FORMAL LANGUAGES AND APPLICATIONS

Jürgen Dassow

and

Bianca Truthe

Otto-von-Guericke-Universität Magdeburg
Fakultät für Informatik

Preface

Contents

1 Fundamentals **7**
1.1 Sets and Multisets of Words . . . 7
1.2 Polynomials and Linear Algebra . . . 13
1.3 Graph Theory . . . 14
1.4 Intuitive Algorithms . . . 16

A SEQUENTIAL GRAMMARS **21**

2 Basic Families of Grammars and Languages **23**
2.1 Definitions and Examples . . . 23
2.2 Normal forms . . . 34
2.3 Iteration Theorems . . . 50

B Formal Languages and Linguistics **133**

8 Some Extensions of Context-Free Grammars **135**
8.1 Families of Weakly Context-Sensitive Grammars . . . 135
8.2 Index Grammars . . . 135
8.3 Tree-Adjoining Grammars . . . 135
8.4 Head Grammars . . . 135
8.5 Comparison of Generative Power . . . 135

9 Contextual Grammars and Languages **137**
9.1 Basic Families of Contextual Languages . . . 137
9.2 Maximally Locally Contextual Grammars . . . 137

10 Restart Automata **139**

D Formal Languages and Pictures **225**

14 Chain Code Picture Languages **227**
14.1 Chain Code Pictures . . . 227
14.2 Hierarchy of Chain Code Picture Languages . . . 235
14.3 Decision Problem for Chain Code Picture Languages . . . 239
14.3.1 Classical Decision Problems . . . 239
14.3.2 Decidability of Properties Related to Subpictures . . . 249
14.3.3 Decidability of "Geometric" Properties . . . 252

14.3.4 Stripe Languages 255
14.4 Some Generalizations 261
14.5 Lindenmayer Chain Code Picture Languages and Turtle Grammars 263
14.5.1 Definitions and some Theoretical Considerations 263
14.5.2 Applications for Simulations of Plant Developments 267
14.5.3 Space-Filling Curves 269
14.5.4 Kolam Pictures 272

15 Siromoney Matrix Grammars and Languages **275**
15.1 Definitions and Examples 277
15.2 Hierarchies of Siromoney Matrix Languages 282
15.3 Hierarchies of Siromoney Matrix Languages 282
15.4 Decision Problems for Siromoney Matrix Languages 285
15.4.1 Classical Problems 285
15.4.2 Decision Problems related to Submatrices and Subpictures 290
15.4.3 Decidability of geometric properties 294

16 Collage Grammars **301**
16.1 Collage Grammars 303
16.2 Collage Grammars with Chain Code Pictures as Parts 312

Bibliography **317**

Chapter 1

Fundamentals

In this chapter, we recall some notions, notations, and facts concerning sets, words, languages as sets of words, matrices and their eigenvalues, linear difference equations, graphs, and intuitive algorithms. which will be used in the book. Sometimes we illustrate the notions by examples, especially if the notions are used very often in the sequel and are basic for the understanding of the theory of formal languages. All facts are given without proofs; for proofs and further detailed information we refer to the textbooks [9], [19], [6], [8], [2], [5], [22].

1.1 Sets and Multisets of Words

We assume that the reader is familiar with set theory. Here we only give some notation.

If a set A is contained in a set B, then we write $A \subseteq B$. If the inclusion is proper, we write $A \subset B$. By $\#(M)$ we denote the cardinality of M.

By $\mathbb{N}$ we designate the set of all positive integers, i. e., $\mathbb{N} = \{1, 2, \dots\}$. $\mathbb{N}_0$ denotes the set of all non-negative integers, i. e., $\mathbb{N}_0 = \mathbb{N} \cup \{0\} = \{0, 1, 2, \dots\}$.

A *permutation* p of the set $M = \{1, 2, \dots, n\}$ is a one-to-one mapping of M onto itself. Obviously, p can be given as $(p(1), p(2), \dots, p(n))$. Two elements $p(i)$ and $p(j)$ of p form an *inversion* if $p(i) > p(j)$ and $i < j$. By $I(p)$ we denote the number of inversions of p.

An *alphabet* is a non-empty finite set. Its elements are called letters. Obviously, the usual set of all (small) latin letters $\{a, b, c, \dots, x, y, z\}$ is an alphabet as well as the set $U = \{a, b, c, \gamma, |, \bullet\}$ where only the first four elements are letters in the usual sense. A *word* (over an alphabet V) is a finite sequence of letters (of V). A word is written as the simple juxtaposition of its letters in the order of the sequence. According to these settings, the order of the letters in a word is very important; for example ab and ba are different words (the first letters are different and thus the sequences are different). Moreover, in contrast to words in daily life, words in the above sense do not have necessarily a meaning as can be seen from the words

$$w_1 = acbaa,\ w_2 = \gamma|\bullet\, aa \text{ and } w_3 = \bullet|\bullet\,.$$

over U. By λ we denote the empty word which contains no letter.[1] By V^* (and V^+, respectively) we designate the set of all (non-empty) words over V.

