is undecidable. □

Theorem 5.2 *The membership problem for context-sensitive or monotone grammars is decidable.*

Proof. We only give the proof for monotone grammars since any context-sensitive grammar is monotone.

Let a monotone grammar $G = (N, T, P, S)$ and a word $w \in T^*$ be given. If $w = \lambda$, we have only to check whether or not $S \to \lambda$ belongs to P, because $\lambda \in L(G)$ if and only if $S \to \lambda \in P$ by the definition of monotone grammars. Thus in the remaining proof we assume that $w \neq \lambda$.

Let

$$S = w_0 \Longrightarrow w_1 \Longrightarrow w_2 \Longrightarrow \ldots \Longrightarrow w_n = w$$

be a derivation of w in G. If $w_i = w_j$ for two words w_i and w_j with $i < j$, then

$$S = w_0 \Longrightarrow w_1 \Longrightarrow w_2 \Longrightarrow \dots \Longrightarrow w_i \Longrightarrow w_{j+1} \Longrightarrow w_{j+2} \Longrightarrow \dots \Longrightarrow w_n = w$$

is also a derivation of w in G. Thus we can assume that there is a derivation of w in G such that no intermediate sentential form occurs more than once. Because $|w_{i-1}| > |w_i|$ is impossible in a monotone grammar and there are at most $\#(V)^k$ words of length k over $V = N \cup T$, in any derivation starting with w_i of length k we obtain a word longer than w_i after at most $\#(V)^k$ steps. Thus there is a derivation of w which has at most the length $|w|\#(V)^{|w|+1}$. Since there are at most $\#(P)^{|w|\#(V)^{|w|+1}}$ derivations of length $|w|\#(V)^{|w|+1}$, we can check all derivations of this length (without a repetition of sentential forms) whether or not they lead to the given word w. □

The procedure given in the proof to decide whether or not $w \in L(G)$ holds has double exponential time complexity in $|w|$ (since the number of derivation to be checked) is double exponential in $|w|$) and exponential space complexity (because the number of sentential forms which have to be stored to ensure that no sentential form occurs two times in a derivation is exponential in worst case). Presently, no algorithm with a space complexity lower than exponential is known.

By Theorem 5.2, monotone language are recursive. Moreover, any recursive function is recursively enumerable by Definition 3.9 and Theorem 3.19. Taking into consideration Theorem 3.11, we can finish our results on the Chomsky hierarchy and settle the last properness of the inclusions in Theorem 2.37.

Theorem 5.3 *$\mathcal{L}(CS) \subset \mathcal{L}(RE)$.* □

We are now also in position to add the closure property of $\mathcal{L}(CS)$ under homomorphisms.

Lemma 5.4 *$\mathcal{L}(CS)$ is not closed under arbitrary homomorphisms, but under non-erasing homomorphisms.*

Proof. In order to prove the closure under non-erasing homomorphisms we repeat the proof for $\mathcal{L}(CF)$ (see proof of Theorem 4.9) using the Kuroda normal form instead of the Chomsky normal form. The newly introduced rules $A \to h(a)$ are allowed since $h(a) \neq \lambda$.

Let L be a language in $\mathcal{L}(RE) \setminus \mathcal{L}(CS)$. Then there is a grammar $G = (N, T, P, S)$ in Kuroda normal form such that $L(G) = L$. We consider the grammar $G' = (N, T \cup \{c\}, P', S)$ where P' is constructed from P by a replacement of any rule of the form $A \to \lambda \in P$ by $A \to c$. Then G' is a monotone grammar and therefore $L(G') \in \mathcal{L}(CS)$. The language $L(G')$ differs from $L(G)$ that in some words the additional letter c occurs. Obviously, $h(L(G') = L(G)$ for the homomorphism h with $h(c) = \lambda$ and $h(a) = a$ for $a \in T$. If $\mathcal{L}(CS)$ is closed under arbitrary homomorphisms, we get that $h(L(G') = L(G) = L \in \mathcal{L}(CS)$ in contrast to our choice of L. □

Obviously, since the membership problem for monotone grammars is decidable and any context-free or regular grammar can be transformed into a monotone context-free or monotone regular grammar without a change of the generated language (one has to eliminate erasing rules), the membership problems for context-free and regular grammars are also decidable. However, for context-free and regular grammars, we can give much faster decision procedures, more precisely, there are algorithms to decide the membership problem for context-free and regular grammars which are of cubic or linear time complexity in the length of the word under consideration.