The *product* (concatenation) of words is defined as the juxtaposition of the words. For example, we have

$$w_1 \cdot w_2 = acbaa\gamma|\bullet\, aa,\ w_1 \cdot w_3 = acbaa\bullet|\bullet \text{ and } w_3 \cdot w_1 = \bullet|\bullet\, acbaa.$$

From these example we immediately see that the product is not a commutative operation. Obviously, the product is an associative operation on V^* and λ is the unit element with respect to the product, i. e.,

$$\begin{aligned}&(v_1 \cdot v_2) \cdot v_3 = v_1 \cdot (v_2 \cdot v_3) \text{ for all words } v_1, v_2, v_3,\\ &v \cdot \lambda = \lambda \cdot v = v \text{ for all words } v.\end{aligned}$$

Thus from the algebraic point of view $(V^*, \cdot)$ is a monoid and $(V^*, \cdot)$ is an associative semigroup. More precisely, V^+ is freely generated by the elements of V since the representation of a word as a product of elements of V is unique. As in arithmetics we shall mostly omit the $\cdot$ and simply write vw instead of $v \cdot w$. Furthermore, multiple products of the same word will be written as powers, i. e., instead of $\underbrace{x \cdot x \cdot \ldots \cdot x}_{n \text{ times}}$ we write x^n.

We say that v is a *subword* of w iff $w = x_1 v x_2$ for some $x_1, x_2 \in V^*$. The word v is called a *prefix* of w iff $w = vx$ for some $x \in V^*$, and v is called a *suffix* of w iff $w = xv$ for some $x \in V^*$. Continuing our example, we see that

- *acb*, *cb*, *ba*, *cba*, *a*, and *aa* are subwords of w_1,
- λ, γ, $\gamma|$, $\gamma|\bullet$, $\gamma|\bullet\, a$, and $\gamma|\bullet\, aa$ are the prefixes of w_2, and
- λ, *a*, *aa*, *baa*, *cbaa*, and *acbaa* are the suffixes of w_1.

For a alphabet V, a subset W of V and a word $w \in V^*$, by $\#_W(w)$ we denote the number of occurrences of letters from W in w. If W consists of a single letter, then we write $\#_a(w)$ instead of $\#_{\{a\}}(w)$. The *length* $|w|$ of a word w over V is defined as $|w| = \sum_{a \in V} \#_a(w)$. For example,

$$\begin{aligned}&\#_a(w_1) = 3,\ \#_b(w_1) = \#_c(w_1) = 1,\ \#_\bullet(w_1) = \#_|(w_1) = 0,\ |w_1| = 5,\\ &\#_{\{a,b,c\}}(w_2) = 2,\ \#_{\{a,|,\gamma\}}(w_2) = 4,\\ &\#_a(w_3) = \#_c(w_3) = \#_\gamma(w_3) = 0,\ \#_\bullet(w_3) = 2,\ \#_|(w_3) = 1,\ |w_3| = 3.\end{aligned}$$

Let $V = \{a_1, a_2, \ldots, a_n\}$ where $a_1, a_2, \ldots, a_n$ is a fixed order of the elements of V. Then

$$\Psi_V(w) = (\#_{a_1}(w), \#_{a_2}(w), \ldots, \#_{a_n}(w))$$

is the *Parikh vector* of the word $w \in V^*$. Using the order in which the elements of U are given above, we get

$$\pi_U(w_1) = (3,1,1,0,0,0),\ \pi_U(w_2) = (2,0,0,1,1,1) \text{ and } \pi_U(w_3) = (0,0,0,0,1,2).$$

[1] It is very often important to have such a word. For example, the application of the operation, which deletes all *a*s in a word over the alphabet $\{a, b\}$, maps the word *ababba* onto *bbb*, and we get the λ from *aaa*; without the empty word, no image of *aaa* would be defined. The reader may note the analogy between of empty word and the empty set which occurs naturally as the intersection of disjunct sets.

If the alphabet $V = \{a_1, a_2, \ldots, a_n\}$ is equipped with an order $\prec$ (i. e., without loss of generality, $a_1 \prec a_2 \prec \cdots \prec a_n$, then we extend the order to an order on V^*, which we call *lexicographic order* as follows. For two words $u \in V^*$ and $v \in V^*$, we set $u \prec v$ if and only if

– $|u| < |v|$ or

– $|u| = |v|$, $u = zxu'$ and $v = zyv'$ for some word $z \in V^*$ and some $x, y \in$ with $x \prec y$. [2]

It is easy to see that, for the alphabet $\{a, b\}$ with $a \prec b$, we get

$$\lambda \prec a \prec b \prec aa \prec ab \prec ba \prec bb \prec aaa \prec aab \prec aba \prec abb \prec baa \prec \ldots$$

Throughout the book we shall often use primed or indexed versions of the letters of an alphabet. That means that, with an alphabet V, we associate the alphabets

$$V' = \{a' \mid a \in V\} \text{ or } V^{(i)} = \{a^{(i)} \mid a \in V\}$$

where all letters are primed versions or versions with the upper index i. If $w = a_1 a_2 \ldots a_n$, $a_j \in V$ for $1 \leq j \leq n$, is a word over V, then we define the corresponding words w' over V' and $w^{(i)}$ over $V^{(i)}$ by $w' = a'_1 a'_2 \ldots a'_n$ and $w^{(i)} = a_1^{(i)} a_2^{(i)} \ldots a_n^{(i)}$, respectively. In the same way we define the corresponding words in case double primes, double indices, etc.

A *language* over V is a subset of V^*. Given a language L, we denote the set of letters occurring in the words of L by $alph(L)$. Obviously $alph(L)$ is the smallest alphabet V (with respect to inclusion) such that $L \subseteq V^*$. The set $alph(L)$ is called the alphabet of L. For a language L over the alphabet X, we define the characteristic function $\varphi_{L,X} : X^* \to \{0, 1\}$ by

$$\varphi_{L,X}(w) = \begin{cases} 1 & \text{for } w \in L \\ 0 & \text{for } w \in X^* \setminus L \end{cases} .$$

If the alphabet X is known from the context, we simply write φ_L instead of $\varphi_{L,X}$.

For a language $L \subseteq V^+$, we set

$$\pi_V(L) = \{\pi_V(w) \mid w \in L\}.$$

Since languages are sets, union, intersection and set-theoretic difference of two languages are defined in the usual way. Essentially, this also holds for complement; we have only to say which set is taken as universe. We set $C(L) = \overline{L} = alph(L)^* \setminus L$, i. e., we take the set of all words over the alphabet of L as the universal set.

We now introduce some algebraic operations for languages.

For two languages L and K we define their *concatenation* as

$$L \cdot K = \{wv \mid w \in L,\ v \in K\}\,.$$

and the *Kleene closure* L^* (of L) by

$$\begin{aligned} L^0 &= \{\lambda\}\,, \\ L^{i+1} &= L^i \cdot L \quad \text{for} \quad i \geq 0\,, \\ L^* &= \bigcup_{i \geq 0} L^i\,. \end{aligned}$$

[2] We note the the order used in lexicons, dictionaries etc. differs from the lexicographic order defined above since we first order by length. If we would not do so, then we would start with λ, a, aa, aaa, ... (i. e., with an infinite sequence of words containing only as, which makes no sense. However, since in practice there is no word containing more than three equal letters in succession, in lexicons it is not necessary to order first by length.

The *positive Kleene closure* is defined by

$$L^+ = \bigcup_{i \geq 1} L^i \,.$$

For

$$L_1 = \{ab, ac\} \quad \text{and} \quad L_2 = \{ab^n a : n \geq 1\},$$

we get

$$\begin{aligned}
L_1 \cdot L_2 = L_1^2 &= \{abab, abac, acab, acac\},\\
L_1 \cdot L_2 &= \{abab^n a : n \geq 1\} \cup \{acab^n a : n \geq 1\},\\
L_2^3 &= \{ab^i aab^j aab^k a : i \geq 1, j \geq 1, k \geq 1\},\\
L_1^* &= \{ax_1 ax_2 \ldots ax_r : r \geq 1, x_i \in \{b, c\}, 1 \leq i \leq r\} \cup \{\lambda\},\\
L_2^+ &= \{ab^{s_1} aab^{s_2} a \ldots ab^{s_t} a : t \geq 1, s_j \geq 1, 1 \leq j \leq t\}.
\end{aligned}$$

From the algebraic point of view, L^+ is the smallest set which contains L and is closed with respect to the product of words, i. e., L^+ is the smallest semigroup containing L. Analogously, L^* is the smallest monoid containing L.

We note that, by definition, $L^* = L^+ \cup L^0 = L^+ \cup \{\lambda\}$ always holds, whereas $L^+ = L^* \setminus \{\lambda\}$ only holds, if $\lambda \notin L$.

Let us consider the special case where L only consists of the letters of an alphabet X. Then for any non-negative integer n, L^n consists of all words of length n over X. Thus L^* and L^+ are the sets of all words over X and all non-empty words over X, respectively. This gives a justification for the notation we introduced in the very beginning.