Theorem 5.5 *i) The membership problem for a given context-free grammar* $G = (N, T, P, S)$ *in Chomsky normal form and a word* $w \in T^*$ *can be decided in time* $O(\#(P) \cdot |w|^3\})$.

ii) The membership problem for a given context-free grammar $G = (N, T, P, S)$ *and a word* $w \in T^*$ *can be decided in time* $O(k(G)^2 \cdot |w|^3)$.

Proof. i) Let a context-free grammar $G = (N, T, P, S)$ in Chomsky normal form and a word $w = a_1a_2 \dots a_n$ over T of length n be given. We construct inductively sets $V_{i,j}$ for $0 \leq i < j \leq n$. First we set

$$V_{i-1,i} = \{A \mid A \in N, A \to a_i \in P\}. \tag{5.1}$$

If the sets $V_{i,k}$ and $V_{k,j}$ for $i < k < j$ are already defined, we set

$$V_{i,j} = \{A \mid A \in N, A \to BC \in P, B \in V_{i,k}, C \in V_{k,j} i < k < j\}. \tag{5.2}$$

The set $V_{i,j}$ can be constructed by (5.2) in at most $\#(P) \cdot n$ steps since there are at most n possible values k and for each k one has to go through all rules of P. Since we have to construct $\frac{n(n+1)}{2}$ sets, the construction of all sets $V_{i,j}$, $0 \leq i < j \leq n$, can be done in time at most $\frac{\#(P)n^2(n+1)}{2}$.

We now prove by induction on the difference $i - j$ that

$$V_{i,j} = \{A \mid A \in N, A \stackrel{*}{\Longrightarrow} a_{i+1}a_{i+2} \dots a_j\}. \tag{5.3}$$

For $j - i = 1$, (5.3) holds by our setting (5.1).

Let $A \in V_{i,j}$ and $j - i \leq 2$. By (5.2), there are nonterminals $B \in V_{i,k}$ and $C \in V_{k,j}$ with $A \to BC \in P$ and $k - i < j - i$ and $j - k < j - i$. By induction hypothesis,

$$B \stackrel{*}{\Longrightarrow} a_{i+1}a_{i+2} \dots a_k \quad \text{und} \quad C \stackrel{*}{\Longrightarrow} a_{k+1}a_{k+2} \dots a_j.$$

Therefore we obtain

$$A \Longrightarrow BC \stackrel{*}{\Longrightarrow} a_{i+1}a_{i+2} \dots a_k C \stackrel{*}{\Longrightarrow} a_{i+1}a_{i+2} \dots a_k a_{k+1}a_{k+2} \dots a_j.$$

Conversely, let $A \stackrel{*}{\Longrightarrow} a_{i+1}a_{i+2} \dots a_j$. Because G is in Chomsky normal form, there are nonterminals B and C and an integer k with $i < k < j$ such that

$$A \to BC \in P, \quad B \stackrel{*}{\Longrightarrow} a_{i+1}a_{i+2} \dots a_k, \quad C \stackrel{*}{\Longrightarrow} a_{k+1}a_{k+2} \dots a_j.$$

By induction hypothesis, we have $B \in V_{i,k}$ and $C \in V_{k,j}$. By (5.2), $A \in V_{i,j}$.

Hence (5.3) is shown.