Let X and Y be two alphabets. A *homomorphism* $h : X^* \to Y^*$ is a mapping where

$$h(wv) = h(w)h(v) \text{ for any two words } w, v \in X^* \,. \tag{1.1}$$

From (1.1) and $w = w \cdot \lambda$ for all $w \in X^*$, we immediately obtain $h(w) = h(w)h(\lambda)$ for all $w \in X^*$, which implies $h(\lambda) = \lambda$. Obviously, a homomorphism can be given by the images $h(a)$ of the letters $a \in X$; an extension to words by

$$h(a_1 a_2 \ldots a_n) = h(a_1)h(a_2)\ldots h(a_n)$$

follows from the homomorphism property (1.1).

A homomorphism $h : X^* \to Y^*$ is called non-erasing if $h(a) \neq \lambda$ for all $a \in X$.

We extend the homomorphism to languages by

$$h(L) = \{h(w) \mid w \in L\} \,.$$

If h is a homomorphism, then the *inverse homomorphism* h^{-1} applied to a language $K \subseteq Y^*$ is defined by

$$h^{-1}(K) = \{w \mid w \in X^*,\ h(w) \in K\} \,.$$

Let the homomorphisms h_1 and h_2 mapping $\{a, b\}^*$ to $\{a, b, c\}^*$ be given by

$$h_1(a) = ab,\ h_1(b) = bb \quad \text{and} \quad h_2(a) = ac,\ h_2(b) = \lambda.$$

Obviously, h_1 is non-erasing. We get

$$h_1(abba) = abbbbbab,\ h_1(bab) = bbabbb,\ h_2(abba) = acac),\ h_2(bab) = ac$$

and

$$\begin{aligned}
h_1(\{a^n \mid n \geq 0\} \cup \{b^n \mid n \geq 0\}) &= \{(ab)^n \mid n \geq 0\} \cup \{b^{2n} \mid n \geq 0\},\\
h_2(\{a^n \mid n \geq 0\} \cup \{b^n \mid n \geq 0\}) &= \{(ac)^n \mid n \geq 0\}\\
&\quad \text{(the powers of } b \text{ only give the empty word)},\\
h_1(\{a^n b^n \mid n \geq 1\}) &= \{(ab)^n b^{2n} \mid n \geq 1\},\\
h_2(\{a^n b^n \mid n \geq 1\}) &= \{(ac)^n \mid n \geq 1\},\\
h_1^{-1}(\{ab^n \mid n \geq 1\}) &= \{ab^n \mid n \geq 0\},\\
h_2^{-1}(\{ac, acac\}) &= \{b^i a b^j \mid i \geq 0, j \geq 0\} \cup \{b^r a b^s a b^t \mid r \geq 0, s \geq 0, t \geq 0\},\\
h_1^{-1}(\{a^n b^n \mid n \geq 1\}) &= \{a\},\\
h_2^{-1}(\{a^n b^n \mid n \geq 1\}) &= \emptyset.
\end{aligned}$$

For any homomorphism h and any letter a, $h(a)$ is a uniquely determined word. We extend the notion by dropping this property.

A mapping $\sigma : X^* \to 2^{Y^*}$ is called a *substitution* if the following relations hold:

$$\begin{aligned}
&\sigma(\lambda) = \{\lambda\}\,,\\
&\sigma(xy) = \sigma(x)\sigma(y) \text{ for } x, y \in X^*\,.
\end{aligned}$$

In order to define a substitution it is sufficient to give the sets $\sigma(a)$ for any letter $a \in X$. Then we can determine $\sigma(a_1 a_2 \dots a_n)$ for a word $a_1 a_2 \dots a_n$ with $a_i \in V$ for $1 \leq i \leq n$ by

$$\sigma(a_1 a_2 \dots a_n) = \sigma(a_1)\sigma(a_2)\dots\sigma(a_n)$$

which is a generalization of the second relation in the definition of a substitution. Moreover, for a language $L \subset X^*$, we set

$$\sigma(L) = \bigcup_{x \in L} \sigma(x)\,.$$

For the substitutions σ_1 and σ_2 from $\{a, b\}^*$ in $\{a, b\}^*$ given by

$$\sigma_1(a) = \{a^2\},\ \sigma_1(b) = \{ab\} \quad \text{and} \quad \sigma_2(a) = \{a, a^2\},\ \sigma_2(b) = \{b, b^2\},$$

we obtain

$$\begin{aligned}
\sigma_1(\{aba, aa\}) &= \{a^2aba^2, a^2a^2\} = \{a^3ba^2, a^4\},\\
\sigma_2(\{aba, aa\}) &= \{aba, a^2ba, aba^2, a^2ba^2, ab^2a, a^2b^2a, ab^2a^2, a^2b^2a^2, aa, a^2a, aa^2, a^2a^2\}\\
&= \{aba, a^2ba, aba^2, a^2ba^2, ab^2a, a^2b^2a, ab^2a^2, a^2b^2a^2, a^2, a^3, a^4\}.
\end{aligned}$$

Let $\mathcal{L}$ be a family of languages. A substitution $\sigma : X^* \to Y^*$ is called a *substitution by sets of* $\mathcal{L}$, if $\sigma(a) \in \mathcal{L}$ holds for any $a \in X$.

If a substitution σ maps X^* to X^*, then we can apply σ to $\sigma(L)$, again, i. e., we can iterate the application of σ. Formally, this is defined by

$$\begin{aligned} \sigma^0(x) &= \{x\}, \\ \sigma^{n+1}(x) &= \sigma(\sigma^n(x)) \text{ for } n \geq 1. \end{aligned}$$

For a word $w = a_1 a_2 \dots a_n$ with $n \geq 0$ and $a_i \in V$ for $1 \leq i \leq n$, we set $w^R = a_n a_{n-1} \dots a_1$. The word w^R is called the *mirror image* or *reversal* of w. It is obvious that $\lambda^R = \lambda$ and $(w_1 w_2)^R = w_2^R w_1^R$ for any two words w_1 and w_2. For a language L, we set $L^R = \{w^R \mid w \in L\}$.

The concatenation or product of two words u and v gives the word uv. In arithmetics, the inverse operation is the quotient. An analog would be to consider v as the left quotient of uv and u and u as the right quotient of uv and v. Therefore cancellation of prefixes or suffixes can be regarded as the analog of the quotient. We give the notion for sets. For two languages L and L', we define the *right* and *left quotient* by

$$D_l(L, L') = \{v \mid uv \in L \text{ for some } u \in L'\}$$

and

$$D_r(L, L') = \{u \mid uv \in L \text{ for some } v \in L'\},$$

respectively. For example, for

$$L = \{a^n b^n \mid n \geq 1\} \text{ and } L' = \{a^n \mid n \geq 1\},$$

we get

$$D_l(L, L') = \{a^m b^n \mid m \geq 0, n \geq 1, n \geq m\} \text{ and } D_r(L, L') = \emptyset.$$

A *multiset* M over V is a mapping of V^* into the set $\mathbb{N}_0$ of non-negative integers. $M(x)$ is called the multiplicity of x. The cardinality and the length of a multiset M are defined as

$$\#(M) = \sum_{x \in V^*} M(x) \text{ and } l(M) = \sum_{x \in V^*} M(x)|x|.$$

A multiset M is called finite iff there is a finite subset U of V^* such that $M(x) = 0$ for $x \notin U$. Then its cardinality is the sum of the multiplicities of the elements of U. A finite multiset M can be represented as a "set" where M contains $M(x)$ occurrences of x. Thus a finite multiset M in this representation consists of $\#(M)$ elements. For example, the multiset M over $V = \{a, b\}$ with $M(a) = M(b) = M(aba) = 1$, $M(ab) = M(ba) = 2$ and $M(x) = 0$ in all other cases can be represented as $M = [a, b, ab, ab, ba, ba, aba]$[3]. Obviously, as for sets, the order of the elements in the multiset M is not fixed and can be changed without changing the multiset. For a multiset $M = [w_1, w_2, \dots, w_n]$ (in such a representation) we have $l(M) = |w_1 w_2 \dots w_n|$. Moreover, for a multiset M over V and $a \in V$, we set $\#_a(M) = \#_a(w_1 w_2 \dots w_n)$.

[3]We use the brackets [and] instead of { and } in order to distinguish multisets from sets.

1.2 Polynomials and Linear Algebra

A function $p : \mathbb{R} \to \mathbb{R}$ is called a *polynomial* (over the real numbers) if

$$p(x) = a_n x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_2 x^2 + a_1 x + a_0, \tag{1.2}$$

for some $n \in \mathbb{N}_0$ and $a_i \in \mathbb{R}$ for $0 \leq i \leq n$. The number n is called the *degree* of n, and the reals a_i are called the *coefficients* of p.

A (complex) number α is called a *root* of a polynomial p if $p(\alpha) = 0$. If p is a polynomial with m different roots α_i, $1 \leq i \leq m$, then there are natural numbers $t_i \in \mathbb{N}$, $1 \leq i \leq m$, such that

$$p(x) = (x - \alpha_1)^{t_1} \cdot (x - \alpha_2)^{t_2} \cdot \cdots \cdot (x - \alpha_m)^{t_m} \text{ and } n_1 + n_2 + \cdots + n_m = n.$$

For $1 \leq i \leq m$, the number t_i is called the *multiplicity* of the root α_i.