From (5.3), we obtain immediately $S \stackrel{*}{\Longrightarrow} a_1a_2 \dots a_n = w$ if and only if $S \in V_{0,n}$. Thus $w \in L(G)$ and $S \in V_{0,n}$ are equivalent. Thus, to decide whether or not $w \in L(G)$, it is sufficient to construct the sets $V_{i,j}$, $0 \leq i < j \leq n$, and to check whether or not $S \in V_{0,n}$. Therefore $w \in L(G)$ can be decided in time $O(\#(P) \cdot |w|^3)$ by the above estimation for the construction of the sets $V_{i,j}$.

ii) Let $G = (N, T, P, S)$ be a context-free grammar. We construct a context-free grammar $G' = (N', T, P', S')$ in Chomsky normal form from G such that $L(G) = L(G')$. G' can be constructed in time $O(k(G)^2)$ and satisfies $\#(P') \leq k(G') \in O(k(G)^2)$ by Theorem 2.26. Now the result follows from i). □

The algorithm presented in the preceding proof was independently given by J. COCKE, D. H. YOUNGER, and T. KASAMI[1] in the papers [4], [33], and [15]. Therefore it is often called Cocke-Younger-Kasami algorithm. We illustrate the algorithm by an example.

Example 5.6 Let the context-free grammar

$$G = (\{S, T, U\}, \{a, b\}, P, S)$$

with

$$P = \{S \to ST, T \to TU, T \to TT, U \to TS, S \to a, T \to a, U \to b\}$$

be given. We first look whether or not the word $w = aabaa$ belongs to $L(G)$. we have to determine the associated sets $V_{i,j}$, where $0 \leq i < j \leq 5$. We get

$$\begin{aligned}
V_{0,1} &= \{A \mid A \to a \in P\} = \{S, T\}, \\
V_{1,2} &= \{A \mid A \to a \in P\} = \{S, T\}, \\
V_{2,3} &= \{A \mid A \to b \in P\} = \{U\}, \\
V_{0,2} &= \{A \mid A \to BC \in P, B \in V_{0,1}, C \in V_{1,2}\} = \{S, T, U\}, \\
V_{1,3} &= \{A \mid A \to BC \in P, B \in V_{1,2}, C \in V_{2,3}\} = \{T\}, \\
V_{0,3} &= \{A \mid A \to BC \in P, B \in V_{0,1}, C \in V_{1,3}\} \\
&\quad \cup \{A' \mid A' \to B'C' \in P, B' \in V_{0,2}, C' \in V_{2,3}\} \\
&= \{S, T\} \cup \{T\} = \{S, T\}.
\end{aligned}$$

The remaining sets can be seen from the following table where the ith symbol of w is given in the meet of the row i and column i and the set $V_{i,j}$ is given in the meet of row i and column j and instead of the sets only their elements are given.

	0	1	2	3	4	5
0		S, T	S, T, U	S, T	S, T, U	S, T, U
1		a	S, T	T	T, U	T, U
2			a	U	$\emptyset$	$\emptyset$
3				b	S, T	S, T, U
4					a	S, T
5						a

Because $S \in V_{0,5}$, $w = aabaa \in L(G)$.

For $v = abaaa$ we get the following table.

[1] JOHN COCKE (1925–2002)and DANIEL H. YOUNGER, both North-American computer scientists; TADAO KASAMI (1930-2007), Japanese scientist worked in information theory and theory of codes

	0	1	2	3	4	5
0		S,T	T	T,U	T,U	T,U
1		a	U	$\emptyset$	$\emptyset$	$\emptyset$
2			b	S,T	S,T,U	S,T,U
3				a	S,T	S,T,U
4					a	S,T
5						a

and therefore $v \notin L(G)$ by $S \notin V_{0,5}$.