Theorem 1.1 *Let $a_n x^n + a_{n-1}x^{n-1} + a_{n-2}x^{n-2} + \cdots + a_2 x^2 + a_1 x + a_0$ be a polynomial of degree n with the roots α_i of multiplicity t_i, $1 \leq i \leq s$, and $\sum_{i=1}^{s} t_i = n$. Then the linear difference equation*

$$a_n f(m+n) + a_{n-1} f(m+n-1) + \cdots + a_2 f(m+2) + a_1 f(m+1)x + a_0 f(m) = 0$$

for $m \geq 0$ has the solution

$$f(m) = \sum_{i=1}^{s} (\beta_{i,0} + \beta_{i,1} m + \beta_{i,2} m^2 + \ldots \beta_{i,t_i-1} m^{t_i-1}) \alpha_i^m$$

with certain constants $\beta_{i,j}$, $1 \leq i \leq s$, $0 \leq j \leq t_i - 1$. □

A (m,n)-*matrix* M is a scheme of $m \cdot n$ (real) numbers $a_{i,j}$, $1 \leq i \leq m$ and $1 \leq j \leq n$. The scheme consists of m rows where the i-th row consists of the elements $a_{i,1}, a_{i,2}, \ldots, a_{i,n}$, $1 \leq i \leq m$. Equivalently, it is given by n columns where the j-th column is built by the numbers $a_{1,j}, a_{2,j}, \ldots, a_{m,j}$, $1 \leq j \leq n$. Thus we get

$$M = \begin{pmatrix} a_{1,1} & a_{1,2} & a_{1,3} & \ldots & a_{1,n} \\ a_{2,1} & a_{2,2} & a_{2,3} & \ldots & a_{2,n} \\ \cdot & \cdot & \cdot & \cdots & \cdot \\ a_{m,1} & a_{m,2} & a_{m,3} & \ldots & a_{m,n} \end{pmatrix}$$

We write $M = (a_{i,j})_{m,n}$ and omit the index m, n if the size of the matrix is known from the context. The numbers $a_{i,j}$ are called *coefficients* of the matrix M.

Obviously, row vectors are $(1,n)$-matrices and column vectors are $(m,1)$-matrices. A matrix is called a *square* matrix, if it is an (n,n)-matrix for some n. Let $E_{n,n}$ be the square (n,n)-matrix with $a_{i,i} = 1$ for $1 \leq i \leq n$ and $a_{j,k} = 0$ for $j \neq k$ (again, we omit the index if the size is understood by the context); $E_{n,n}$ is called the *unity matrix*. By O we denote the *zero matrix* where all entries are the real number 0.

Let $M_1 = (a_{i,j})_{m,n}$ and $M_2 = (b_{k,l})_{r,s}$ be two matrices, and let d be a (real) number. Then the *product* $d \cdot M_1$ is defined by

$$d \cdot M_1 = (d \cdot a_{i,j})_{m,n}\,.$$

The *sum* $M_1 + M_2$ is defined iff $m = r$ and $n = s$ by setting

$$M_1 + M_2 = (a_{i,j} + b_{i,j})_{m,n} .$$

The *product* $M_1 \cdot M_2$ is defined iff $n = r$ by setting

$$M_1 \cdot M_2 = (\sum_{j=1}^{n} a_{i,j} b_{j,l})_{m,s} . \tag{1.3}$$

The *transposed matrix* $(M_1)^T$ is formed by interchanging the rows and columns, i. e.,

$$(M_1)^T = (a_{j,i})_{n,m} .$$

The *determinant* of an (n, n)-matrix M is defined by

$$\det(M) = \sum_{p=(i_1,i_2,\ldots,i_n)} (-1)^{I(p)} a_{1,i_1} a_{2,i_2} \ldots a_{n,i_n}$$

where the sum is taken over all permutations of $1, 2, \ldots, n$. By definition, det maps matrices to reals.

The *characteristic polynomial* $\chi_A(x)$ of a (square) (n, n)-matrix A is defined as

$$\chi_A(x) = \det(A - xE) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \cdots + a_2 x^2 + a_1 x + a_0 .$$

We note that $a_n = (-1)^n$ and $a_0 = \det(A)$.

A complex number μ is called an *eigenvalue* of the square matrix A iff $\det(A - \mu E) = 0$, i. e., iff μ is a root of χ_A.

The following theorem is named after the English mathematicians Cayley[4] and Hamilton[5].

Theorem 1.2 *For any square matrix A, $\chi_A(A) = O$.* □

If we give a complete writing of the characteristic polynomial $\chi_A(A)$, then this means

$$\chi_A(A) = a_n A^n + a_{n-1} A^{n-1} + a_{n-2} A^{n-2} + \cdots + a_2 A^2 + a_1 A + a_0 E = O .$$

1.3 Graph Theory

A *directed graph* is a pair $G = (V, E)$ where V is a finite non-empty set and E is a subset of $V \times V \setminus \{(v, v) \mid v \in V\}$. The elements of V are called *vertices* or *nodes*; the elements of E are called *edges*. We note that, by our definition, a graph does not contain loops, i. e., edges connecting a node u with itselves, and no multiple edges since E is a set instead of a multiset.

A directed graph $H = (U, F)$ is called a subgraph of the directed graph $G = (V, E)$, if U is a subset of V and F is the restriction of E to $U \times U$.

[4] Arthur Cayley, 1821–1895

[5] William Rowan Hamilton, 1805–1865

A graphic representation of a graph can be given as follows: We interpret the vertices as "small" circles in a plane, and we draw a (directed) line from u to v if there is an edge (u, v).

A directed *path* from a node u to a node v is a sequence $u_0, u_2, \ldots, u_n$, $n \geq 0$, of nodes such that $u = u_0$, $u_n = v$ and $(u_i, u_{i+1}) \in E$ for $0 \leq i \leq n-1$. If there is a path from u to v, we say that u and v are connected in G. By $n = 0$, we ensure that u is connected with u. The non-negative number n is called the length of the path.

A directed graph is called a *directed tree*, if there is no edge u such that there is a path of length $n \geq 1$ from u to u.

A task which has to be solved very often is the determination of all nodes which are connected with a given node u in a given graph G. Two algorithms to solve this problem are *depth-first-search(G,u)* and *breadth-first-search(G,u)*, where all nodes connected with u are marked, which can be given by

1. Mark u.
2. For all nodes w with $(u, w) \in E$ such that w is not marked, do *depth-first-search(G,w)*.

and

1. Mark u and put u in a queue Q.
2. While Q is not empty, do the following steps:
   (a) Cancel the first element w of Q.
   (b) For all nodes z with $(w, z) \in E$ such that z is not marked, mark z and put z into the queue Q.

respectively. It is easy to see that both algorithm do a finite number of steps for each node and each edge of the graph. Hence we have

$$t_{depth-first-search(G,u)} \in O(\#(V) + \#(E)) \text{ and } t_{breadth-first-search(G,u)} \in O(\#(V) + \#(E)).$$

In many applications of graph, the edges describe a connection between the nodes which has no direction. Therefore undirected graphs have also been introduced.

A *undirected graph* is a pair $G = (V, E)$ where V is a finite non-empty set and E is a set of two-element subsets of V. The elements of V and E are also called nodes and edges. Instead of an directed edges (u, v) we have sets $\{u, v\}$ in an undirected graph.

The notions of a subgraph, of a path and of a tree can easily be transferred to undirected graphs.

Let $G = (V, E)$ be an undirected graph. The *degree* $d(u)$ of a node u is the number of nodes v such that $\{u, v\} \in E$.

We define some special undirected graphs.

- An undirected graph $G = (V, E)$ is called *k-regular* if and only if all nodes of V have the degree k, i. e., $d(u) = k$ for all $u \in V$.
- An undirected graph $G = (V, E)$ is called *regular* if and only if it is k-regular for some k.
- A 2-regular graph $G = (V, E)$ is also called a *simple closed curve*.
- An undirected graph $G = (V, E)$ is called a *simple curve*, if all its nodes have a degree at most 2.
- An undirected graph $G = (V, E)$ is called *Eulerian* if there is a path of length $\#(E)$ which contains any edge of E.

- An undirected graph $G = (V, E)$ is called *Hamiltonian*, if there is a path without repetitions of length $\#(V) - 1$.
- An undirected graph $G = (V, E)$ is called edge-colourable by k colours, if there is a mapping from E to $\{1, 2 \ldots, k\}$ such that, for any nodes three nodes $u, v_1, v_2 \in V$ with $\{u, v_1\} \in E$ and $\{u, v_2\} \in E$, $\{u, v_1\}$ and $\{u, v_2\}$are mapped to different numbers.
- An undirected graph $G = (V, E)$ is called *bipartite*, if there is a partition of V into two sets V_1 and V_2 (i. e., $V = V_1 \cup V_2$ and $V_1 \cap V_2 = \emptyset$) such that, for any edge $\{u, v\} \in E$, $\{u, v\} \notin V_1$ and $\{u, v\} \notin V_2$, i. e., any edge connects an element of V_1 with an element of V_2.

The following facts are known.

- A graph $G = (V, E)$ is Eulerian if and only if
  - all nodes of V have an even degree or
  - there are two nodes u and v in V such that u and v have an odd degree and all nodes of V different from u and v have even degree.
- A graph $G = (V, E)$ is Hamiltonian if and only if it contains a subgraph H which is a simple curve and contains all nodes of G.
- A graph is bipartite if and only if it is edge-colourable with two colours.

1.4 Intuitive Algorithms

An *intuitive algorithm*

- transforms input data in output data,
- consists of a finite sequence of commands such that
  - there is a uniquely determined command which has to be performed first,
  - after the execution of a command there is a uniquely determined command which has performed next, or the algorithm stops.