The construction of the sets $V_{i,j}$ shows a certain analogy to the multiplication of matrices since in both cases the new element is obtained by combining the elements of the corresponding rows and columns. A detailed investigation of this analogy leads to an improvement of the Cocke-Younger-Kasami algorithm. Since the multiplication of matrices of type (n,n) can be done in time $O(n^{\log_2(7)})$ by Strassen's algorithm, L. Valiant (see [30] gave an algorithm for the membership problem which works in time $O(|w|^{\log_2(7)})$ (if the grammar is fixed and therefore its size can be considered as a constant).

For regular languages, we can considerably decrease the complexity.

Theorem 5.7 *For a regular grammar $G = (N,T,P,S)$ and a word $w \in T^*$, it is decidable in time $O(k(G)^2 \cdot |w|)$ whether or not $w \in L(G)$ holds.*

Proof. First we construct from G a regular grammar $G' = (N',T,P',S')$ which satisfies $L(G') = L(G)$ and only contains rules of the form $A \to aB$ and $A \to a$ with $A,B \in N'$ and $a \in T$ according to the proof of Theorem 2.28. This transformation can be done in time $O(k(G)^2)$. Moreover, we have $\#(N') \in O(k(G))$ and $\#(P') \in O(k(G))$.

Let $w = a_1a_2\dots a_n$. We set $M_0 = \{S\}$ and

$$M_i = \{A \mid B \to a_iA \text{ für ein } B \in M_{i-1}\}$$

for $1 \leq i \leq n-1$. The determination of M_i from M_{i-1}, $1 \leq i \leq n-1$, can be done in time $O(\#(N')\#(P'))$ since we have to go through all rules of P' for each nonterminal in M_{i-1}. It is easy to see that $A \in M_i$ if and only if $S \stackrel{*}{\Longrightarrow} a_1a_2\dots a_iA$. Now we check whether or not M_{n-1} contains a nonterminal A such that $A \to a_n \in P'$. Again, we need time $O(\#(N')\#(P'))$ for this check. If such a nonterminal exists, we have a derivation

$$S \stackrel{*}{\Longrightarrow} a_1a_2\dots a_{n-1}A \Longrightarrow a_1a_2\dots a_{n-1}a_n = w,$$

i. e., we have $w \in L(G') = L(G)$. If such a nonterminal does not exist we cannot generate a_n from an element of M_{n-1} which means that $w \notin L(G') = L(G)$. Combining all the estimations we get an algorithm to decide $w \in L(G)$ with time complexity in $O(k(G)^2|w|)$.
□

We now discuss the decidability status of the remaining decision problems.

Theorem 5.8 *The emptiness and finiteness problems are undecidable for arbitrary phrase structure grammars and monotone (or context-sensitive) grammars.*

Proof. i) We start with the undecidability of the emptiness problem for arbitrary phrase structure grammars

Let G be an arbitrary phrase structure grammars and w be an arbitrary word over the terminal set of G. Then $\{w\}$ is a regular language. By Theorem 4.6, there is a phrase structure grammar G' such that $L(G') = L(G) \cap \{w\}$. Moreover, since all proofs of the closure properties are constructive, G' can be constructed from G and w. Obviously,

$$L(G') = \begin{cases} \{w\} & \text{if } w \in L(G) \\ \emptyset & \text{otherwise} \end{cases} .$$

Therefore $L(G')$ is empty if and only if w is not contained in $L(G)$. Thus the decidability of the emptiness problem implies the decidability of the membership problem. Since the latter problem is undecidable by Theorem 5.1, the emptiness problem is undecidable, too.

ii) We now consider the emptiness problem for monotone grammars. Let $G = (N, T, P, S)$ be an arbitrary phrase structure grammar, again. We construct the grammar $G' = (N', T, P', S')$ in Kuroda normal form with $L(G') = L(G)$. Let $P' = P_1 \cup P_2$ where P_1 contains all rules of the forms $A \to BC$, $A \to B$, $AB \to CD$ and $A \to a$ with $A, B, C, D \in N'$ and $a \in T$, and P_2 contains all rules of the form $A \to \lambda$ with $A \in \lambda$. We consider the grammar

$$G'' = (N', T \cup \{\$\}, P_1 \cup \{A \to \$ \mid A \to \lambda \in P_2\}, S'.$$

Obviously, G'' is a monotone grammar. Furthermore, for any word $w \in L(G')$, there is a word $w' = \$^{n_1} w_1 \$^{n_2} w_2 \ldots \$^{n_k} w_k \$^{n_{k+1}} \in L(G'')$ where $k \geq 1$, $n_i \geq 0$ for $1 \leq i \leq k+1$, $w_j \in T^*$ for $0 \leq j \leq k$, and $w_1 w_2 \ldots w_k = w$. Conversely, if a word $v = \$^{n_1} v_1 \$^{n_2} v_2 \ldots \$^{n_k} v_k \$^{n_{k+1}}$ is in $L(G'')$ for some $k \geq 1$, $n_i \geq 0$ for $1 \leq i \leq k+1$, and $v_j \in T^*$ for $0 \leq j \leq k$, then $v_1 v_2 \ldots v_k$ is a word from $L(G')$. Thus $L(G'')$ is empty if and only if $L(G'')$ is empty. Therefore the decidability of the emptiness problem for monotone grammars implies the decidability of the emptiness problem for arbitrary grammars. By part i) of this proof, the emptiness problem for monotone grammars is undecidable.

iii) Let $G = (N, T, P, S)$ be a (monotone) grammar. Let a be a letter of T. Then $\{a\}^*$ is regular language. By the proof of Theorem 4.7, we can construct a (monotone) grammar G' such that $L(G') = L(G) \cdot \{a\}^*$. It is easy to see that $L(G')$ is finite if and only if $L(G')$ is empty if and only if $L(G)$ is empty. Hence the decidability of the finiteness problem for (monotone) grammars implies the decidability of the emptiness problem for (monotone) grammars. By i) and ii) of this proof, the finiteness problem for (monotone) grammars is undecidable, too. □

Theorem 5.9 *i) For a context-free grammar $G = (N, T, P, S)$, it is decidable in time $O(k(G)^2)$ whether or not $L(G)$ is empty.*

ii) For a context-free grammar $G = (N, T, P, S)$, it is decidable in time $O(k(G)^2)$ whether or not $L(G)$ is finite.

Proof. i) First we construct a grammar $G' = (N, T, P', S)$ where P' is obtained from P by a cancellation of all terminal symbols. Obviously, A can generate a terminal word in G if and only if A can generate the empty word in G'. Now we determine the set M of all nonterminals generating the empty word as we have done this in the proof of Lemma 2.22 which can be done in $O(k(G)^2)$. Because $L(G)$ is non-empty if and only if there is a word of T^* which can be generated from S in G if and only if S generates the empty word in G', we have only to check whether $S \in M$ holds. This requires at most a time $O(k(G))$.

ii) First we determine as in part i) the set M of all nonterminals which derive at least one terminal word. Then we construct the sets

$$\begin{aligned} Q_0 &= \{S\}, \\ Q_{i+1} &= \{B \mid A \to xBy \in P \text{ for some } A \in Q_i\} \cup Q_i \text{ for } i \geq 0, \\ Q &= \bigcup_{i \geq 0} Q_i. \end{aligned}$$

It is easy to see that $Q_i = Q_{i+1}$ implies $Q_i = Q_k$ for all $k \geq 0$. Moreover, $Q_{\#(N)} = Q_{\#(N)+1}$ since we add in each step at least one nonterminal or get $Q_i = Q_{i+1}$ for some $i \leq \#(N)$. Therefore $Q = Q_{\#(N)}$. Now as in the proof of Lemma 2.22 we can show that Q can be constructed in time $O(k(G)^2)$. Furthermore, it can easily be shown by induction on i that Q_i contains all nonterminals A such that there is a derivation $S \stackrel{*}{\Longrightarrow} uAv$ of length i. Consequently, Q consists of all nonterminals which occur in some sentential form.