We define the running time $t_{\mathcal{A}}(w)$ of an algorithm $\mathcal{A}$ on an input w as the number of commands (or steps) performed by the algorithm on input w. Therefore, we assume that a command can be executed in one time unit. This is not satisfied in reality; for instance, the multiplication of two integers requires much more time than the addition of two integers. Moreover, the exact running time of single commands depends on the implementation of the commands, the used data structures etc. However, if c is the maximal running time of the execution of a single command, then the realistic running time of $\mathcal{A}$ on w is bounded by $c \cdot t_{\mathcal{A}}(w)$. Thus $t_{\mathcal{A}}(w)$ can be considered as a useful approximation of the real running time, and it is independent of the special features of implementation.

Now let M and M' be two sets. Moreover, let $k : M \to \mathbb{R}$ and $k' : M' \to \mathbb{R}$ be two functions which associate with each element of M and M', respectively, a size of the element. Furthermore, let $\mathcal{A}$ be an algorithm which transforms an element $m \in M$ into an element $\mathcal{A}(m) \in M'$. Then we set

$$t_{\mathcal{A}}(n) = \max\{t_{\mathcal{A}}(m) \mid m \in M,\ k(m) = n\}$$

and

$$u_{\mathcal{A}}(n) = \max\{k'(\mathcal{A}(m)) \mid m \in M,\ k(m) = n.\}$$

$t_{\mathcal{A}}(n)$ and $u_{\mathcal{A}}(n)$ give the maximal running time and the maximal size obtained from element of M with size n. Since we do not require that there is an element of size n for any n, $t_{\mathcal{A}}$ and $u_{\mathcal{A}}$ are not defined for all natural numbers.

As an example, let us consider the sets

$$M = \{(A, B) \mid A \text{ and } B \text{ are } (m, m)\text{-matrices}, \ m \in \mathbb{N}\}$$

and

$$M' = \{A \mid A \text{ is an } (m, m)\text{-matrix}, \ m \in \mathbb{N}\}.$$

If A and B are (m, m)-matrices for some are $m \in \mathbb{N}$, then we set

$$k((A, B)) = 2m^2 \text{ and } k'(A) = m^2,$$

i. e., we take the number of numbers contained in the matrices as the size. Let $\mathcal{A}$ be the algorithm which computes the product $A \cdot B = \mathcal{A}((A, B)$ according to (1.3). Obviously, $\mathcal{A}$ transforms inputs from M into outputs of M'.

Let A and B be two $m, m)$-matrices. Then $k'(\mathcal{A}((A, B)) = m$. Since the calculation of one element of $A \cdot B$ requires m multiplications and $m - 1$ additions of numbers and we have to compute m^2 elements, we get $t_{\mathcal{A}}((A, B)) = (2m - 1)m^2$. We note that, for a given n, the running time and the size of the product are identical for all pairs (A, B) with $k((A, B)) = n$ (i. e., A and B are (m, m)-matrices with $n = 2m^2$). Thus we have

$$\begin{aligned} t_{\mathcal{A}}(n) &= \max\{t_{\mathcal{A}}((A, B)) \mid A \text{ and } B \text{ are } (m, m)\text{-matrices and } n = 2m^2\} \\ &= \max\{2m^3 - m^2 \mid n = 2m^2\} \\ &= \frac{n^{\frac{3}{2}}}{\sqrt{2}} - \frac{n}{2} \end{aligned}$$

and

$$u_{\mathcal{A}}(n) = \max\{k'(\mathcal{A}((A, B)) \mid A \text{ and } B \text{ are } (m, m)\text{-matrices and } n = 2m^2\} = m^2 = \frac{n}{2}.$$

In most cases it is very hard to determine the functions $t_{\mathcal{A}}$ and $u_{\mathcal{A}}$ and it is sufficient to give upper bounds for these functions which can be considered as good approximations. Formally, for a function $f : \mathbb{N} \to \mathbb{N}$, we set

$$\begin{aligned} O(f) = \{g \mid g : \mathbb{N} \to \mathbb{N}, \ &\text{there are a real number } c > 0 \text{ and an } n_0 \in \mathbb{N} \\ &\text{such that } g(n) \leq c \cdot f(n) \text{ for all } n \geq n_0\}. \end{aligned}$$

Intuitively, the set $O(f)$ consists of all functions which differ from f by a multiplicative factor. Therefore, in many cases, it is sufficient to use functions f and g instead of the exact functions $t_{\mathcal{A}}$ and $u_{\mathcal{A}}$ such that $t_{\mathcal{A}} \in O(f)$ and $u_{\mathcal{A}} \in O(g)$.

Thus in the sequel, for short, we use the formulation that the algorithms works in time $O(f(k(m))$ and that $k'(\mathcal{A}(m)) \in O(g(k(m))$. This can be done since $n = k(m)$ and $k'(\mathcal{A}(m)) \leq u(n)$.

If the size depends on some parameters, then we take into considerations functions f and g which also depend on these parameters.