Let $N'' = Q \cup M$ and P'' be the set of all rules which contain only terminals and nonterminals of N''. It is clear that $G'' = (N'', T, P'', S)$ also generates $L(G)$ because nonterminals of $A \in N \setminus M$ cannot be terminated and letters from $N \setminus Q$ cannot occur in sentential forms derivable from S. Obviously, $k(G') \leq k(G)$. Now we construct from G'' the corresponding grammar $G''' = (N''', T, P''', S)$ in Chomsky normal form. This requires $O(k(G')^2) = O(k(G)^2)$ (see Theorem 2.26). Obviously, $L(G) = L(G''')$, and therefore $L(G)$ is finite if and only if $L(G''')$ is finite. From G''' we construct the directed graph $H = (N'', E)$ where E is defined as follows: $(A, B) \in E$ if and only if there is a rule $A \to BC$ or $A \to CB$ for some $C \in N'''$ in P'''. We prove that $L(G''')$ is infinite if and only if there are a nonterminal A and a path from A to A of length $n \geq 1$.

Assume that H contains a path from A to A of length $n \geq 1$. By the definition of edges in H, this path is associated with a derivation $A \stackrel{*}{\Longrightarrow} w_1 A w_2$ with $w_1 w_2 \neq \lambda$. Because each nonterminal of N''' can generate a terminal word and occurs in some sentential form of G''' (this property of G'' is preserved by the transformation to the Chomsky normal form), for any $n \geq 0$, there is a derivation

$$S \stackrel{*}{\Longrightarrow} u_1 A u_2 \stackrel{*}{\Longrightarrow} u_1 w_1 A w_2 u_2 \stackrel{*}{\Longrightarrow} u_1 w_1^2 A w_2^2 u_2 \stackrel{*}{\Longrightarrow} \dots \stackrel{*}{\Longrightarrow} u_1 w_1^n A w_2^n u_2 \stackrel{*}{\Longrightarrow} u_1'(w_1')^n v (w_2')^n u_2'$$

where

$$u_1 \stackrel{*}{\Longrightarrow} u_1',\ u_2 \stackrel{*}{\Longrightarrow} u_2',\ w_1 \stackrel{*}{\Longrightarrow} w_1',\ w_2 \stackrel{*}{\Longrightarrow} w_2', \text{ and } A \stackrel{*}{\Longrightarrow} v$$

are terminating derivations. Therefore $L(G''')$ is infinite.

If $L(G''')$ is infinite, then there exist a nonterminal A with a derivation $A \stackrel{*}{\Longrightarrow} w_1 A w_2$ with $w_1 w_2 \neq \lambda$ (since otherwise there is only a finite number of derivations in G''' and therefore $L(G)$ is finite). Then the graph H contains a path from A to A of length $n \geq 1$.

The existence of a path from some node A to A can be checked by breadth-first-search or depth-first-search in time $O(\#(N''') + \#(E))$ and therefore in time $O(k(G)^2)$. Thus the finiteness of $L(G''')$ (and hence that of $L(G)$) can be checked in time $O(k(G)^2)$. $\square$

Finally we consider the equivalence problem.

Theorem 5.10 *The equivalence problem is undecidable for context-free grammars.*

Proof. By Theorem 3.32, it is sufficient to show that the decidability of the equivalence problem for context-free grammars implies the decidability of the Post Correspondence Problem.

Let $U = \{(u_1, v_1), (u_2, v_2), \ldots, (u_n, v_n)\}$ be a set of pairs where $u_i, v_i \in T^*$ for $1 \leq i \leq n$. We consider the context-free grammars

$$G_1 = (N, T \cup \{c\}, P, S) \quad \text{and} \quad G_2 = (N \cup \{S', S''\}, T \cup \{c\}, P \cup P', S')$$

with

$$\begin{aligned}
N &= \{S, S_u, S_r, S_l\}, \\
P &= \{S_u \to c, S_l \to c, S_r \to c\} \cup \{S \to xS_u y \mid x, y \in T, x \neq y\} \\
&\quad \cup \{S_u \to xS_u y \mid x, y \in T\} \\
&\quad \cup \bigcup_{x \in T} \{S \to xSx, S \to xS_l, S \to S_r x, S_l \to xS_l, S_r \to S_r x\}, \\
P' &= \{S' \to S, S' \to S''\} \cup \bigcup_{i=1}^{n} \{S'' \to u_i S'' v_i^R, S'' \to u_i c v_i^R\}.
\end{aligned}$$

The languages generated by these grammars are

$$L(G_1) = \{\alpha c \beta^R \mid \alpha, \beta \in T^+, \alpha \neq \beta\}$$

and

$$L(G_2) = L(G_1) \cup \{u_{i_1} u_{i_2} \ldots u_{i_k} c v_{i_k} v_{i_{k-1}} \ldots v_{i_1} \mid k \geq 1,\ 1 \leq i_j \leq n,\ 1 \leq j \leq k\}. \tag{5.4}$$

This can be seen as follows. All non-terminal sentential forms of G_1 have one of the following forms:

$$\begin{aligned}
&\alpha S \beta^R \quad \text{mit} \quad |\alpha| = |\beta|, \alpha = \beta, \\
&\alpha S_u \beta^R \quad \text{mit} \quad |\alpha| = |\beta|, \alpha \neq \beta, \\
&\alpha S_r \beta^R \quad \text{mit} \quad |\alpha| < |\beta|, \\
&\alpha S_l \beta^R \quad \text{mit} \quad |\alpha| > |\beta|.
\end{aligned}$$

Because a derivation can only terminate if one of the rules $S_u \to c$ or $S_r \to c$ or $S_l \to c$ is applied, it is clear that $L(G_1)$ contains only words of the form $\alpha c \beta^R$ with $\alpha \neq \beta$. It is easy to see that all words of this form can be obtained. From the axiom of G_2, we generate S or S''. From S the words of $L(G_1)$ are generated. Starting with S'' we can only apply the rules of the form $S'' \to u_i S'' v_i^R$ or $S'' \to u_i c v_i^R$ with $1 \leq i \leq n$, i. e., we generate a certain u_i to the left and the reversal of the corresponding v_i to the right. Consequently, we get from S'' words of the form $u_{i_1} u_{i_2} \ldots u_{i_k} c v_{i_k} v_{i_{k-1}} \ldots v_{i_1}$ where $k \geq 1$, $1 \leq i_j \leq n$, $1 \leq j \leq k$. Now (5.4) follows.

Furthermore, $L(G_1) = L(G_2)$ if and only if S'' generates no word $\alpha c \alpha^R$ for some $\alpha \in T^*$. Therefore, $L(G_1) = L(G_2)$ if and only if the Post Correspondence Problem for U has no solution. □

Theorem 5.11 *Given two regular grammars* $G_1 = (N_1, T, P_1, S_1)$ *and* $G_2 = (N_2, T, P_2, S_2)$, *it is decidable in time* $O(k^4)$, *where* $k = max\{k(G_1), k(G_2)\}$, *whether or not* $L(G_1) = L(G_2)$.

Proof. Obviously, $L(G_1) = L(G_2)$ if and only if $(L(G_1) \setminus L(G_2)) \cup (L(G_2) \setminus L(G_1)) =$. By the constructions given in Section 4.1, we can construct a regular grammar G such that

$$L(G) = (L(G_1) \setminus L(G_2)) \cup (L(G_2) \setminus L(G_1)),$$

and we have to check whether $L(G)$ is empty. According to the Exercises ??? and ??? G can be constructed in time $O(k^2)$ and $k(G) \in O(k^2)$. Taking into consideration Theorem 5.9 i), we get the statement. □